

CUDB Import and Export Procedures

USER GUIDE

Copyright

© Ericsson AB 2016, 2017. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

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

Trademark List

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



Contents

1	Introduction	1
1.1	Scope	1
1.2	Revision Information	1
1.3	Typographic Conventions	1
2	Overview	3
3	Import Procedure	5
3.1	Preconditions	5
3.2	Input	5
3.2.1	PLDB Entries Excluding DEs	6
3.2.2	PLDB DEs	7
3.2.3	DS Entries	8
3.3	Execution	8
3.3.1	Importing PLDB Entries	10
3.3.2	Importing DS Entries	11
3.3.3	Configuration File Directives	11
3.3.4	Limitations and Recommendations	12
3.4	Import Verification	12
3.4.1	Creating the LDIF File Used for Search	13
3.4.2	Retrieving Data	13
3.4.3	File Comparison	15
4	Export Procedure	17
4.1	Preconditions	17
4.2	Execution	17
4.2.1	Configuration File Directives	18
4.2.2	LDAP Filters	21
4.2.3	Export Examples	22
4.2.4	Limitations and Recommendations	24
4.3	Complex Export Queries Using Filters and Scripts	24
	Glossary	27
	Reference List	29





1 Introduction

This document describes the import and export procedures used to load and extract data in the Ericsson Centralized User Database (CUDB).

1.1 Scope

This document provides a guide to the import and export procedures in CUDB. The document assumes basic knowledge of the CUDB system and Lightweight Directory Access Protocol (LDAP).

1.2 Revision Information

- | | |
|---------------|--|
| Rev. A | This document is based on 6/1553-HDA 104 03/9 with the following changes: <ul style="list-style-type: none">• Terminology updates throughout the document because of Virtualized Network Function (VNF) support. |
| Rev. B | Other than editorial changes, this document has been revised as follows: <ul style="list-style-type: none">• Terminology updated throughout the document. |
| Rev. C | Other than editorial changes, this document has been revised as follows: <ul style="list-style-type: none">• Section 4.2 on page 17: Updated -1 <output-file> with security related information. |

1.3 Typographic Conventions

Typographic conventions can be found in the following document:

- *Typographic Conventions*





2 Overview

The import procedure is used to load massive amounts of data in a more efficient and faster way than by using the standard LDAP interface. See Section 3 on page 5 for more information.

The export procedure is used to extract massive amounts of data from CUDB also in a more efficient and faster way than by using the standard LDAP interface. See Section 4 on page 17 for more information.





3 Import Procedure

The import procedure is used to load large amount of data into CUDB, for example when migrating data from an old system into CUDB. It offers a more efficient and faster method to load data than the standard LDAP interface by storing entries in groups (minimizing the number of required database accesses) and always targeting local database replicas, therefore avoiding proxy operations between CUDB nodes.

The procedure requires that the data to import is provided in LDAP Data Interchange Format (LDIF) files, partitioned as explained in Section 3.2 on page 5.

The import procedure is based on the `slapadd` command.

This section describes the following aspects of the input procedure:

- The preconditions to fulfill before the import process is executed.
- The input required by the process.
- Executing the import process.
- The optional verification procedure for the import procedure.

3.1 Preconditions

Before starting the import procedure, the following conditions must be fulfilled:

- The CUDB nodes comprising the CUDB system must be up and running.
- The CUDB reallocation process must not be running. For more information on this process, refer to *CUDB Subscription Reallocation*, Reference [1].
- Optionally, if the verification processes must be executed after the import, an LDAP user with `readModeInPL=LP` must be created. For more information on the `readModeInPL` user attribute and on creating CUDB users, refer to *CUDB System Administrator Guide*, Reference [2].

3.2 Input

The import procedure requires that the data to be imported is provided in a set of LDIF files partitioned as follows:

- Processing Layer Database (PLDB) entries, excluding Distribution Entries (DEs)



- DEs
- Data Store (DS) entries

For the definitions of the PLDB entries, DEs and DS entries, refer to *CUDB Glossary of Terms and Acronyms*, Reference [3].

Note: After the input, an optional verification procedure is also available, see Section 3.4 on page 12 for more information. To ease this procedure, the object classes and attributes of the entries must follow the order described in Section 3.4.2 on page 13.

Note: The maximum length supported for a line in an LDIF file is 4096 characters. If an attribute contains a value of a length such that this maximum is surpassed, the line must be split into several lines so that none of them goes beyond this limit. In order to split a line into several lines, the continuation lines must be preceded exactly by one space character at the beginning of each continued line. For more details on splitting lines in LDIF files, see *RFC 2849 The LDAP Data Interchange Format (LDIF) - Technical Specification*, Reference [5].

