ERICSSON

CUDB Backup and Restore Procedures

USER GUIDE

1/1553-CNH 160 6130/10 Uen A

Contents

1 Introduction

This document describes the data, software and configuration backup and restore procedures available in the Ericsson Centralized User Database (CUDB).

1.1 Document Purpose and Scope

The CUDB software, configuration, and data stored in the database of a CUDB node can be backed up and optionally transferred to an external storage. This backup can be then used to restore CUDB in case of system failure. The purpose of this document is to cover the available CUDB backup and restore procedures allowing secure backup and restore of CUDB software, configuration and data when needed.

1.2 Revision Information

Rev. A

This document is based on 1/1553-CNH 160 6130/9 with the following changes:

- Removed obsolete hardware information throughout the document.
- Updated virtualization terminology throughout the document.
- Section 3.1.1.3 on page 5: Updated section to cover virtualized deployments.
- Section 3.2.1.1 on page 15: Updated description of unit data restore commands.
- Section 3.2.1.3 on page 18: Updated terminology.
- Section 3.3.1.2 on page 31: Added new information on backup and restore command.
- Section 4 on page 32: Updated scenarios where backup/restore operations cannot be performed. Updated the list of data included in the backup.
- Section 4.1 on page 34: Updated the description of the `SQL` file.
- Section 4.2.1 on page 35: Updated the `Alive` mode description.
- Section 4.2.2 on page 36: Updated Step 2 and Step 6.

1.3 Prerequisites

Before performing any of the operations described in this document, make sure that the solutions of the known issues described in the Release Notes are all available beforehand.

1.4 Typographic Conventions

Typographic conventions can be found in the following document:

- *Typographic Conventions*

2 Overview

This section provides an overview of the available backup and restore operations supported by the CUDB system.

The CUDB system supports following two main types of backups:

- Data backup and restore: This type of backup and restore contains the information stored in the database units of the CUDB System.
- Software and configuration backup and restore: This type of backup and restore contains the software and configuration information of the CUDB node.

2.1 Data Backup and Restore

A CUDB system is composed of several data partitions, including the Processing Layer (PL) Database (PLDB) and a number of Data Store Groups (DSGs). This type of backup and restore is data centric, and data is distributed on each CUDB node.

This backup is stored on the disk storage system inside the CUDB node, and is not exported automatically to any external storage area. If the data backup is to be kept in external storage, the backup files can be extracted.

The data backup and restore types are as follows:

- Unit:
  - Combined backup and restore
  - Backup
  - Restore
- Group:
  - Restore
- System:
  - Backup
  - Restore

Data backup and restore can recover the following typical scenarios:

- Out-of-synch replica of the PLDB or a DSG.

- Whole PLDB or DSG after failure in all of its replicas, or some failure scenarios.
- CUDB system after a general system failure.

See Section 3 on page 4 for more information on data backup and restore.

2.2 Software and Configuration Backup and Restore

This type of backup and restore is node driven, that is not distributed, and each CUDB node has its own software and configuration backup and restore.

This backup is stored on the disk storage system inside the CUDB node, and is not exported automatically to any external storage area. If the software and configuration backup is to be kept in external storage, the backup files can be extracted.

See Section 4 on page 32 for more information on SW and configuration backups.

3 Data Backup and Restore

This section describes the data backup and restore operations available in CUDB.

3.1 Data Backup

This section describes data backup operations available in CUDB.

3.1.1 Unit Data Backup

This section describes how to perform a unit data backup in the CUDB system.

3.1.1.1 Description

The unit data backup is used to create an individual unit data backup of any PLDB or DSG replica of the CUDB node.

Two unit data backups can be stored on the CUDB node at the same time. Subsequent unit data backups remove the oldest backup already stored, and keep the newer one.

3.1.1.2 Performing Unit Data Backup

To perform a unit data backup for any replica of the PLDB or DSG clusters, perform the following steps:

1. Establish a new administrative CUDB CLI session towards one of the System Controllers (SCs) of the CUDB node:

   ssh *<admin_user>@<CUDB_Node_OAM_VIP_Address>*

2. Find out the corresponding Data Store (DS) Unit for the DSG to backup executing:

   SC_2_1# **cudbManageStore --all --order status**

   Refer to *CUDB Node Commands and Parameters*, Reference [4] for more information on `cudbManageStore`.

3. Locate the CUDB node where the master replica is stored:

   SC_2_1# **cudbSystemStatus -R**

   Refer to *CUDB System Administrator Guide*, Reference [2] for more information.

4. Establish a new administrative CUDB CLI session towards one of the SCs of the CUDB node where the backup is needed:

   ssh *<admin_user>@<CUDB_Node_OAM_VIP_Address>*

5. Run the following command to back up the PLDB or a specific DSG cluster of the node:

   - SC_2_1# **cudbManageStore --pl --order backup**
   - SC_2_1# **cudbManageStore --ds *<dsId>* --order backup**

     In the command above, *<dsId>* stands for the identifier of the DS unit that needs the backup.

   Refer to *CUDB Node Commands and Parameters*, Reference [4] for more information on `cudbManageStore`.

6. Optionally, store backup data in an external storage. Refer to Section 3.1.1.3 on page 5 for more information.

3.1.1.3 Store Backup in an External Storage

For safety reasons, it is recommended to store backup files in an external storage.

Run the following command to copy the backup in an external storage:

```
SC_2_1# scp [-l <bandwidth>] <file> <destination>
```

Where:

- ***<bandwidth>*** represents a value to limit the bandwidth in kbit/s. In case the CUDB system is deployed on native BSP 8100 hardware on Generic Ericsson Processor version 3 (GEP3) nodes or in case the CUDB system is deployed on a cloud infrastructure with vCUDB_2CPU_6GB hardware type, it is recommended to set the value to 10 000 for stability reasons; in case the CUDB system is deployed on native BSP 8100 hardware on Generic Ericsson Processor version 5 (GEP5) nodes or in case the CUDB system is deployed on a cloud infrastructure with vCUDB_16CPU_47GB hardware type, it is recommended not to use the `-l` command-line switch to allow `scp` to use all the available bandwidth.

  Note: Avoid creating parallel `scp` copies of the different backup files.

- ***<file>*** represents the output backup files.

- ***<destination>*** represents the path where the output backup files will be stored in the external storage.

3.1.1.4 Alarms

In case any failure occurs during the backup phase, the system raises the *Storage Engine, Backup Fault In DS*, Reference [5] or *Storage Engine, Backup Fault In PLDB*, Reference [6] alarms.

3.1.1.5 Output

The `cudbManageStore` command creates a fresh backup of the data stored in the selected PLDB or DSG replica. The backup is made up of several files which are created in the following directory located on the disk storage system of every participating blade or Virtual Machine (VM):

```
/local/cudb/mysql/ndbd/backup/BACKUP/BACKUP-YYYY-MM-DD_HH
-mm
```

The identifiers of this participating blade or VM are as follows:

- For DSGs:

  PL_2_x, where x represents the following sequence:

  - from ***<Total_PLDB_blades_or_vm>*** + 1 + 2*****<dsId>***
  - to ***<Total_PLDB_blades_or_vm>*** + 2 + 2 *****<dsId>***.

- For the PLDB:

  PL_2_x, where x is the sequence:

- from 3
- to *<Total_PLDB_blades_or_vm>* + 2.

The backup components consist of the following files:

- Metadata file (`.ctl` extension)
- Table records file (`.data` extension)
- Transaction log file (`.log` extension)

Typically, the backup files are generated in all the blades or VMs of the database unit. However, in case of degraded clusters (refer to *CUDB High Availability*, Reference [3] for more information on degradation), the blades or VMs not participating in the backup do not generate the files, so these will be created in other blades or VMs of the cluster. See Section 3.2.1 on page 15 for further details on how these files are generated and moved to the slaves replicas.

3.1.2 System Data Backup

This section describes the system data backup procedures available in the CUDB system.

3.1.2.1 Description

A CUDB system is composed of several data partitions, including the PLDB and a number of DSGs. According to this structure, a system data backup consists of a number of pieces, each piece being the backup of an individual data partition of the whole CUDB system. One backup of the PLDB and one backup of each of the DSGs make up the full CUDB system data backup.

Refer to *CUDB Data Storage Handling*, Reference [1] for more information.

