

# CUDB Automatic Handling of Network Isolation Output Description

---

## INTERWORK DESCRIPTION

**Copyright**

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

**Disclaimer**

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

**Trademark List**

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



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Scope	1
1.2	Revision Information	1
1.3	Target Groups	1
1.4	Typographic Conventions	2
<b>2</b>	<b>Overview</b>	<b>3</b>
<b>3</b>	<b>Log File Description</b>	<b>5</b>
3.1	Common Terms in Output Logs	5
3.2	Data Repair Output Log Files	6
	<b>Glossary</b>	<b>19</b>
	<b>Reference List</b>	<b>21</b>





# 1 Introduction

The Automatic Handling of Network Isolation function is an automatic process within the Ericsson Centralized User Database (CUDB). It attempts to handle and repair data loss that possibly might have happened due to network split or unexpected Processing Layer Database (PLDB) or Data Store Unit Group (DSG) mastership change.

**Note:** Unexpected mastership change means that the former master replica rejoins the system as a new slave replica, but it is not able to establish replication with the current master.

The Automatic Handling of Network Isolation process has the following two steps:

- Selective Replica Check
- Data Repair

This document provides a detailed description of the logs produced by Data Repair.

## 1.1 Scope

This document describes the structure of the output logs produced by Automatic Handling of Network Isolation function. The description of any application Front End (FE) specific data or internal CUDB data stored in Lightweight Directory Access Protocol (LDAP) entries is out of the scope of this document.

## 1.2 Revision Information

**Rev. A** This document is based on 1/155 19-CRH 109 0575/9 with the following changes:

- Terminology updates throughout the document because of virtualized deployment support.

## 1.3 Target Groups

This document is intended for users working with the Automatic Handling of Network Isolation output logs. This document assumes the general knowledge of the LDAP standard. Basic understanding of the CUDB node architecture and the Linux file system is assumed.



## 1.4 Typographic Conventions

Typographic conventions can be found in the following document:

- *Typographic Conventions*



## 2 Overview

Selective Replica Check is executed on a database cluster which used to be a master replica but just became a slave replica. This database cluster is not able to synchronize with the current master replica. The output of Selective Replica Check is a list of the LDAP entries that were changed in this database cluster when it was a master replica, but was possibly not replicated into the current master replica, along with their contents.

For more information about Selective Replica Check interactions, refer to *CUDB Data Storage Handling*, Reference [1].

Data Repair contrasts the LDAP entries in the output of Selective Replica Check with data in the current master replica. Then, on an entry-by-entry basis, Data Repair decides whether to keep the data in the current master replica as it is or to modify or delete the data in the current master replica based on the data from the former master replica. Each entry in the input is logged to the Data Repair output logs, either to the repaired entries log or the unrepaired entries log.

For more information about Data Repair interactions, refer to *CUDB Data Storage Handling*, Reference [1].

This document provides detailed description of the Data Repair logs.

The log files can be located based on the Automatic Handling of Network Isolation task ID that is sent in the active description of all alarms related to Automatic Handling of Network Isolation.

For an overview of the alarm relationships of the Automatic Handling of Network Isolation process, refer to *CUDB Node Fault Management Configuration Guide*, Reference [2].







## 3 Log File Description

This section describes the output log files for Data Repair.

---

---

### Warning!

Log files are compressed. For log file decompression, avoid `less` and `zless` commands, because those use space in RAM, which is very limited, and the log files have a high compression ratio.

It is advised to copy the log files from the traffic handling payload blades or Virtual Machines (VMs) to another machine for extensive processing and analysis.

---

---

### 3.1 Common Terms in Output Logs

In this specification, the LDAP entry attribute value dump means the list of LDAP attribute names and their value(s), in LDIF format as produced by the `ldapsearch` utility. That is:

- A logical line may be formed from several physical lines as follows: Each line beginning with a single white space character is concatenated to the previous line without the leading space. This is valid for comment lines as well.
- Each {attribute name, value} pair is in a separate logical line.
- For multi-valued attributes, the attribute name is repeated in subsequent logical lines containing the same attribute name and the different values.
- Attribute value encoding:
  - If the attribute name is followed by a single colon (':'), the attribute value is represented in a 7 bit ASCII encoding (printable value).
  - Otherwise, the attribute value is followed by a double colon ( '::'), which means Base64 encoding for the attribute value.