3.2.1 PLDB Entries Excluding DEs

One file containing the PLDB entries excluding DEs must be provided. The entries belonging to the default CUDB DIT must not be included in this file.

This file can be split to several files to allow the launching of several import processes in parallel to speed up the loading of the data.

Note: The ancestors of an entry cannot be in a different LDIF file to the one where the entry is.



```
dn:MSISDN=9100000000000000,dc=MSISDN,ou=identities,
  dc=operator,dc=com
objectclass:alias
objectclass:MSISDN
msisdn:9100000000000000
aliasedObjectName:mscId=98765,ou=multiSCs,
  dc=operator,dc=com
```

```
dn:serv=Auth,ou=mscCommonData,
  dc=operator,dc=com
objectclass:top
objectclass:CommonServiceData
serv:Auth
```

```
dn:ou=commonProfile,serv=Auth,ou=mscCommonData,
  dc=operator,dc=com
objectclass:top
objectclass:organizationalUnit
objectClass:AuthIMSIData
ou:commonProfile
other_attr:other_value
```

Example 1 LDIF File Contents with PLDB Entries Excluding DEs

3.2.2

PLDB DEs

One file containing the DEs must be provided. The DEs in this file must include the `DSUnitGroup` attribute indicating on which DS Unit Group (DSG) the entries below each DE must be stored.

This file may be split into several files to allow the launching of several import processes in parallel to speed up the loading of the data.

Note: The ancestors of an entry cannot be in a different LDIF file to the one where the entry is.

```
dn:mscId=98765,ou=multiSCs,dc=operator,dc=com
objectclass:CUDBMultiServiceConsumer
mscId: 98765
DSUnitGroup: 1
```

```
dn:assocId=12398,ou=associations,dc=operator,dc=com
objectclass:CUDBAssociation
assocId: 12398
DSUnitGroup: 2
```

Example 2 LDIF File Contents with DEs



3.2.3 DS Entries

For DS entries, one file must be provided for each DSG where data is to be imported.

Each of these files may be split into several files to allow the launching of several import processes in parallel to speed up the loading of the data.

Note: The ancestors of an entry cannot be in a different LDIF file to the one where the entry is.

```
dn:serv=CSPS,mscId=98765,ou=multiSCs,  
  dc=operator,dc=com  
objectclass:CUDBService  
serv:CSPS  
objectclass:CsPsMSISDNSubscriber  
MSISDN:910000000000000  
IMSI:123456700000000
```

```
dn:ei=gprs,serv=CSPS,mscId=98765,ou=multiSCs,  
  dc=operator,dc=com  
objectClass:alias  
objectclass:CUDBExtensibleObject  
ei:gprs  
aliasedObjectName: PDPCP=1,ou=gprsProfiles,serv=CSPS,  
  ou=mscCommonData,dc=operator,dc=com
```

Example 3 DS Entries

3.3 Execution

The import process is executed by running the `slapadd` command from a System Controller (SC), located in the following location:

```
/opt/ericsson/cudb/OAM/bin/slapadd
```

The available command options are as follows:

- `-l`: Used when passing an LDIF file as input parameter.
- `-f`: Used when passing the default configuration file.
- `--dsg`: Used to import DS entries.
- `--pldb`: Used to import any kind of PLDB entries.

The below examples show the use of the `--dsg` and `--pldb` options of the `slapadd` command mentioned above:



```
/opt/ericsson/cudb/OAM/bin/slapadd -f /cluster/home/cudb/
dataAccess/ldapAccess/import-export/config/slaptool.conf
-l <PLDB_non-DE_importFile>.ldif --pldb
```

```
/opt/ericsson/cudb/OAM/bin/slapadd -f /cluster/home/cudb/d
ataAccess/ldapAccess/import-export/config/slaptool.conf -l
<PLDB_DE_importFile>.ldif --pldb
```

```
/opt/ericsson/cudb/OAM/bin/slapadd -f /cluster/home/cudb/d
ataAccess/ldapAccess/import-export/config/slaptool.conf -l
<DS_importFile>.ldif --dsg 1
```

To speed up import operations, the import process packs the group of entries to import into groups called “batches”. The entries included in a “batch” are written atomically in the PLDB or the DS databases: therefore, any problem on any of these entries causes the complete batch to fail (that is, none of the entries in the batch will be imported).

If an error occurs when executing `slapadd`, the process stops and the import tool prints the following:

- The DN's of the failed entries included in the failed batch.
- The error found during the batch execution.

In case the exact entry causing the failure must be found, perform the following steps:

1. Reduce the batch size to 1 by using the `import-batch-size` parameter. See Section 3.3.3 on page 11 for more details.
2. Create an auxiliary LDIF file with the entries included in the failed batch (they are printed in the `slapadd` error output).