This type of backup contains the information stored in all database units of the CUDB system.

Unlike most Operation and Maintenance (OAM) operations that affect a single CUDB node, a full CUDB system data backup is a system-wide operation.

The criteria to choose the replica to be used to create the backup is the following:

- The backup of the PLDB is performed always in the master replica.
- The DSG backups are performed in a slave replica, if at least one is available, otherwise they are performed in the master replica.

The distribution of these pieces and the topology of the system makes system data backup to be a “data centric” backup.

To better understand this “data centric” concept, see Figure 1 showing the subscriber data distribution over the system.

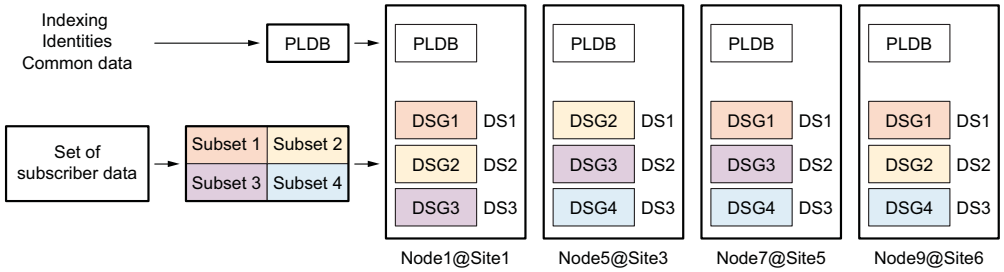

Figure 1 Data Distribution Example

Data backup of the whole CUDB system consists of the following:

- A backup of PLDB made of any node of the system.
- A backup of DSG1 made of node 1, 7, or 9.
- A backup of DSG2 made of node 1, 5, or 9.
- A backup of DSG3 made of node 1 ,5, or 7.
- A backup of DSG4 made of node 5, 7, or 9.

System data backup cannot be initiated in the following cases:

- No master PLDB replica exists.
- The master PLDB replica is degraded (refer to *CUDB High Availability*, Reference [3] for more information on degradation).
- All slave replicas and also the master replica of any DSGs are down or degraded in a geographically redundant system.

Note: In case Self-Ordered Backup and Restore is running on some nodes of the system, system data backup might fail for the DS or PLDB unit, which is used as backup source for the Self-Ordered Backup and Restore.

Refer to the ‘‘cudbSystemStatus’’ section of *CUDB Node Commands and Parameters*, Reference [4] for further information about the status of the Self-Ordered Backup and Restore process.

The execution of a backup does not affect regular traffic operations and traffic operations do not affect the backup. If provisioning continues during a system data backup, it can cause misalignment between the data in the Processing Layer (PLDB) backup and the data in the Data Store Layer (DSG) backup. This misalignment will only become apparent in the unlikely case of a system data restore and the amount of effected Lightweight Directory Access Protocol (LDAP) entries will be negligible compared to the data loss caused by the

restoration of the system data backup in most of the usage scenarios. The misalignment can be partially corrected by the reconciliation process. The remaining misalignment can be detected and corrected by application Front End (FE) level procedures.

Note: In case of group data restore and system data restore operations, reconciliation must be scheduled manually, unless the automatic execution of Selective Replica Check and Data Repair processes is disabled by configuration. Check the details of the `CudbDsGroupRepairAndResync` class in *CUDB Node Configuration Data Model Description*, Reference [8] for more information about configuring these processes.

CUDB offers the function of sending a notification to the Ericsson Provisioning Gateway (PG) when a system data backup is started and finished so that the PG can withhold provisioning operations towards the CUDB system while the backup is being created.

Although this is the default behavior, the operator must decide whether ensuring full consistency between PLDB and DSG backup is preferred or avoiding impact in provisioning every time system data backup is performed. If continuous provisioning is preferred then sending notifications to PG can be disabled by using command-line parameters in the `cudbSystemDataBackupAndRestore` command. For more information on command parameters for `cudbSystemDataBackupAndRestore`, refer to *CUDB Node Commands and Parameters*, Reference [4]. For instructions on how to schedule a backup without notifications, refer to Step 3.

Note: As a prerequisite for the notification sending function, CUDB must be configured with the necessary PG information. Refer to *CUDB System Administrator Guide*, Reference [2] for more information on configuring PG nodes. For alarms related to PG notifications, see Section 3.1.2.3 on page 10.

3.1.2.2 Performing System Data Backup

A system data backup in CUDB consists of the following steps:

1. Select one of the nodes in the CUDB system to perform the system data backup.

   Note: If the automatic backup file collection is invoked, the cluster area of the selected node is the target for the collection.

   Refer to *CUDB Node Commands and Parameters*, Reference [4] for more information on this command.

2. Establish a new administrative CUDB CLI session towards one of the SCs of the CUDB node mentioned in the previous step:

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

3. If previously made backup files remained in the output folders of the `cudbSystemDataBackupAndRestore` command, remove their contents to prevent potential failures. The output folders are as follows:

   - On every node of the CUDB system: `/home/cudb/systemDataBackup`
   - On the node where the `cudbSystemDataBackupAndRestore` command is performed: `/home/cudb/automatedBackupStorage`

   Run the following commands on an SC to remove the contents of the above folders:

   ```
   SC_2_1# ssh <CUDB_Node_OAM_VIP_Address> "rm -rf
   /home/cudb/systemDataBackup/*"
   ```

   ```
   SC_2_1# ssh <CUDB_Node_OAM_VIP_Address> "rm -rf
   /home/cudb/automatedBackupStorage/*"
   ```

4. Run the `cudbSystemDataBackupAndRestore` command as follows:

   ```
   SC_2_1# cudbSystemDataBackupAndRestore -c
   ```

   Refer to *CUDB Node Commands and Parameters*, Reference [4] for more information on this command.

5. Optionally, store backup data in an external storage. Refer to Section 3.1.1.3 on page 5 for more information.

Note: In case any failure occurs during the backup phase, see the alarms that could be raised in Section 3.1.2.3 on page 10.

3.1.2.3 Alarms

If it is not possible to inform the PG about the backup, the backup fails and the system raises the `Storage Engine, Backup Notification Failure To Provisioning Gateway` alarm. Refer to *Storage Engine, Backup Notification Failure To Provisioning Gateway*, Reference [7] for more information.

In case any failure occurs during the backup phase, the system raises the *Storage Engine, Backup Fault In DS*, Reference [5] or *Storage Engine, Backup Fault In PLDB*, Reference [6] alarms.

3.1.2.4 Output

The following output directories are created by the `cudbSystemDataBackupAndRestore` command:

- Temporary backup files and directories are created on those CUDB nodes where the unit data backups are taken in the directory:

/home/cudb/systemDataBackup/

- Those temporary backup files, distributed in the entire CUDB system, are copied automatically from those CUDB nodes to the CUDB node where cudbSystemDataBackupAndRestore was ordered, to the following output directory:

  /home/cudb/automatedBackupStorage/*<CUDBNodeId>*/*<DSG>*/*<BackupFileName>*

  Note: The automatic copy of the temporary backup files is performed only if the following conditions are met:

  - The CUDB node where cudbSystemDataBackupAndRestore was performed contains enough free space for the backups.
  - cudbSystemDataBackupAndRestore was performed without the -k | --keep-local switch.

  If any of these conditions are not met, the backup files are kept in the temporary directories, and they will not be copied to one single CUDB node.

  In the above path, the variables stand for the following:

  – *<CUDBNodeId>* is the CUDB Node Identifier of the node where the backed up data partitions belongs. Refer to *CUDB Node Configuration Data Model Description*, Reference [8] for further information about the CUDB Node Identifier.

  – *<DSG>* is the DSG Identifier of the DSG where the cluster belongs. If the value is 0, the backup belongs to the PLDB.

  – *<BackupFileName>* is the name of the backup file: BACKUP-*YYYY-MM-DD_HH-mm-<CUDBNodeId>.<DSG>.<mysqlndbId>*.tar

  This single directory is the final output directory of the system data backup, containing the copies of the system data backup files. For each data partition (PLDB and each of the DSGs) there will be as many TAR files as participating blades or VMs there were in the cluster where the corresponding unit backup was taken.