The `binlogTimestamp` contains the time when the entry was last modified on the former master according to the `binlog` during the analyzed period. Each `binlogTimestamp` value is later than or equal to the `incidentTimestamp`.



## 3.2 Data Repair Output Log Files

This section provides the location, naming, and contents of the Data Repair output logs.

### 3.2.1 Location and Naming of the Data Repair Output Logs

The Data Repair output logs are located on the payload blade or VM where the Data Repair process is executed (identified in the active description of the Data Repair related alarms), in the directory `/local2/cudb/ahsi/replica_repair`. The output log files produced by Data Repair follow the LDIF format and are `gzip` compressed. However, they must not be used as LDIF file input to LDAP utilities in the CUDB system.

For each repair task, two kinds of output log files are written:

- typically one, or in case of error recovery via re-execution, several repaired log file(s), containing the information about LDAP entries that have been successfully repaired, with filename `datarepair_<task_id>_repaired_<unix_timestamp>.ldif.gz`
- typically one, or in case of error recovery via re-execution, several unrepaired log file(s), containing the information about LDAP entries that have not been repaired, with filename `datarepair_<task_id>_unrepaired_<unix_timestamp>.ldif.gz`

where `<task_id>` is structured as:

`ahsi.UTC_<YYYY-MM-DD-hh-mm-ss>_S<site>_N<node>_D<dsg>`

`<YYYY-MM-DD-hh-mm-ss>` is the UTC time stamp containing the time of the network incident that triggered the Automatic Handling of Network Isolation process. The `S`, `N`, and `D` parameters are the CUDB site, node and DSG identifiers respectively, DSG 0 being the PLDB.

`<unix_timestamp>` is the Unix epoch value (in seconds) as an integer, referring to the start of execution of the Data Repair task.

**Note:**

- During normal execution of Data Repair, one repaired and/or unrepaired log file is produced for a task ID.
- In case of multiple execution (error recovery via re-execution) of Data Repair with the same input for the same task ID, several output logs can be written. In this case, the `<task_id>` is the same in the log files, but multiple `<unix_timestamp>` values are seen. Listing the files in ascending order based on the `<unix_timestamp>` gives the chronological order of files, and thereby the order of repair of the LDAP entries listed in them.



In Example 1, one repaired and one unrepaired log file belongs to the task ID `ahsi.UTC_2015-08-31-20-04-14_S2_N116_D1`.

```
CUDB_115 PL_2_3# cd /local2/cudb/ahsi/replica_repair
CUDB_115 PL_2_3# ll
total 56
-rw-r----- 1 root cudbadmin 21662 Aug 31 20:04
datarepair_ahsi.UTC_2015-08-31-20-04-14_S2_N116_D1_repaired_1441044257.ldif.gz
-rw-r----- 1 root cudbadmin 377 Aug 31 20:04
datarepair_ahsi.UTC_2015-08-31-20-04-14_S2_N116_D1_unrepaired_1441044257.ldif.gz
```

### Example 1 Log Files, Normal Execution

In Example 2, two repaired and two unrepaired log files belong to the task ID `ahsi.UTC_2015-08-31-20-04-14_S2_N116_D1`.

```
CUDB_115 PL_2_3# cd /local2/cudb/ahsi/replica_repair
CUDB_115 PL_2_3# ll
total 56
-rw-r----- 1 root cudbadmin 21662 Aug 31 20:04
datarepair_ahsi.UTC_2015-08-31-20-04-14_S2_N116_D1_repaired_1441044257.ldif.gz
-rw-r----- 1 root cudbadmin 21662 Aug 31 20:04
datarepair_ahsi.UTC_2015-08-31-20-04-14_S2_N116_D1_repaired_1441044281.ldif.gz
-rw-r----- 1 root cudbadmin 377 Aug 31 20:04
datarepair_ahsi.UTC_2015-08-31-20-04-14_S2_N116_D1_unrepaired_1441044257.ldif.gz
-rw-r----- 1 root cudbadmin 377 Aug 31 20:04
datarepair_ahsi.UTC_2015-08-31-20-04-14_S2_N116_D1_unrepaired_1441044281.ldif.gz
```