Importing that auxiliary LDIF file with `import-batch-size` set to 1 results in the identification of the first failed entry.

Note: Do not forget to remove the `import-batch-size` directive after the error has been found, as performance in subsequent imports would be otherwise degraded.

The import tool also provides the `-c` command line option to force the continuous processing of the entries. With this option, any error during importing is reported but the tool continues importing new entries in the LDIF file. If this option is not specified, the command exits when it finds the first failed entry. Failed entries can be imported again if the failure occurs because of a temporary error.

Note: The `-c` option also forces the import process to reduce the batch size to 1, resulting in serious import performance degradation.

PLDB entries must be imported before DS entries.



DS entries to be imported in different nodes may be imported in parallel without expecting any interactions between the import processes.

3.3.1 Importing PLDB Entries

The procedure to import the PLDB entries consists of the following steps:

1. Find the CUDB node hosting the master replica of the PLDB and login to one of its SCs. For information on how to locate the CUDB node, refer to *CUDB System Administrator Guide*, Reference [2].
2. Copy the LDIF files containing the PLDB entries (including the files containing the DEs) to be imported to the SC. Make sure there is enough free space before copying the files.

3. Import the PLDB entries excluding the DEs by issuing the `/opt/ericsson/cudb/OAM/bin/slapadd` command and passing the file containing the PLDB entries excluding the DEs. For example:

```
# /opt/ericsson/cudb/OAM/bin/slapadd -f /cluster/home/cudb/dataAccess/ldapAccess/import-export/config/slaptopool.conf -l <path_to_ldif_file>/PLDBEntries_PL_0.ldif --pldb
```

Note: Several commands can be launched in parallel or in sequence, if the entries are split to several files.

See Section 3.3.4 on page 12 for the guidelines on the limits of import command parallelization.

4. Optionally, check that the entries have been replicated to other CUDB nodes:

Using an LDAP user with `readModeInPL=LP` an LDAP search operation looking for the last entry in the LDIF file can be sent to every CUDB node hosting a slave replica of the PLDB to check that the entry in the imported file is stored in the local PLDB.

5. Import the PLDB DEs by issuing the `/opt/ericsson/cudb/OAM/bin/slapadd` command and passing the file containing the DEs. For example:

```
# /opt/ericsson/cudb/OAM/bin/slapadd -f /cluster/home/cudb/dataAccess/ldapAccess/import-export/config/slaptopool.conf -l <path_to_ldif_file>/Distribution_PL_0.ldif --pldb
```

Note: Several commands can be launched in parallel or in sequence, if the entries are split to several files.

See Section 3.3.4 on page 12 for the guidelines on the limits of import command parallelization.

6. Optionally, check that the entries have been replicated to other CUDB nodes using the same procedure as in Step 4.



3.3.2 Importing DS Entries

The procedure to import the DS entries consists of following the next steps for each DSG where data is to be imported:

1. Find the CUDb node hosting the master replica for the DSG where entries are to be imported, and login to one of its SCs. For information on how to locate these nodes, refer to *CUDb System Administrator Guide*, Reference [2].
2. Copy the LDIF file containing the DS entries to be imported into the selected DSG to the SC. Make sure there is enough free space before copying the file.
3. Import the DS entries by issuing the `/opt/ericsson/cudb/OAM/bin/slapadd` command and passing the corresponding DS entries file. For example:

```
# /opt/ericsson/cudb/OAM/bin/slapadd -f /cluster/home/cudb/dataAccess/ldapAccess/import-export/config/slaptool.conf -l <path_to_ldif_file>/DSG_1.ldif --dsg 1
```

Note: Several commands can be launched in parallel or in sequence, if the entries are split to several files.

See Section 3.3.4 on page 12 for the guidelines about the limits of import command parallelization.

3.3.3 Configuration File Directives

This section describes the configuration file directives that are specific to the import process.

The configuration file where the `import-batch-size` parameter can be modified is located in the below location:

```
/home/cudb/dataAccess/ldapAccess/import-export/config/slaptool-config/slaptool-options.conf
```

3.3.3.1 import-batch-size Directive (Optional Directive)

The `import-batch-size` directive is used to configure the size of the group of entries that are sent in each data store access.

This is an optional directive that must not be changed, except for troubleshooting purposes.

Higher values provide better import performance, but stability issues may appear depending on traffic load. Therefore, it is not recommended to change the default value.

*Table 1 Permitted Values of the import-batch-size Directive*

import-batch-size Directive	Description
value	Size of the group of entries that are stored in each data store access. ⁽¹⁾ Minimum value: 1 Maximum value: 500 Default value: 200

(1) The import-batch-size directive is ignored whenever the continuous mode (-c flag) is used.