  Those TAR files contain the following:

  – Metadata file (.ctl extension)

  – Table records file (.data extension)

  – Transaction log file (.log extension)

A sample output of a full CUDB system data backup is shown below:

```
SC_2_2# cudbSystemDataBackupAndRestore -c
```

```
Checking /home/cudb/automatedBackupStorage on local node
--------------------------------


        Local node: ... EMPTY

Checking /home/cudb/systemDataBackup on nodes
--------------------------------


        44: ... EMPTY
        43: ... EMPTY

CREATE PART
--------------------------------


/home/cudb/systemDataBackup
Listening for current PLDB and DSGs status reports (may take upto 2 minutes)
Starting backup...
Before calling pg_notification
BackupStart [INFO] - PS Notification trying stop notification was successfully sent.
PS Notification successfully sent to : 172.31.233.139
PS Notification failed to : NONE
PS not notified : NONE


cudb_backup ver(1.3.4)

BEGIN MANAGEMENT FOR LAST BACKUP
Backup 2014-05-14_20-11 finished successfully in :
PLDB in CUDB node 44
    NDB node PL0
    NDB node PL1
    NDB node PL2
    NDB node PL3
DSG#1 in CUDB node 43
    NDB node DS1_0
    NDB node DS1_1
DSG#2 in CUDB node 43
    NDB node DS2_0
    NDB node DS2_1
Attempting to de-block Provisioning Gateway. This may take up to a couple of minutes.

COLLECT PART
--------------------------------


        Copying backup piece from node 44 for dsg 0 for ndb node id 3 ... OK
        Copying backup piece from node 44 for dsg 0 for ndb node id 4 ... OK
        Copying backup piece from node 44 for dsg 0 for ndb node id 5 ... OK
        Copying backup piece from node 44 for dsg 0 for ndb node id 6 ... OK
        Copying backup piece from node 43 for dsg 1 for ndb node id 3 ... OK
        Copying backup piece from node 43 for dsg 1 for ndb node id 4 ... OK
        Copying backup piece from node 43 for dsg 2 for ndb node id 3 ... OK
        Copying backup piece from node 43 for dsg 2 for ndb node id 4 ... OK


Please copy /home/cudb/automatedBackupStorage to a safe location to be able to
restore the backup pieces when needed.

ERROR SUMMARY
--------------------------------

None.
```

As a result of the example above (where automatic backup file collection was invoked), the following files are generated and stored in the `/home/cudb/automatedBackupStorage/` NFS directory:

For the PLDB backup:
`/home/cudb/automatedBackupStorage/44/0/BACKUP-2014-05-14_20-11-44.0.3.tar`

```
/home/cudb/automatedBackupStorage/44/0/BACKUP-2014-05-14_
20-11-44.0.4.tar
/home/cudb/automatedBackupStorage/44/0/BACKUP-2014-05-14_
20-11-44.0.5.tar
/home/cudb/automatedBackupStorage/44/0/BACKUP-2014-05-14_
20-11-44.0.6.tar
```

For DSG1 backup from node 43:

```
/home/cudb/automatedBackupStorage/43/1/BACKUP-2014-05-14_
20-11-43.1.3.tar
/home/cudb/automatedBackupStorage/43/1/BACKUP-2014-05-14_
20-11-43.1.4.tar
```

For DSG2 backup from node 43:

```
/home/cudb/automatedBackupStorage/43/255/BACKUP-2014-05-1
4_20-11-43.255.3.tar
/home/cudb/automatedBackupStorage/43/255/BACKUP-2014-05-1
4_20-11-43.255.4.tar
```

3.1.2.5 Scheduling a Periodic System Data Backup in a CUDB System

The CUDB system supports scheduled periodic data backups.

Note: Scheduled data backups must be configured only in one CUDB node of the system. If the CUDB node configured to perform periodic system data backups is down or isolated from the rest of the CUDB system at the time the periodic system backup is to be taken, the periodic system backup will not be executed. Do not schedule or perform other tasks in parallel with periodic data backups, except data consistency checks: system data backups have no collision with data consistency checks, so they can run in parallel.

Perform the following steps to configure a periodic data backup task:

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. Login to one of the SCs with the following command:

   ssh OAM*<X>*

   <X> stands for the SC number, which is 1 or 2.

3. Run the `cudbDataBackup` command with a `cron` expression as follows to schedule a CUDB data backup:

   SC_2_*<X>*# **cudbDataBackup -s *<CRON_EXPRESSION>***

The format of the *<CRON_EXPRESSION>* follows the standard UNIX cron expression format. For example, if the *<CRON_EXPRESSION>* is set to **'5 0 * * *'**, then the backup is launched at 0:05 every day.

Note: Any additional options for the scheduled backup must be stated after the `-s` option. For example, the below command schedules a backup at 0:05 every day, with parallel mode and PG notifications disabled:

```
cudbDataBackup -s '5 0 * * *' -q -L
```

4. Log in the other SC as described in Step 2, and configure the scheduled backup as described in Step 3. Configuring the periodic data backup on both SCs is necessary to make sure that the backup is performed even if one of them is unavailable.

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

   ```
   exit
   ```

3.1.2.6 Checking the Configured Periodic System Data Backups

Perform the following steps to print the scheduled data backups configured in the system:

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. Run the following command to check the scheduled backups:

   ```
   SC_2_<X># cudbDataBackup --print
   ```

3.1.2.7 Canceling a Periodic System Data Backup in a CUDB System

Perform the following steps to cancel a periodic data backup task:

1. Establish a new administrative CUDB CLI session towards the target CUDB node in which the periodic system data backup has been scheduled with the following command:

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

2. Log in one of the SCs with the following command:

   ```
   ssh OAM<X>
   ```

   <X> stands for the SC number, which is 1 or 2.

3. Run the `cudbDataBackup` command with a corresponding `cron` expression as follows to unschedule a CUDB data backup:

```
SC_2_<X># cudbDataBackup -U <CRON_EXPRESSION>
```

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

   ```
   exit
   ```

5. Disable the scheduled backup by repeating the same steps in the other SC.

3.2 Data Restore

This section describes how to perform a data restore in the CUDB system.

3.2.1 Unit Data Restore

This section describes how to perform a unit data restore in the CUDB system.

3.2.1.1 Description

CUDB unit data restore utilizes backup files generated by unit data backup to restore the PLDB or a specific DSG cluster.

After restoration, it is necessary to store SQL procedures as described in Section 3.2.1.3 on page 18.

Unit data restore commands also allow cluster backups to be restored on clusters which contain a different number of database nodes. This can happen in case of configuring a system on hybrid hardware, or if one of the involved clusters is crippled. In these cases, the following scenarios apply:

- If the cluster at the backup source has more blade or VM servers taking part in the backup and restore than the cluster targeted for restore (that is, the "target cluster"), the files for the missing blade or VM server(s) must be transferred to one of the working servers in the target cluster.
- If the cluster at the backup source has less blade or VM servers taking part in the backup and restore than the target cluster, no files must be transferred to the target blade or VM server equivalent to the missing server at the backup source.

Figure 2 shows the file distribution in case of a restoring the backup of a 4-database-node cluster to a 3-database-node cluster.