### Example 2 Log Files, Re-Execution Due to Error Recovery

#### Note:

- The epoch values 1441044257 and 1441044281 seen in the filenames above correspond to time stamps Mon Aug 31 20:04:17 CEST 2015 and Mon Aug 31 20:04:41 CEST 2015, respectively. See also the file modification dates.
- The log files have ownership of `root`, group `cudbadmin`, and readable only for the owner and group, as shown in the examples above.
- The highest `<unix_timestamp>` value belonging to a repair task is sent in the active alarm description of the Storage Engine, Data Inconsistency between Replicas Repaired and Storage Engine, Unrepaired Data Inconsistency between Replicas alarms.
- The `<task_id>` is sent in the active description of the aforementioned alarms.

The mentioned alarms above are sent when the repair task and the output is completed.

## 3.2.2 Contents of the Data Repair Output Logs

Figure 1 shows the sequence of records in the output file.

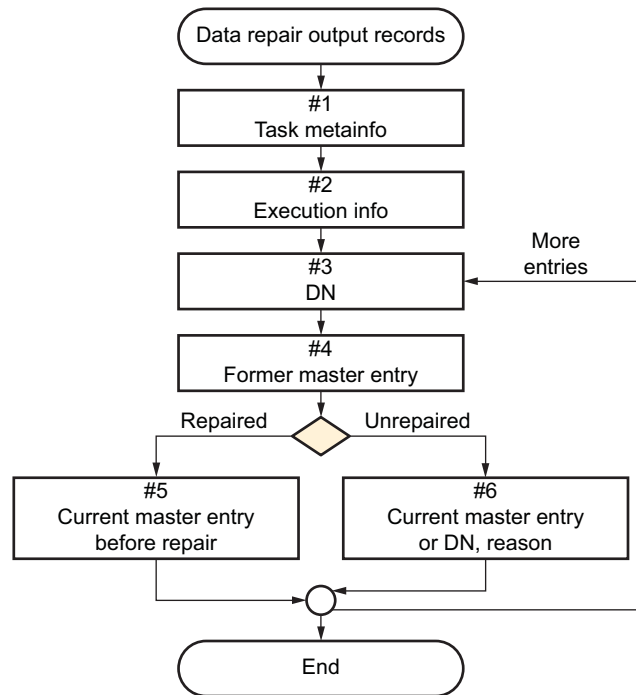


Figure 1 Record Sequence in the Output File

### 3.2.2.1 Record #1: Task Metainfo

This record contains basic information about the time and place of execution, for both Data Repair logs.

```
## CUDB Data Repair: {Repaired|Unrepaired} entries
## Execution time: UTC_<YYYY-MM-DD-hh-mm-ss>
## Execution node: N<node>,<payload blade or VM>
## Input: <task_id>
## Timestamp: <unix_timestamp>
```

The `<unix_timestamp>` is the Unix epoch value of the repair execution, represented as decimal integer.

### 3.2.2.2 Record #2: Execution Info

This record specifies that Data Repair was executed after Selective Replica Check, and states the time of the network incident that triggered the Automatic Handling of Network Isolation process.

```
## Execution: Selective Check
## Former master: S<site>_N<node>_D<dsg>
## incidentTimestamp: <YYYYMMDDhhmmss>Z
```



##

## <Header lines as LDIF comments from the Selective Check output file.>

The `incidentTimestamp` is the starting time of binlog analysis by Selective Replica Check.