3.3.4 Limitations and Recommendations

Consider the following limitations and recommendations before performing import operations:

- Not conflicting or shared identities exist in the CUDB system (for example, EPS data exists for the HLR/AUC subscriber).
- Import time can be reduced in CUDB nodes by running several `slapadd` commands simultaneously, evenly distributed between the two SCs.
- In case the CUDB system hosts the standard Ericsson HLR Profile or the standard Ericsson IMS Profile, it is not recommended to run more than 8 commands at the same time on each SC. This limit may vary for systems hosting other profiles.
- Imports are data-intensive operations, resulting in extra load on the CUDB system. If the extra load causes problems, the import operation must be executed at low traffic hours.
- As a general guideline, it is safe to import data in different DSGs at the same time. Using several import commands for achieving faster data import in the PLDB or the specific DS is possible, but it is not recommended to use more than 4 commands at the same time. This limit may vary depending on the traffic conditions in the node, or on the traffic profiles used in the node.
- For CUDB systems deployed on native BSP 8100 containing both GEP3 and GEP5 nodes, import must be performed only when the PLDB master replica is hosted on a GEP3 node in order to prevent over-provisioning of GEP3.

3.4 Import Verification

After data is loaded into CUDB, an optional verification procedure can be executed to check that the LDAP entries have been successfully imported in CUDB.



The procedure consists of extracting the just imported data into LDIF files and comparing these with the LDIF files used to import the data. To extract the data, a set of search LDIF files must be created.

The verification is done in the following steps:

1. Creating the LDIF file used for search.
2. Retrieving data.
3. Verifying imported data by file comparison.

3.4.1 Creating the LDIF File Used for Search

The search LDIF files must be created with the DNs of all the entries in the import LDIF file. These search LDIF files are passed to an `ldapsearch` command to extract the imported data which creates entries in the same order as the import files for ease of comparison.

Using the data from Example 1, Example 2, and Example 3, the following search LDIF files must be created:

- `Search_PLDB_0.ldif`

```
MSISDN=9100000000000000,dc=MSISDN,ou=identities,
dc=operator,dc=com
serv=Auth,ou=mscCommonData,dc=operator,dc=com
ou=commonProfile,serv=Auth,ou=mscCommonData,
dc=operator,dc=com
```

- `Search_Distribution_0.ldif`

```
mscId=10000000000000000,ou=multiSCs,dc=operator,dc=com
assocId=12398,ou=associations,dc=operator,dc=com
```

- `Search_DS_1.ldif`

```
serv=CSPS,mscId=98765,ou=multiSCs,dc=operator,dc=com
ei=gprs,serv=CSPS,mscId=98765,ou=multiSCs,
dc=operator,dc=com
```

3.4.2 Retrieving Data

The imported data can be retrieved by using the commands described below. Each of these commands is executed in the same node as its associated import file. To retrieve the DSG data, the commands must be executed in the node where the master replica of that DSG is located. The queries must be made with an LDAP user with `readModeInPL=LP` so the information that is needed from the PLDB is also extracted from the local node.



Note: Retrieval time can be reduced in CUDB nodes by running several `ldapsearch` commands simultaneously, evenly distributed between the two SCs.

In case the CUDB system hosts the standard Ericsson HLR profile or the standard Ericsson IMS profile, it is not recommended to run more than 8 commands at the same time on each SC. This limit may vary for systems hosting other profiles. In any case, it is recommended to send each `ldapsearch` command to a different LDAP Front End (FE) in the node.

The following search commands retrieve the data for the imported entries:

- ```
#/opt/ericsson/cudb/ldapfe/bin/ldapsearch -x -LLL -H
ldap://PL2:389 -D <ldapUser> -w <ldapUserPwd> -b <Root DN>
-f <path_to_ldif_file>/Search_PLDB_0.ldif "(entryDN=%s)"
> <path_to_ldif_file>/Output_PLDB_0.ldif
```
- ```
# /opt/ericsson/cudb/ldapfe/bin/ldapsearch -x -LLL -H
ldap://PL2:389 -D <ldapUser> -w <ldapUserPwd> -b <Root DN>
-f <path_to_ldif_file>/Search_Distribution_0.ldif "(entry
DN=%s)" > <path_to_ldif_file>/Output_Distribution_0.ldif
```
- ```
/opt/ericsson/cudb/ldapfe/bin/ldapsearch -x -LLL -H
ldap://PL2:389 -D <ldapUser> -w <ldapUserPwd> -b <Root DN>
-f <path_to_ldif_file>/Search_DS_1.ldif "(entryDN=%s)" >
<path_to_ldif_file>/Output_DS_1.ldif
```

The `ldapsearch` command performs one search for each line in the given file. The given filter is treated as a pattern where the first occurrence of `%s` is replaced with a line from the file. This executes a sequence of search operations in which the filter is applied to each line in the given file, that is, each of the DNs.