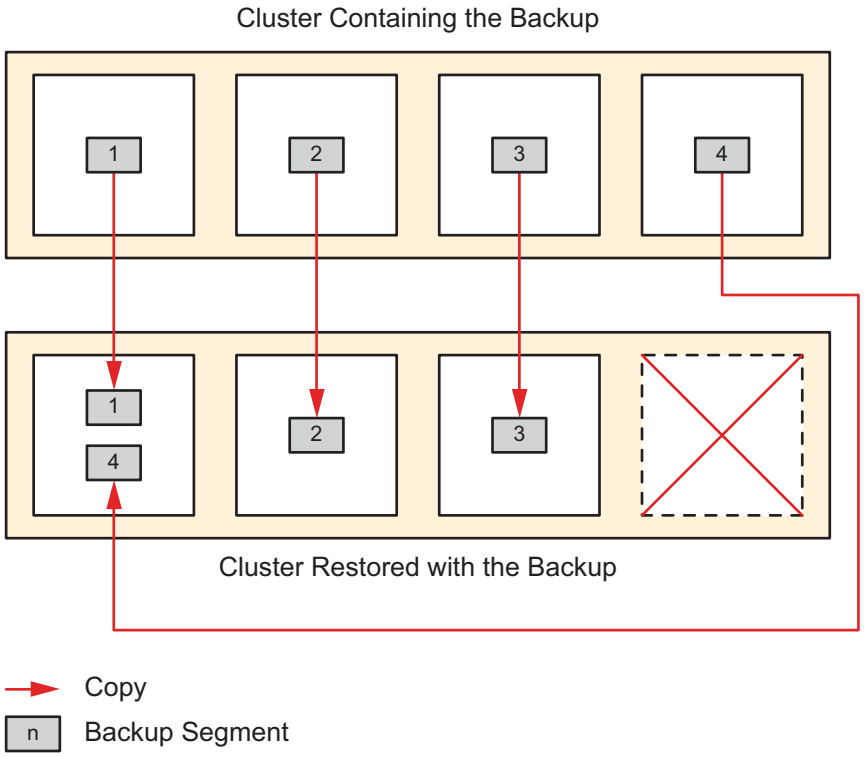

Figure 2 File Distribution for Restoring a 4-Database-Node Backup in a 3-Database-Node Cluster

Figure 3 shows the file distribution in case a 3-database-node cluster backup is restored on a 4-database-node cluster.

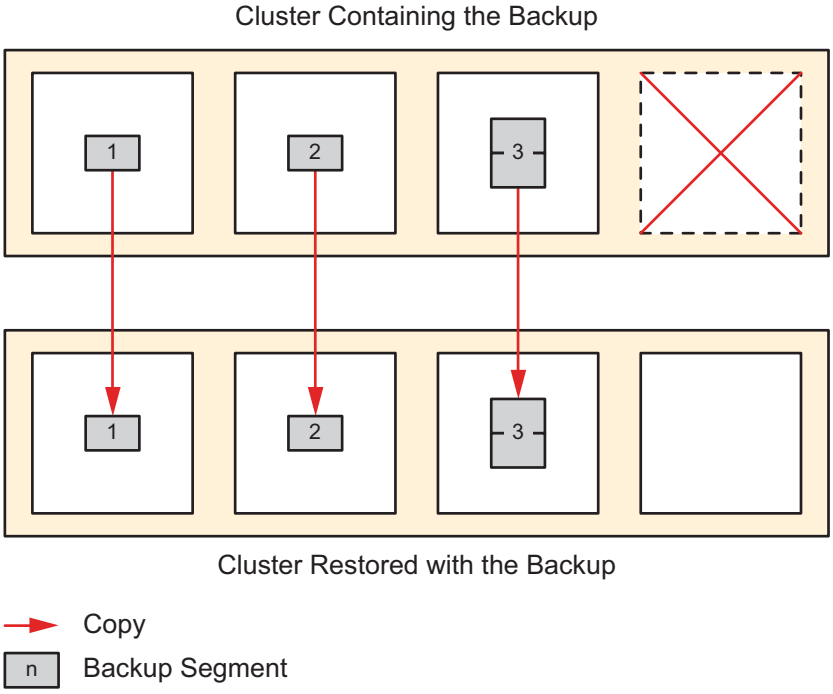

Figure 3 File Distribution for Restoring a 3-Database-Node Backup in a 4-Database-Node Cluster

Compared to data restores taking place between clusters where the number of blades or VMs is the same, the data restore procedure takes more time if the number of blades or VMs is different in the backup source and restore destination clusters.

3.2.1.2 Performing Unit Data Restore

Perform the following steps to restore a unit data backup:

1. Establish a new administrative CUDB CLI session with the following command towards the CUDB node with the master replica where the backup is stored:

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

2. Before restoration, these backup files must be transferred to the equivalent blades or VMs of the slave replica to be restored. See Section 3.1.1.5 on page 6 for further details. Use the directory of the equivalent blade or VM as destination. Consider the scenarios of the distribution in case of restoring a backup where source has different blades or VMs taking part in the backup and restore than the target cluster. See Figure 2 and Figure 3 for further details.

3. Run the `cudbManageStore` command as follows:

   - To restore the PLDB cluster, enter the following command:

```
SC_2_1# cudbManageStore --pl --order restore
--location <nameOfTheBackupDirectory> --restore-stored
-procedures
```

- To restore a specific DSG cluster, enter the following command:

  ```
  SC_2_1# cudbManageStore --ds <dsId> --order
  restore --location <nameOfTheBackupDirectory>
  --restore-stored-procedures
  ```

3.2.1.3 Recreating Stored Procedures After Data Restore

To restore the stored procedures and application counter procedures on a payload blade or VM, enter the following command:

```
shell>cudbManageStore -p -o restorestoredprocedures
```

To restore the stored procedures and application counter procedures on a DS, enter the following command:

```
shell>cudbManageStore -d <dsId> -o restorestoredprocedures
```

Note: The scripts of the stored SQL procedures are kept in the following location on all CUDB nodes:

```
/home/cudb/oam/performanceMgmt/appCounters/procedur
es/
```

3.2.1.4 Output

The output is the restoration of the PLDB or a specific DSG cluster in the CUDB node.

3.2.2 Group Data Restore

This section describes how to perform a group data restore in the CUDB system.

3.2.2.1 Description

Group data restore is basically a set of unit data restore procedures performed on all replicas of the affected data partition (PLDB or DSG). See Section 3.2.1 on page 15 for more information.

Following a successful group data restore, the replication channel must be synchronized properly.

Note: All replicas of the affected data partition must be restored from an earlier backup by performing a unit data restore on each replica belonging to the group, so make sure to use the same backup to restore all replicas.

Data in the restored partition is likely to be inconsistent with the rest of the CUDB system since the restored data is older than the rest. A group data restore in a DSG requires reconciliation for that DSG. A group data restore in PLDB requires reconciliation for all DSGs.

In case of group data restore and system data restore operations, reconciliation must be scheduled manually, unless the automatic execution of the Selective Replica Check and Data Repair processes is disabled by configuration. Check the details of the `CudbDsGroupRepairAndResync` class in *CUDB Node Configuration Data Model Description*, Reference [8] for more information about configuring these processes.

When no DSG database slave replicas are available, a group data restore is the same as a unit data restore in such deployments.

Warning!

Group data restore affects traffic. All subscribers whose data is stored on restored DSG or PLDB are impacted while the restore is running.

Warning!

The data restore procedure might change temporary subscriber data, such as location information. Refresh such data on the application FE level.

3.2.2.2 Performing Group Data Restore

To perform a group data restore, follow the steps below:

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

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

2. Prevent subscribers being created in the DSG that is to be restored, execute in any node in the CUDB system:

   ```
   cudbDsgProvisioningManage -d <dsgId>
   ```

3. Repeat the steps described in Section 3.2.1 on page 15 in all the replicas of the data partition, with the `--no-start-replication` option included in the `cudbManageStore` command.

   - To restore the PLDB cluster and the application counter procedures, enter the following command:

     ```
     SC_2_1# cudbManageStore --pl --order restore
     --location <nameOfTheBackupDirectory> --restore-stored
     -procedures --no-start-replication
     ```

   - To restore a specific DSG cluster and application counter procedures, enter the following command:

     ```
     SC_2_1# cudbManageStore --ds <dsId> --order
     restore --location <nameOfTheBackupDirectory>
     --restore-stored-procedures --no-start-replication
     ```

4. After the data restore is performed on all replicas of the data partition, make the slave replicas to connect and synchronize with the master replica as follows:

   Establish a new administrative CUDB CLI session with the following command towards the CUDB node with the slave replica:

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

   - To get the PLDB slaves synchronized, enter the following command:

     ```
     SC_2_1# cudbManageStore --pl --order kickstart
     ```

   - To get the DSG slaves synchronized, enter the following command:

     ```
     SC_2_1# cudbManageStore --ds <dsId> --order kickstart
     ```

5. To reallow subscribers to be created in the DSG, enter:

   ```
   cudbDsgProvisioningManage -e <dsgId>
   ```

6. If reconciliation must be scheduled manually (see the note in Section 3.2.2.1 on page 18 for more information), then schedule it for the specific DSG if it has been restored, or for all DSGs if the full PLDB group has been restored. Refer to the section on requesting reconciliation manually in *CUDB System Administrator Guide*, Reference [2] for more information.