### 3.2.2.3

#### Record #3: DN

This record contains a sequence number given to the input records, in the order they are found in the input file.

## <sequence number> DN: <dn>

### 3.2.2.4

#### Record #4: Former Master Entry

This is the content of the LDAP entry that was queried from the former master by Selective Replica Check when the former master still contained the unreplicated data changes.

## Former master

## <operation line>

<LDAP entry attribute value dump>

<empty line>

The <operation line> can be one of the following:

- **Modify:**

## Operation: modify binlogTimestamp: <UTC\_timestamp>

This means that this LDAP entry was modified or added in the former master replica.

- **Delete:**

## Operation: delete binlogTimestamp: <UTC\_timestamp>

This means that the LDAP entry was deleted from the former master replica. Only the DN of the entry is listed as an attribute.

- **Error:**

The following error records may appear:

## Operation: error\_missing\_insert binlogTimestamp: <UTC\_timestamp> <error text>

Selective Replica Check has found data on the former master inconsistent with the operational logs, but Data Repair attempted to repair such entries like in the case of **Modify** operation.



```
## Operation: error_missing_delete binlogTimestamp: <UTC_timestamp> <error text>
```

Selective Replica Check has found data on the former master inconsistent with the operational logs, but Data Repair attempted to repair such entries like in the case of **Delete** operation. Only the DN of the entry is listed as an attribute.

```
## Operation: error_entry_unattainable binlogTimestamp: <UTC_timestamp> binlogOperation:
<operation> <error text>
```

The entry could not be fetched from the former master. *<operation>* is the last LDAP operation performed on the entry, can be **Modify** or **Delete**. Only the DN of the entry is listed as an attribute.

### 3.2.2.5 Record #5: Current Master Entry Before Repair (Repaired log)

This is the content of the `repaired` LDAP entry, queried from the current master before the repair attempt.

```
## Current master
```

```
## Repair status: {repaired|already_identical|already_deleted}
```

```
<LDAP entry attribute value dump>
```

```
<empty line>
```

The meaning of Repair status:

- `repaired`: repaired by Data Repair.
- `already_identical`: the content of the entry is identical with the former master data.
- `already_deleted`: a delete operation was attempted, but the entry was already deleted on the current master before the repair attempt. In this case only the DN is listed as an attribute.

### 3.2.2.6 Record #6: Current Master Entry Before Repair (Unrepaired log)

This is the content of the `unrepaired` LDAP entry, queried from the current master before the repair attempt.

```
## Current master
```

```
## Unrepair reason: {timestamp|no_timestamp|error_record|skip_insert|traffic_conflict|repair_cancelled|ld
ap_error <error description>}
```

```
<LDAP entry attribute dump>
```

```
<empty line>
```

The meaning of Unrepair reason:

- `timestamp`: the time stamp of the entry on the current master is later than the incident timestamp.



- `no_timestamp`: the entry has no timestamp.
- `error_record`: the entry could not be fetched from the former master.
- `skip_insert`: the entry does not exist on the current master, and inserts are not performed by Data Repair, therefore the entry is logged as unrepaired.
- `traffic_conflict`: concurrent write access detected between repair and traffic through the CDC mechanism.
- `repair_cancelled`: the repair process was interrupted. If this reason appears, all subsequent entries are recorded into the `unrepaired` log with this reason.
- `ldap_error <error description>`: LDAP access error happened during repair,

where *<error description>* ::= '[' *<LDAP error code>* ']' '[' *<error message>* ']' '[' *<diagnostic message>* ']'.

**Note:** `<diagnostic message>` may be empty.

### 3.2.3 Example Data Repair Log Files

The two example files below demonstrate how the corresponding repaired and unrepaired logs, respectively, are written.

**Note:** Comments for understanding the examples are enclosed between single # characters in the unrepaired log, but will not be present in the output log.

The sample files are shown in uncompressed form.

Example 3 shows the repaired log.