The order of the entries retrieved with the `ldapsearch` command is the following:

- a Each entry starts with the DN followed by the list of object classes associated with the entry. The structural object class goes first and then the auxiliary object classes in the same order as introduced. The object class `top` is never part of the list, that is, the line `objectClass:top` never appears.
- b After the list of object classes, their attributes are returned in the same order. That is, first the attributes of the structural object class and then those of the auxiliary object classes in the same order as before.
- c The attributes of each object class are listed in the same order as in the schema definition of the object class.

The case of the attribute and object class names is as in the LDAP schema. Attribute values are returned as introduced.



For example, the Home Location Register Front End (HLR-FE) profile of a Multi Service Consumer (MSC) is returned as follows:

- Structural object class:  
objectclass:CUDBService
- Auxiliary OCs (same order as introduced):

```
objectclass:CP1
objectClass:CPR01
...
```

- Structural OC attributes (CUDBService):  
serv:CSPS
- Structural OC attributes (CP1):

```
MSISDN:9100000000000000
IMSI:1234567000000000
CDC:0
NAM:0
TSMO:0
...
```

- Second auxiliary OC attributes (CPR01):

```
IMSICHO:731022149181613
IMSICHODATE::5qWC
IMSICHOST:-89
...
```

### 3.4.3 File Comparison

The import is verified by comparing the imported LDIF files with the corresponding retrieved LDIF file. For example, comparing `Distribution_0.ldif` with `Output_Distribution_0.ldif`. The comparison between the files can be done line by line if the attributes of the entries in the import LDIF files are kept in the same order as the CUDB returns them.





## 4 Export Procedure

The export procedure can be used to extract data massively from CUDB, for example to perform external data audit processes. It offers a more efficient and faster method to extract the data than the standard LDAP interface by always accessing local database replicas.

The procedure is based on the `slapcat` command that writes the data exported to one or several LDIF files, one for each command launched.

This section describes the preconditions to be fulfilled before the export process is run, how the export procedure can be executed, limitations and recommendations for the procedure and how complex export queries may be produced by means of filters and the help of scripts.

### 4.1 Preconditions

Before starting the export procedure, the following conditions must be fulfilled:

- The CUDB nodes comprising the CUDB system must be up and running and there must be enough free space in the SCs to hold the exported data.
- Neither the CUDB reallocation process, nor the CUDB reconciliation process must be running on the CUDB system.

### 4.2 Execution

The export process is executed by running the `/opt/ericsson/cudb/OAM/bin/slapcat` command from an SC, passing the following parameters:

- **-f <configuration-file>**: Configuration file for the command. It allows the configuration of certain features of the export command using directives.

The default configuration file is `/cluster/home/cudb/dataAccess/ldapAccess/import-export/config/slaptool.conf`.

Each directive appears at the beginning of a line in the configuration file and the value set on the directive appears right after the directive name. See Section 4.2.1 on page 18 for details on the directives supported by the configuration file.

This configuration must be done in all CUDB nodes where an export is to be executed.

- **-l <output-file>**: Output LDIF file where the data exported will be stored.



Due to security reasons, only `root` or users in `cudbadmin` group (using the `sudo` command) can access the output LDIF file.

- `-s <subtree-dn>`: Only dump entries in the subtree specified by this DN.
- `-a <filter>` (Optional): LDAP filter used to select the entries to be exported. See Section 4.2.2 on page 21 for details on the format and allowed elements in the filter.
- `-o ldif-wrap=no`: Lines in the LDIF output file will not be wrapped with this option.

Example:

```
/opt/ericsson/cudb/OAM/bin/slappcat -f /cluster/home/cudb
/dataAccess/ldapAccess/import-export/config/slaptool.conf
-l Data_Node1.ldif -s "dc=operator,dc=com" -o ldif-wrap=no
```

The export command can only extract data from database replicas hosted in the CUDB node from which it is invoked. Depending on the desired replicas to be accessed in the export process (master or slave), the mastership allocation of the replicas and the entries to be extracted (all or a set of the whole CUDB entries) a different strategy may be followed. Several examples are shown in Section 4.2.3 on page 22 for different cases. For information on how to locate the desired replicas on CUDB nodes, refer to *CUDB System Administrator Guide*, Reference [2].

## 4.2.1 Configuration File Directives

This section describes the configuration file directives that are specific to the export process.

The configuration file where the parameters `filter-treatment` and `target=ds` can be modified is located in the below location:

```
/home/cudb/dataAccess/ldapAccess/import-export/config/slapp
tool-config/slaptool-options.conf
```

### 4.2.1.1 filter-treatment Directive

The `filter-treatment` directive is used to configure how the LDAP filters passed to the export command are applied on the LDAP entries subject to be exported.