3.2.2.3 Output

Group data restore affects the information stored in all replicas of the affected the data partition (PLDB or DSG) targeted for restore.

Data stored in other partitions are unaffected by group data restore.

3.2.3 System Data Restore

This section describes how to restore a system data backup in a CUDB system.

3.2.3.1 Description

This type of restore affects the information stored in all database units of the CUDB system.

Warning!

During system data restore, the CUDB system is unable to handle traffic.

Warning!

The data restore procedure might change temporary subscriber data, such as location information. Refresh such data on the application FE level.

Note: To avoid collision with the Self-Ordered Backup and Restore function, do not perform system data restore if a Self-Ordered Backup and Restore process is running on any node of the system.

Refer to the ‘‘cudbSystemStatus’’ section of *CUDB Node Commands and Parameters*, Reference [4] for further information about the status of the Self-Ordered Backup and Restore process.

System data restore with the `cudbSystemDataBackupAndRestore` command is available in two different methods. The main difference between these methods is the location of the backup files:

- System data can be restored from a single source.
- System data can also be restored from distributed sources.

The difference is related to the system data backup procedure which is available both with and without automatic file collection.

Note: The available system data restore methods are not exclusive to any system data backup methods. Therefore, for example, system data can be restored from a single source even if the system data backup was performed without automatic file collection.

The following sections describe how to restore system data with both methods. For the purposes of description, an example CUDB system is used, containing

three CUDB nodes with triple geographical redundancy configuration. The system contains three DSGs.

The nodes, DSGs and the DSG replicas selected as the source replica during the system data backup creation are indicated in Example 1.

```
PLDB in CUDB node 81    PLDB source replica was the replica that resides on Node 81
NDB node PL0            <-
NDB node PL1            <-
NDB node PL2            <-
NDB node PL3            <-
DSG#1 in CUDB node 81
NDB node DS1_0
NDB node DS1_1
DSG#2 in CUDB node 81   DSG2 source replica was the replica that resides on Node 81
NDB node DS2_0          <-
NDB node DS2_1          <-
DSG#3 in CUDB node 81
NDB node DS3_0
NDB node DS3_1

PLDB in CUDB node 82
NDB node PL0
NDB node PL1
NDB node PL2
NDB node PL3
DSG#1 in CUDB node 82   DSG1 source replica was the replica that resides on Node 82
NDB node DS3_0          <-
NDB node DS3_1          <-
DSG#2 in CUDB node 82
NDB node DS2_0
NDB node DS2_1
DSG#3 in CUDB node 82
NDB node DS1_0
NDB node DS1_1

PLDB in CUDB node 83
NDB node PL0
NDB node PL1
NDB node PL2
NDB node PL3
DSG#1 in CUDB node 83
NDB node DS2_0
NDB node DS2_1
DSG#2 in CUDB node 83
NDB node DS1_0
NDB node DS1_1
DSG#3 in CUDB node 83   DSG3 source replica was the replica that resides on Node 83
NDB node DS3_0          <-
NDB node DS3_1          <-
```

Example 1 Example CUDB System and Backup Fileset

Note: Each replica belonging to a DSG can be placed in different locations on each node. In the above example, each replica is located in different positions on each node.

3.2.3.2 Preparing System Data Restore

Before loading a backup to a CUDB node, prepare the system data restore by performing the steps of Section 3.2.3.2.1 on page 23 or Section 3.2.3.2.2 on page 24 (depending on how the source backup files are stored).

Note: System data restore can be performed only on a system:

- which contains nodes with the same node ID as in the system where the data backup set was created,
- where the same DSGs are configured on each node as in the system where the data backup set was created.

Also, from a restore perspective, it is irrelevant from which CUDB nodes the backup files were taken, as long as their CUDB Node Identifiers are part of the system where the restore is performed.

Finally, notice that automated system data restore works in fully operating systems only.

During the system restore process, each backup file is transferred automatically to all the CUDB nodes where the specific DSG - referred by the backup file - has a replica. The files are copied to the following NFS directory on the nodes:

```
/home/cudb/systemDataBackup/
```

After the copy is finished, the files are extracted and transferred automatically to the local directories of the respective CUDB blades or VMs (see Section 3.2 on page 15 for more information). By default, the local directory is the following:

```
/local/cudb/mysql/ndbd/backup/BACKUP
```

After the restore is performed, the slave PLDB and DSG replicas are kick-started, that is, forced to connect and synchronize with the master replica.

3.2.3.2.1 Restoring from a Single Source

In case system restore is performed from a single source, copy all backup files of the requested backup to the CUDB node where the system data restore is to be ordered. The guidelines for copying are the following:

- Copy backup files from their storage location to any NFS directory at the target CUDB node, for example here:

  ```
  /home/cudb/automatedBackupStorage/
  ```

- Within the NFS directory, the backup files must be copied to a relative path that is determined by the backup filenames as follows:

  /<CUDBNodeID>/<DSG>/<BackupFileName>

  In the above relative path, `<CUDBNodeID>` and `<DSG>` stand for the CUDB node ID and the DSG ID, which can be read from the backup file name. For more information about backup file names, see Section 3.1.2.4 on page 10.

Note: This relative path is similar to the one in `/home/cudb/automatedBackupStorage/`, where the files were originally placed during system data backup, if automatic file collection was used that time.

In case of restoring the example backup (described in Section 3.2.3.1 on page 21) from a single source, the backup files must be copied to an NFS directory (for example to `/home/cudb/automatedBackupStorage/`) with the following file names and paths:

	CUDB node:	**Path within CUDB node:**
PLDB backup files:	*<nodeID>*	*<NFS path>*/81/0/BACKUP-*<YYYY-MM-DD_HH-mm>*-81.0.3.tar
	<nodeID>	*<NFS path>*/81/0/BACKUP-*<YYYY-MM-DD_HH-mm>*-81.0.4.tar
	<nodeID>	*<NFS path>*/81/0/BACKUP-*<YYYY-MM-DD_HH-mm>*-81.0.5.tar
	<nodeID>	*<NFS path>*/81/0/BACKUP-*<YYYY-MM-DD_HH-mm>*-81.0.6.tar
DSG1 backup files:	*<nodeID>*	*<NFS path>*/82/1/BACKUP-*<YYYY-MM-DD_HH-mm>*-82.1.3.tar
	<nodeID>	*<NFS path>*/82/1/BACKUP-*<YYYY-MM-DD_HH-mm>*-82.1.4.tar
DSG2 backup files:	*<nodeID>*	*<NFS path>*/81/2/BACKUP-*<YYYY-MM-DD_HH-mm>*-81.2.3.tar
	<nodeID>	*<NFS path>*/81/2/BACKUP-*<YYYY-MM-DD_HH-mm>*-81.2.4.tar
DSG3 backup files:	*<nodeID>*	*<NFS path>*/83/3/BACKUP-*<YYYY-MM-DD_HH-mm>*-83.3.3.tar
	<nodeID>	*<NFS path>*/83/3/BACKUP-*<YYYY-MM-DD_HH-mm>*-83.3.4.tar

In the above example, *<nodeID>* is the ID of the selected CUDB node (same for all files), while *<NFS path>* is the NFS directory (also the same for all files).