[illegible]



1/155 19-CRH 109 0575/10-V1 Uen A | 2016-11-29



### Example 3 Repaired Log

```
## CUDB Data Repair: Unrepaired entries
## Execution time: UTC 2015-04-16-14-59-10
## Execution node: N115,PL_2_3
## Input: ahsl UTC 2015-04-16-14-57-16_S1_N115_D1
## Timestamp: 1429189150
## Execution: Selective Check
## Former master: S1_N115_D1
## incidentTimestamp: 20150416145701Z
##
## File: CUDB115 PL 2 3:/local/cudb_ddci/replica_repair/ahsl UTC 2015-04-16-14-57-16_S1_N115_D1.ldif.gz
## Site:1,Node:115,DSG:1
## Entries impacted: 42
## Binlog files used: 3
## PL_2_3: /local/cudb/mysql/mysqldMaster/binlogs/
## -rw-rw---- 1 mysql mysql      189 Apr 27 21:58 log-bin.000002
## -rw-rw---- 1 mysql mysql    2368707 Apr 29 00:07 log-bin.000003
## PL_2_4: /local/cudb/mysql/mysqldMaster/binlogs/
## -rw-rw---- 1 mysql mysql    1264601 Apr 29 00:29 log-bin.000027

# Entry on new master too new #
## 3 DN: serv=auth,mscId=100000000000000000000000001482822216,ou=multiSCs,dc=operator,dc=com
## Former master
## Operation: modify binlogTimestamp: 20150416145704Z
dn: serv=auth,mscId=100000000000000000000000001482822216,ou=multiSCs,dc=operator
,dc=com
objectClass: CUDBService
```

[illegible]

15



1/155 19-CRH 109 0575/10-V1 Uen A | 2016-11-29



IMSI: 214020010025304  
NAM: 0  
CDC: 3  
CSP: 51  
SUBSCSPVERS: 7  
PDPCP: 61  
SUBSPDPCPVERS: 3  
RSA: 1  
SUBSRSAVERS: 0  
serv: CSPS  
CSLOC: 2  
PSLOC: 2  
SGSNNUM: 1949101923101  
GSMUEFEAT: 0  
OBO: 3  
ARD: 1  
PRET: 1  
STYPE: 1  
CAT: 5  
DBSG: 1  
OFA: 0  
SOCB: 1  
PWD: 1111  
PWDC: 0  
SOCFB: 1  
SOCFNRC: 1  
SOCFNRY: 1  
SOCFU: 1  
SODCF: 1  
SOSDCF: 7  
SOCLIP: 1  
SOCLIR: 1  
SOCOLP: 1  
TS11: 1  
TS21: 1  
TS22: 1  
AOC: 1  
CAW: 1  
HOLD: 1  
MPTY: 1  
TIN: 1  
BAIC: 1  
BAOC: 1  
BICRO: 1  
BOIC: 1  
BOIEXH: 1  
CFB: 1  
CFNRC: 1  
CFNRY: 1  
CFU: 1  
DCF: 1  
SPN: 1  
CLIP: 1  
CLIR: 1  
COLP: 1  
COLR: 1  
BSGDCFREG: 1  
BSGDCFACOTOP: 1  
CAWTS10ST: 8  
CFBTS10ST: 8  
CFUTS10ST: 8  
CFNRCTS10ST: 8  
CFNRYTS10ST: 8  
DCFTS10ST: 6  
DCFTS10FNUM: kZSZKQAA8A==  
DCFTS10TIME: 10  
DCFTS10CCREL: 0  
DCFTS10ZCREL: 0  
SPNTS10ST: 6  
SPNTS10FNUM: kZSZKQAA8A==  
SPNTS10CCREL: 0  
SPNTS10ZCREL: 0  
BAICTS10ST: 8  
BAOCTS10ST: 8  
BICROTS10ST: 8  
BOICTS10ST: 8



### Example 4 Unrepaired Log



## 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 Data Storage Handling*
- [2] *CUDB Node Fault Management Configuration Guide*
- [3] *CUDB Glossary of Terms and Acronyms*