*Table 2 Permitted Values for the filter-treatment Directive*

| <b>filter-treatment Directive</b> | <b>Description</b>                                                                                                                                                                                                                                                                            |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| subordinate                       | The export command searches for objects that match the filter passed to the command. It then returns those objects that match the filter together with all their children. This means that the children are always returned even if they do not individually match the filter. <sup>(1)</sup> |
| standard                          | The export command reads all the entries that are inside the specified scope. Then it checks if they match the filter and returns only those entries that match. This means that children are returned only if they individually match the filter. Default behavior.                          |

(1) Entries below the DEs are not considered as children, in case the subordinate filter option was used.

#### 4.2.1.2 target-ds Directive

The `target-ds` directive is used to specify the replicas to be accessed when exporting data from a CUDB node: only master replicas, only master replicas of DSGs and the local replica of the PLDB or only local replicas.

*Table 3 Permitted Values for the target-ds Directive*

| <b>target-ds Directive</b> | <b>Description</b>                                                                                                                                                                                                     |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| master                     | <code>slapcat</code> command only reads from master PLDB and master DSGs. The export is stopped when the mastership moves from a DS for which the export is not yet started or which is in progress. Default behavior. |



| target-ds Directive | Description                                                                                                                                                                                                                                                            |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| masterAndPl         | slapcat command only reads from master DSGs, and the local PLDB irrespective of whether it is a master or a slave replica. The export is stopped when the mastership moves from a DS - except PLDB - for which the export is not yet started, or which is in progress. |
| all                 | slapcat command reads from the local PLDB and DSG databases, irrespective of whether they are master or slave replicas.                                                                                                                                                |

The value of the `target-ds` directive depends on the goal of the export operations.

With the `master` value, the command makes full-system data exports more convenient because issuing a `slapcat` command in every CUDB node in the system results in a full-system data dump with no duplicated entries. Setting the `all` value and choosing a specific DSG, runs the `slapcat` command on slave replicas. This is recommended to avoid impacting the master replicas which will be handling the live traffic operations.

#### 4.2.1.3 export-batch-size Directive (Optional Directive)

The `export-batch-size` directive is used to configure the size of the group of entries that are retrieved together in each data store access.

This is an optional directive that must not be changed except for troubleshooting purposes.

Higher values provide better export performance, but stability issues may appear depending on the traffic load and the subscriber profiles used in the node. Therefore, it is not recommended to change the default value.

*Table 4 Permitted Values of the export-batch-size Directive*

| export-batch-size Directive | Description                                                                                                                                            |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| value                       | Size of the group of entries that are retrieved in each data store access.<br><br>Minimum value: 1<br><br>Maximum value: 1000<br><br>Default value: 50 |





#### 4.2.1.4 maxExportWorkerThreads Directive (Optional Directive)

The `maxExportWorkerThreads` directive is used to configure the number of worker threads that contribute to entry export.

This is an optional directive that must not be changed except for troubleshooting purposes.

Higher values provide better export performance, but stability issues (including traffic overload errors and high CPU consumption in the SCs) may appear, depending on the traffic load and the subscriber profiles used in the node. Therefore, it is not recommended to change the default value.

*Table 5 Permitted Values of the maxExportWorkerThreads Directive*

| <b>maxExportWorkerThreads Directive</b> | <b>Description</b>                                                                                                                          |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| value                                   | <p>Number of working threads contributing to the export process</p> <p>Minimum value: 1</p> <p>Maximum value: 6</p> <p>Default value: 4</p> |

#### 4.2.2 LDAP Filters

The export tool allows an LDAP filter to be passed as filter criteria for the data to be exported. The filter must conform to the string representation for search filters as defined in RFC 4515. See *RFC 4515 LDAP: String Representation of Search Filters*, Reference [6] for details.

To optimize the export commands, the `objectClass` attribute and the LDAP Search Indexes can be used in filters. Refer to *CUDB Application Integration Guide*, Reference [4] for more information.

In addition to the attributes and object classes defined in the CUDB and application schemas, the special `entryDS` attribute may be used in the filters of the export tool to filter data from specific database clusters. The `entryDS` attribute specifies the target DSG. If it is equal to 0 that means that the PLDB is the target database cluster. Therefore, for example, to obtain just LDAP entries from DSG 1 when running the export command, the `(entryDS=1)` filter must be passed as input to the tool.

##### 4.2.2.1 Limitations

The `entryDS` attribute is only optimized for a single level of nested filters.



For example, single level of nesting in the following filter limits the search to DS Unit Group 1 (entries in other DS Unit Groups are not accessed).

```
(& (serv=CSPS) (entryDS=1) (CAWTRACKING=1))
```

In the following example, even though the filter is applied correctly (only DS Unit Group 1 entries are returned), the export tool searches entries in all DS Unit Groups, therefore making the export less efficient.

```
(& (& (serv=CSPS) (entryDS=1)) (CAWTRACKING=1))
```

## 4.2.3 Export Examples

This section provides some examples of the use of the export tool for different export requirements.