3.2.3.2.2 Restoring from a Distributed Source

Copy all backup files of the requested backup to the CUDB node where the file was generated during the system data backup procedure. The destination node of each backup file is indicated in the filename itself (see Section 3.1.2.4 on page 10 for the backup file name template). Copy the backup files from their storage location to the following directory of the target CUDB node:

/home/cudb/systemDataBackup

In case of restoring the example backup (described in Section 3.2.3.1 on page 21) from distributed sources, the backup files must be copied to the directory mentioned above with the following file names and the following relative paths:

	CUDB node:	Path within CUDB node:
PLDB backup files:	81	/home/cudb/systemDataBackup/BACKUP-<*YYYY-MM-DD_HH-mm*>-81.0.3.tar
	81	/home/cudb/systemDataBackup/BACKUP-<*YYYY-MM-DD_HH-mm*>-81.0.4.tar
	81	/home/cudb/systemDataBackup/BACKUP-<*YYYY-MM-DD_HH-mm*>-81.0.5.tar
	81	/home/cudb/systemDataBackup/BACKUP-<*YYYY-MM-DD_HH-mm*>-81.0.6.tar
DSG1 backup files:	82	/home/cudb/systemDataBackup/BACKUP-<*YYYY-MM-DD_HH-mm*>-82.1.3.tar
	82	/home/cudb/systemDataBackup/BACKUP-<*YYYY-MM-DD_HH-mm*>-82.1.4.tar
DSG2 backup files:	81	/home/cudb/systemDataBackup/BACKUP-<*YYYY-MM-DD_HH-mm*>-81.2.3.tar
	81	/home/cudb/systemDataBackup/BACKUP-<*YYYY-MM-DD_HH-mm*>-81.2.4.tar
DSG3 backup files:	83	/home/cudb/systemDataBackup/BACKUP-<*YYYY-MM-DD_HH-mm*>-83.3.3.tar
	83	/home/cudb/systemDataBackup/BACKUP-<*YYYY-MM-DD_HH-mm*>-83.3.4.tar

3.2.3.3 Performing System Data Restore

Perform the steps of Section 3.2.3.3.1 on page 25 or Section 3.2.3.3.2 on page 26 (depending on how the source backup files are stored) to perform system data restore.

3.2.3.3.1 Restoring from a Single Source

Log on to the CUDB node where the backup files were copied earlier (see Section 3.2.3.2 on page 22) and execute the following command:

```
SC_2_1# cudbSystemDataBackupAndRestore -r <backupDir>
```

In the above command, ***<backupDir>*** is the folder where backup files are stored, for example the following:

```
/home/cudb/automatedBackupStorage/
```

Refer to *CUDB Node Commands and Parameters*, Reference [4] for more information on this command.

Note: In case any failure occurs, the alarms described in Section 3.2.3.5 on page 26 could be raised. After the restoration, all stored procedures related to the application counter process are lost, and therefore must be created again. See Section 3.2.1.3 on page 18 for more information.

3.2.3.3.2 Restoring from a Distributed Source

Log on to any of CUDB nodes in the CUDB system, and run the following command:

```
SC_2_1# cudbSystemDataBackupAndRestore -R <backup_date>
```

In the above command, ***<backup_date>*** is the date when the backup was created. The date of creation can be checked from any of the backup filenames (see Section 3.1.2.4 on page 10 for the backup filename template).

Example of backup date:

```
2014-05-14_20-11
```

Refer to *CUDB Node Commands and Parameters*, Reference [4] for more information on this command.

Note: In case any failure occurs, the alarms described in Section 3.2.3.5 on page 26 could be raised. After the restoration, all stored procedures related to the application counter process are lost, and therefore must be created again. See Section 3.2.1.3 on page 18 for more information.

3.2.3.4 Reconciliation

A system data backup can contain misalignments between the data in the PLDB backup and the data in the DSG backup. Therefore, a system data restore requires reconciliation for all DSGs. Refer to the section on requesting reconciliation manually in *CUDB System Administrator Guide*, Reference [2] for more information.

Note: In case of group data restore and system data restore operations, reconciliation must be scheduled manually, unless the automatic execution of Selective Replica Check and Data Repair processes is disabled by configuration. Check the details of the `CudbDsGroupRepairAndResync` class in *CUDB Node Configuration Data Model Description*, Reference [8] for more information about configuring these processes.

3.2.3.5 Alarms

In case any failure occurs during the restore phase, the system sends an error message and raises the *Storage Engine, Restore Fault In DS*Reference [9] or *Storage Engine, Restore Fault In PLDB*, Reference [10] alarms.

3.2.3.6 Output

Sample outputs of the CUDB system restore process are provided below both for single source (see Section 3.2.3.6.1 on page 27) and distributed source (see Section 3.2.3.6.2 on page 29) restore processes.

3.2.3.6.1 Output of Restoring from a Single Source

```
CUDB_44 SC_2_2# cudbSystemDataBackupAndRestore -r
/home/cudb/automatedBackupStorage/

Checking /home/cudb/systemDataBackup on nodes
---------------------------------

        44: ... EMPTY
        43: ... EMPTY


VALIDATE PART
---------------------------------

NODE: 43
        DSG: 1
                NDB: 3
                NDB: 4
        DSG: 255
                NDB: 3
                NDB: 4
NODE: 44
        DSG: 0
                NDB: 3
                NDB: 4
                NDB: 5
                NDB: 6


DISTRIBUTE PART
---------------------------------

Creating /home/cudb/systemDataBackup on node 44

        Node 44 has DSG 0       -> copying ... id3-OK id4-OK id5-OK id6-OK
        Node 44 has DSG 1       -> copying ... id3-OK id4-OK
        Node 44 has DSG 255     -> copying ... id3-OK id4-OK

Creating /home/cudb/systemDataBackup on node 43

        Node 43 has DSG 0       -> copying ... id3-OK id4-OK id5-OK id6-OK
        Node 43 has DSG 1       -> copying ... id3-OK id4-OK
        Node 43 has DSG 255     -> copying ... id3-OK id4-OK


EXTRACT PART
---------------------------------

        Node 44 has DSG 0       -> extracting ... id3-OK id4-OK id5-OK id6-OK
        Node 44 has DSG 1       -> extracting ... id3-OK id4-OK
        Node 44 has DSG 255     -> extracting ... id3-OK id4-OK
```

```
        Node 43 has DSG 0       -> extracting ... id3-OK id4-OK id5-OK id6-OK
        Node 43 has DSG 1       -> extracting ... id3-OK id4-OK
        Node 43 has DSG 255     -> extracting ... id3-OK id4-OK


RESTORE PART
--------------------------------

Starting System Data Backup with command cudbDataRestore -B 2014-05-14_20-25 -L 2>&1 ...
cudbDataRestore     ver(1.3.6)

Trying for restore the backup from [2014-05-14_20-25] data files.

Restore command in PL/DS node in CUDBNode#44 sucessfully executed.
Restore command in PL/DS node in CUDBNode#43 sucessfully executed.

Restarting ldap frontends.................



Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Restarting of slapd processes is finished
Sucessfully in executing the ldap frontends restart  command  in CUDBNode#44
Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Restarting of slapd processes is finished
Sucessfully in executing the ldap frontends restart  command  in CUDBNode#43



KICKSTART PART
--------------------------------

Checking that every DSG is Master or Slave ...
Kickstarting pl on node 43 with command
        cudbManageStore -o kickstart -p ...
cudbManageStore stores to process: pl.

Launching order kickstart to pl in dsgroup 0.
Waiting for kickstart order(s) to be completed in CUDB Node 43 for stores :  pl.
Replication channel is up for store pl
kickstart order finished successfully in CUDB Node 43 for store pl.
kickstart order(s) completed in CUDB Node 43 for stores :  pl.
Stores where order kickstart was successfully completed:  pl.
cudbManageStore command successful.

Kickstarting ds unit 1 on node 43 with command
        cudbManageStore -o kickstart -d 1 ...
cudbManageStore stores to process:   ds1 (in dsgroup1).

Launching order kickstart to ds1 in dsgroup 1.
Waiting for kickstart order(s) to be completed in CUDB Node 43 for stores :  ds1.
Replication channel is up for store ds1
kickstart order finished successfully in CUDB Node 43 for store ds1.
kickstart order(s) completed in CUDB Node 43 for stores :  ds1.
Stores where order kickstart was successfully completed:  ds1.
cudbManageStore command successful.

Kickstarting ds unit 2 on node 43 with command
        cudbManageStore -o kickstart -d 2 ...
cudbManageStore stores to process:   ds2 (in dsgroup255).
```

```
Launching order kickstart to ds2 in dsgroup 255.
Waiting for kickstart order(s) to be completed in CUDB Node 43 for stores :  ds2.
Replication channel is up for store ds2
kickstart order finished successfully in CUDB Node 43 for store ds2.
kickstart order(s) completed in CUDB Node 43 for stores :  ds2.
Stores where order kickstart was successfully completed:  ds2.
cudbManageStore command successful.


ERROR SUMMARY
--------------------------------

None.
```

3.2.3.6.2 Output of Restoring from a Distributed Source

```
CUDB_44 SC_2_2# cudbSystemDataBackupAndRestore -R
2014-05-14_20-25

Checking /home/cudb/systemDataBackup on nodes
--------------------------------

        44: ... No unnecessary files found
        43: ... No unnecessary files found


VALIDATE PART
--------------------------------

NODE: 44
        DSG: 0
                NDB: 3
                NDB: 4
                NDB: 5
                NDB: 6
NODE: 43
        DSG: 1
                NDB: 3
                NDB: 4
        DSG: 255
                NDB: 3
                NDB: 4


DISTRIBUTE PART
--------------------------------

Creating /home/cudb/systemDataBackup on node 44

        Node 44 has DSG 0       -> copying ... Files already present on this node
        Node 44 has DSG 1       -> copying ... id3-OK id4-OK
        Node 44 has DSG 255     -> copying ... id3-OK id4-OK

Creating /home/cudb/systemDataBackup on node 43

        Node 43 has DSG 0       -> copying ... id3-OK id4-OK id5-OK id6-OK
        Node 43 has DSG 1       -> copying ... Files already present on this node
        Node 43 has DSG 255     -> copying ... Files already present on this node


EXTRACT PART
--------------------------------

        Node 44 has DSG 0       -> extracting ... id3-OK id4-OK id5-OK id6-OK
        Node 44 has DSG 1       -> extracting ... id3-OK id4-OK
        Node 44 has DSG 255     -> extracting ... id3-OK id4-OK
        Node 43 has DSG 0       -> extracting ... id3-OK id4-OK id5-OK id6-OK
        Node 43 has DSG 1       -> extracting ... id3-OK id4-OK
```

```
        Node 43 has DSG 255     -> extracting ... id3-OK id4-OK


RESTORE PART
--------------------------------

Starting System Data Backup with command cudbDataRestore -B 2014-05-14_20-25 -L 2>&1 ...
cudbDataRestore     ver(1.3.6)

Trying for restore the backup from [2014-05-14_20-25] data files.

Restore command in PL/DS node in CUDBNode#44 sucessfully executed.
Restore command in PL/DS node in CUDBNode#43 sucessfully executed.

Restarting ldap frontends.................



Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Restarting of slapd processes is finished
Sucessfully in executing the ldap frontends restart  command  in CUDBNode#44
Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Making sure cudbLdapFeMonitor is running (to start slapd)...OK
.Restarting of slapd processes is finished
Sucessfully in executing the ldap frontends restart  command  in CUDBNode#43


KICKSTART PART
--------------------------------

Checking that every DSG is Master or Slave ...
Kickstarting pl on node 43 with command
        cudbManageStore -o kickstart -p ...
cudbManageStore stores to process: pl.

Launching order kickstart to pl in dsgroup 0.
Waiting for kickstart order(s) to be completed in CUDB Node 43 for stores :  pl.
Replication channel is up for store pl
kickstart order finished successfully in CUDB Node 43 for store pl.
kickstart order(s) completed in CUDB Node 43 for stores :  pl.
Stores where order kickstart was successfully completed:  pl.
cudbManageStore command successful.

Kickstarting ds unit 1 on node 43 with command
        cudbManageStore -o kickstart -d 1 ...
cudbManageStore stores to process:   ds1 (in dsgroup1).

Launching order kickstart to ds1 in dsgroup 1.
Waiting for kickstart order(s) to be completed in CUDB Node 43 for stores :  ds1.
Replication channel is up for store ds1
kickstart order finished successfully in CUDB Node 43 for store ds1.
kickstart order(s) completed in CUDB Node 43 for stores :  ds1.
Stores where order kickstart was successfully completed:  ds1.
cudbManageStore command successful.

Kickstarting ds unit 2 on node 43 with command
        cudbManageStore -o kickstart -d 2 ...
cudbManageStore stores to process:   ds2 (in dsgroup255).
```

```
Launching order kickstart to ds2 in dsgroup 255.
Waiting for kickstart order(s) to be completed in CUDB Node 43 for stores :  ds2.
Replication channel is up for store ds2
kickstart order finished successfully in CUDB Node 43 for store ds2.
kickstart order(s) completed in CUDB Node 43 for stores :  ds2.
Stores where order kickstart was successfully completed:  ds2.
cudbManageStore command successful.


ERROR SUMMARY
-------------------------------

None.
```

3.3 Combined Data Backup and Restore

This section describes the combined unit data backup and restore procedures available in the CUDB system.

3.3.1 Combined Unit Data Backup and Restore

This section describes how to perform a combined unit data backup and restore in the CUDB system.

3.3.1.1 Description

The combined data backup and restore is used to restore a slave PLDB or DSG replica by using a fresh backup taken on the corresponding master replica.

3.3.1.2 Performing Combined Unit Data Backup and Restore

Perform the following steps for a combined CUDB unit data backup and restore for any slave replica of the PLDB or DSG cluster:

1. Establish a new administrative CUDB CLI session towards one of the SCs of the CUDB node where the replica to be restored is hosted:

   ssh *<admin_user>@<CUDB_Node_OAM_VIP_Address>*

2. Use the command as follows:

   - To backup and restore a PLDB replica:

     SC_2_1# **cudbUnitDataBackupAndRestore -p -n *<nodeId>***

     In the above command, ***<nodeId>*** is the identifier of the CUDB node that contains the slave replica to be restored.

Note: The command will not work if issued for PLDB in a node where any DSG master replica is present.

If the *<nodeId>* where the restore is to be performed does not have a configured PLDB, the command returns an error message.

- To backup and restore a DSG replica:

  SC_2_1# **cudbUnitDataBackupAndRestore -d *<dsgId>* -n *<nodeId>***

  In the above command, ***<dsgId>*** is the identifier of the DSG that the slave replica to be restored belongs to, while ***<nodeId>*** is the identifier of the CUDB node that contains the replica to be restored.

  Refer to *CUDB Node Commands and Parameters*, Reference [4] for more information on `cudbUnitDataBackupAndRestore`.

  `cudbUnitDataBackupAndRestore` creates a fresh backup of the data stored in the master replica of the selected PLDB or DSG. The backup is then copied to the selected slave replica, where restore is performed immediately afterwards. At the end of the backup-restore procedure, the backup files and directories are automatically removed from the master and slave replicas.

3.3.1.3 Alarms

In case any failure occurs during the backup phase, the system raises the *Storage Engine, Backup Fault In DS*, Reference [5] or *Storage Engine, Backup Fault In PLDB*, Reference [6] alarms.

3.3.1.4 Output

On one hand, the output is a new backup taken on the corresponding master replica:

```
/local/cudb/mysql/ndbd/backup/BACKUP/BACKUP-YYYY-MM-DD_HH
-mm
```

On the other hand, the output is the restoration of the slave replica in the CUDB node that contains the replica to be restored.

4 Software and Configuration Backup and Restore

CUDB supports the backup and restore of the configuration information stored in the system. The software and configuration backup (or restore) is performed separately in each CUDB node of the CUDB system.

The CUDB software and configuration backup contains the following data:

- The middleware configuration of the CUDB node.
- The configuration information of the CUDB node.
- All software packages installed on all blades or VMs of the CUDB node.
- Internal CUDB information stored in the PLDB (only for nodes with PLDB).
- Internal CUDB model configuration.

Note: Software and configuration backup/restore operations cannot be performed in the following cases:

- There are pending changes in the node configuration, meaning that the last committed changes were not applied by executing the `applyConfig` action. Backup/restore will fail with the following message:

  ```
  ERROR: there are pending configuration changes.
  Please, execute applyConfig action to apply them.
  ```

  To fix this error, execute the `applyConfig` action.

- The execution of the `applyConfig` action is ongoing. Backup/restore will fail with the following message:

  ```
  ERROR: applyConfig action is ongoing, please
  retry later.
  ```

  In this case, check the `applyConfigSatus` attribute to see the status of the current execution. After execution is finished, perform backup/restore again.