### 4.2.3.1 Data Export from Master Replicas

If the export is to be done on master replicas the most convenient strategy is to set the `target-ds` directive in the configuration file to `master`, and then launch an export command on every node in the system holding any database cluster master replica. The CUDB root entry must also be passed as `subtree-dn` as shown below:

```
/opt/ericsson/cudb/OAM/bin/slapcat -f /cluster/home/cudb
/dataAccess/ldapAccess/import-export/config/slaptool.conf
-l <path_to_ldif_file>/Data_NodeX.ldif -s "<Root_DN>" -o
ldif-wrap=no
```

With this strategy, an LDIF file would be obtained on each node where the export command is launched, holding the data for the database cluster master replicas on that node.

If the data from each database cluster must be stored in a different file (that is, one file must be created for the PLDB data, and separate files for each DSG), then launch several export commands from each node, and on every execution, pass a filter with the `entryDS` attribute specifying the database cluster where the data must be exported.

For example, in a CUDB system where one of the nodes is hosting the master replica for the PLDB and for DSG1 and DSG3, the following export commands must be launched from this node:

```
/opt/ericsson/cudb/OAM/bin/slapcat -f /cluster/home/cudb
/dataAccess/ldapAccess/import-export/config/slaptool.conf
-l <path_to_ldif_file>/Data_PLDB.ldif -s "<Root DN>" -a
"entryDS=0" -o ldif-wrap=no
```

```
/opt/ericsson/cudb/OAM/bin/slapcat -f /cluster/home/cudb
/dataAccess/ldapAccess/import-export/config/slaptool.conf
```



```
-l <path_to_ldif_file>/Data_DSG1.ldif -s "<Root DN>" -a
"entryDS=1" -o ldif-wrap=no
```

```
/opt/ericsson/cudb/OAM/bin/slapcat -f /cluster/home/cudb
/dataAccess/ldapAccess/import-export/config/slaptool.conf
-l <path_to_ldif_file>/Data_DSG3.ldif -s "<Root DN>" -a
"entryDS=3" -o ldif-wrap=no
```

To have the complete set of data in LDIF files in the above example, similar export commands must be launched in the rest of nodes hosting master replicas for any database cluster.

**Note:** The export can also be done on a per-branch basis by specifying the branch to export with the `-s` parameter.

#### 4.2.3.2

#### Complete Data Export from Slave Replicas

If the export is to be done on slave replicas then the `target-ds` directive in the configuration file must be set to `all` and the data must be extracted in the following groups:

- *PLDB entries not including DEs:* An export command must be run for each branch of PLDB entries not including DEs in the node hosting the desired PLDB replica.

The below example shows how to extract entries from the CUDB predefined identities and common branches:

```
/opt/ericsson/cudb/OAM/bin/slapcat -f /cluster/home
/cudb/dataAccess/ldapAccess/import-export/config/sla
ptool.conf -l <path_to_ldif_file>/ServCommon_0.ldif -s
"ou=servCommonData,<Root DN>" -o ldif-wrap=no
```

```
/opt/ericsson/cudb/OAM/bin/slapcat -f /cluster/hom
e/cudb/dataAccess/ldapAccess/import-export/config/sl
aptool.conf -l <path_to_ldif_file>/MscCommon_0.ldif -s
"ou=mscCommonData,<Root DN>" -o ldif-wrap=no
```

```
/opt/ericsson/cudb/OAM/bin/slapcat -f /cluster/ho
me/cudb/dataAccess/ldapAccess/import-export/config
/slaptool.conf -l <path_to_ldif_file>/Iden_0.ldif -s
"ou=identities,<Root DN>" -o ldif-wrap=no
```

- *DEs:* An export command must be run for each DE parent branch in the node hosting the desired PLDB replica. In this case, to extract only PLDB entries, the `entryDS=0` filter must be used.

The below example shows how to extract entries from the CUDB predefined multiSCs and associations DEs parent branches:

```
/opt/ericsson/cudb/OAM/bin/slapcat -f /cluster/ho
me/cudb/dataAccess/ldapAccess/import-export/config/
```

```
slaptool.conf -l <path_to_ldif_file>/Msc_PLDB.ldif -s
"ou=multiSCs,<Root DN>" -a "(entryDS=0)" -o ldif-wrap=no

/opt/ericsson/cudb/OAM/bin/slapcat -f /cluster/home/cudb/dataAccess/ldapAccess/import-export/config/
slaptool.conf -l <path_to_ldif_file>/Assoc_PLDB.ldif
-s "ou=associations,<Root DN>" -a "(entryDS=0)" -o
ldif-wrap=no
```

- *DS entries:* An export command must be run for each DE parent branch and for every DSG in the system in the node hosting the desired replica. In this case, to extract only the entries for a given DSG, the `entryDS=<DSG>` filter must be used.