- An instance of an executed backup or restore operation is already ongoing. Backup/restore will fail with the following message:

  ```
  ERROR: Backup lock file exists.
  ```

  This error message can also be received in special fault situations, even if no other backup or restore instance was executed. In such cases, wait a few minutes, then remove the lock file with the following command:

  `rm -f /cluster/home/cudb/cudbSwBackup.lock`

4.1 Software and Configuration Backup

To order a software and configuration backup in a CUDB node, log on to the CUDB node and issue the `cudbSwBackup` command as follows:

```
SC_2_1# cudbSwBackup -c <backupName>
```

In the above command, *<backupName>* is the name of the software and configuration backup, the resulting backup is made up of the following files:

- A **tar** file (called *<backupname>.tar*) containing middleware configuration stored in `/cluster/storage/no-backup`.
- A **gzip** file (called *<backupname>.gz*) containing CUDB software and configuration stored in `/cluster/home/cudb/swbackup`.
- In case the node has a PLDB unit, the resulting backup also contains the following file:

An **sql** file (called `<backupname>-cudbSmpConfig.sql`) containing auxiliary CUDB information stored in the PLDB unit. The file is stored in `/cluster/home/cudb/swbackup`.

Refer to *CUDB Node Commands and Parameters*, Reference [4] for more information on the `cudbSwBackup` command.

The software and configuration backup process automatically rotates the oldest backup when five backups are created. In case a specific software and configuration backup must be preserved, store it on an external storage. Refer to Section 3.1.1.3 on page 5 for more information.

Note: In case any fault occurs during the backup procedure, the system raises the *Operating System, Server Configuration Backup Fault*, Reference [11] alarm.

4.2 Software and Configuration Restore

Software and configuration restore must be performed in the following cases in the CUDB system:

- Both SCs have failed, and the cluster is down.
- A previous configuration must be restored.

  Note: If the software and configuration is reverted to an earlier version, the restored configuration may not be aligned with the data stored in the system. Therefore, a data restore may also be needed after the software and configuration restore.

4.2.1 Prerequisites of Software and Configuration Restore

To perform a software and configuration restore in a CUDB node, the following conditions must be met:

- The affected cluster must be running. Use the `cudbHaState` to check the cluster state.

  The output of `cudbHaState` must contain the following information in the following sections:

  - `LOTC cluster state`: All `safNode` must be joined to the cluster.
  - `CoreMW HA state`: One instance of CoreMW must be assigned as “ACTIVE” to an SC, and another instance must be assigned as “STANDBY” to the other one.
  - `COM HA state`: One instance of COM must be assigned as “ACTIVE” to an SC, and another instance must be assigned as “STANDBY” to the other one.
  - `SI HA state`: All services must be in “ACTIVE” or “STANDBY” mode.

- In CUDB nodes with PLDB unit, this unit must be in `Alive` mode, and in **up** or **degraded** status (or in **Maintenance** mode, if it was set as such because of degradation). Use the following commands to check the PLDB and DSG status in the node:

  – Use the command as follows to check the PLDB status:

    ```
    SC_2_1# cudbManageStore --pl --order status
    ```

  – Use the command as follows to check the status of all DSGs in the node:

    ```
    SC_2_1# cudbManageStore --ds all --order status
    ```

  The command output must be similar as follows:

  ```
  Store <storage> in dsgroups <DSG group list> is alive and
  reporting status <status>.
  ```

  In the above output, *<storage>* is the ID of the PLDB or DS Unit, while *<status>* can be either `up`, `down`, or `degraded`.

- The files to restore must be placed in the CUDB node to be restored. The path is specified in Section 4.1 on page 34.

- The node ID and hardware type of the current node configuration must match with that of the backup.

In case these conditions are not met, and the restore operation returns an error, refer to the corresponding Alarm Operating Instruction (OPI) of the received error.

4.2.2 Performing Software and Configuration Restore

Follow the steps below to perform a software and configuration restore:

1. Check the name of backup to be use for restoring executing:

   ```
   SC_2_1# cudbSwBackup -l
   ```

2. Login to the CUDB node, and execute the `cudbSwBackup` command as follows:

   ```
   SC_2_1# cudbSwBackup --restore <backupName>
   ```

   In the above command, *<backupName>* is the name of the software and configuration backup containing the three files created during the software and configuration backup procedure.

   If `cudbSwBackup` is successfully finished, the following printout must appear:

```
.............

cudbSwBackup --restore::Restore successfully performed,
cudbSwBackup --restore::Cluster must be rebooted. Use command:
cudbSwBackup --restore::cluster reboot --all
```

Refer to *CUDB Node Commands and Parameters*, Reference [4] for more information on `cudbSwBackup`.

3. Restart the CUDB node with the following command:

   ```
   SC_2_1# cluster reboot --all
   ```

   Note: Connection is lost during cluster reboot. Connection must be reestablished automatically as soon as the CUDB node restart is finished.

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

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

5. Check the node status as described in Section 4.2.1 on page 35.

6. It can occur that the restored backup was created while configuration changes were pending. To check this, use the following command to see if the `cudbOiImmChanges.txt` file exists on the node:

   ```
   SC_2_1# less /cluster/home/cudb/oam/configMgmt/commands
   /config/cudbOiImmChanges.txt
   ```

   If the file exists, the changes can be applied by executing the `applyConfig` administrative operation.

   Refer to the Object Model Modification Procedure in *CUDB Node Configuration Data Model Description*, Reference [8] for more information on all the steps required to modify the object model.

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

   ```
   SC_2_1# exit
   ```

Warning!

During software and configuration restore, the affected CUDB node is unavailable. The CUDB nodes becomes available only after the cluster is rebooted.

4.3 Periodic Software and Configuration Backup in a CUDB System

The CUDB system supports scheduled periodic software and configuration backups.

4.3.1 Configuring Periodic Software and Configuration Backup

Perform the following steps to configure periodic CUDB software and configuration backups.

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. Login to one of the SCs with the following command:

   ssh OAM*<X>*

   <X> stands for the SC number, which is 1 or 2.

3. Run the following command to schedule a backup:

   SC_2_*<X>*# **cudbSwBackup --schedule *<CRON_EXPRESSION>***

   The format of the *<CRON_EXPRESSION>* follows the standard UNIX cron expression format. For example, if the *<CRON_EXPRESSION>* is set to **'5 0 * * *'**, then the backup is launched at 0:05 every day.

   The above command creates a patch for the system-wide `crontab` file, so the configured schedule is not lost after blade or VM reboot.

   The file name of the resulting software and configuration backup has the following format:

   cudb_sw_backup*<YYYY><MM><DD><hh><mm>*

   In the above file name, *<YYYY><MM><DD><hh><mm>* stand for the year, month, day, hour and minute of creating the backup, respectively.

4. Login to the other SC as described in Step 2, and configure the scheduled backup as described in Step 3. Configuring the periodic data backup on both SCs is necessary to make sure that the backup is performed even if one of them is unavailable.

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

   exit

Note: Scheduled software and configuration backups are performed on a per-CUDB node basis, which means that this procedure must be performed separately on every CUDB node in the system where periodic software and configuration backups are required.

Also, use the `crontab` command to disable the periodic software and configuration backup task.

4.3.2 Printing Configured Software and Configuration Backups

Perform the following steps to print the scheduled software and configuration backups configured in the system:

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. Login to one of the SCs with the following command:

   ssh OAM*<X>*

   <X> stands for the SC number, which is 1 or 2.

3. Run the following command to check the scheduled backups:

   SC_2_*<X>*# **cudbSwBackup --print**

4.4 BSP 8100 Configuration Backup and Restore

In case the CUDB system is deployed on native BSP 8100, refer to the ‘‘Backup and Restore BSP Configuration’’ document in the BSP 8100 CPI for a detailed description on how to create, export, and delete a backup from a BSP 8100 node, and also how to import and restore configuration and software from a saved backup.

Glossary

For the terms, definitions, acronyms and abbreviations used in this document, refer to *CUDB Glossary of Terms and Acronyms*, Reference [12].

Reference List

CUDB Documents

[1] *CUDB Data Storage Handling*

[2] *CUDB System Administrator Guide*

[3] *CUDB High Availability*

[4] *CUDB Node Commands and Parameters*

[5] *Storage Engine, Backup Fault In DS*

[6] *Storage Engine, Backup Fault In PLDB*

[7] *Storage Engine, Backup Notification Failure To Provisioning Gateway*

[8] *CUDB Node Configuration Data Model Description*

[9] *Storage Engine, Restore Fault In DS*

[10] *Storage Engine, Restore Fault In PLDB*

[11] *Operating System, Server Configuration Backup Fault*

[12] *CUDB Glossary of Terms and Acronyms*