The below example shows how to extract DS entries under the predefined DEs in CUDB from DSG 1 from a node hosting a slave replica for DSG1:

```
/opt/ericsson/cudb/OAM/bin/slapcat -f /cluster/home/cudb/dataAccess/ldapAccess/import-export/config/
slaptool.conf -l <path_to_ldif_file>/Msc_DS_1.ldif -s
"ou=multiSCs,<Root DN>" -a "(entryDS=1)" -o ldif-wrap=no

/opt/ericsson/cudb/OAM/bin/slapcat -f /cluster/home/cudb/dataAccess/ldapAccess/import-export/config/
slaptool.conf -l <path_to_ldif_file>/Assoc_DS_1.ldif
-s "ou=associations,<Root DN>" -a "(entryDS=1)" -o
ldif-wrap=no
```

#### 4.2.4 Limitations and Recommendations

Consider the following limitations and recommendations when preparing an export procedure:

- Exports are data-intensive operations that impose an extra load on the CUDB system. If extra load causes problem, the export operation must be run at low traffic hours, on slave replicas, or both. The `target-ds` directive must be properly configured for the export operation to be performed on slave replicas.
- It is recommended to execute only one `slapcat` command at a time on each SC.

### 4.3 Complex Export Queries Using Filters and Scripts

The export tool, supported by small scripts, can perform more complex search operations such as retrieving all the entries of an MSC with International Mobile Subscriber Identity (IMSI) or Mobile Station ISDN Number (MSISDN) in a specified range. The reason for requiring the combination of scripts and the export tool is that the latter does not support alias dereferencing.



The steps of Example 4 show how a complex query extracts entries of MSCs with a MSISDN in a specified range. Data of MSC entries is stored in a particular DS. This query is performed through a combination of searches and scripts. This example is specific to the CUDb predefined DIT, but the procedure would be the same for other LDAP trees.

#### *Example 4 Complex Export Query Using Filters and Scripts*

The steps of creating a complex export query are as follows:

1. Execute the `slapcat` command as shown below to extract all the Auth entries of MSCs with MSISDN in a specified range.

```
/opt/ericsson/cudb/OAM/bin/slapcat -f /cluster/home/cudb/dataAccess/ldapAccess/import-export/config/slaptool.conf -s "dc=MSISDN,ou=identities,<Root DN>" -a "(&(MSISDN<=913391886)(MSISDN>=901449889))" -o ldif-wrap=no > /local/slapcatTmp.ldif
```

2. Create filters for each different DE returned by `slapcat` by using a script, for example `CreateMSCsfilters.sh`:

```
cd /local
```

```
vi CreateMSCsfilters.sh
```

An example of the script is shown below:

```
#!/bin/bash
OLD=0;
CURRENT=0;
SLAPD_CONF_FILE="/home/cudb/dataAccess/ldapAccess/ldapFe/config/slapd.conf";
ROOT_DN=`cat ${SLAPD_CONF_FILE} | grep operator_suffix | awk '{ print $2 }' | tr -d ' '`;

while read data; do
 # echo "data = $data"
 CURRENT=$(echo $data | awk '/aliasedObjectName:/ {print}' | awk ' BEGIN { FS = "mScId=" } ; {print $2}' |
 { FS = ", " } ; {print $1}')
 if [-n "$CURRENT"]; then
 if [["$OLD" -ne "$CURRENT"]]; then
 echo "serv=Auth,mScId=$CURRENT,ou=multiSCs,${ROOT_DN}"
 OLD=$CURRENT
 fi
 fi
done < /local/slapcatTmp.ldif
```

Execute the following script to fetch the data:

```
chmod +x /local/CreateMSCsfilters.sh
```

```
/local/CreateMSCsfilters.sh > /local/MSCfiltersOut.txt
```

The following filter is created for each different DE returned by `slapcat` command:

```
serv=Auth,mScId=100913391886,ou=multiSCs,dc=operator,dc=com
```



3. Execute the below command to fetch the list of filters in *<searchFile>*, used to obtain the required entries:

```
/opt/ericsson/cudb/ldapfe/bin/ldapsearch -x -LLL
-H ldap://PL0:389 -D <trafficUser> -w <traffic_pwd> -b
<RootDN> -f /local/MSCFiltersOut.txt "(entryDN=%s)"
```



## 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 Subscription Reallocation*
- [2] *CUDB System Administrator Guide*
- [3] *CUDB Glossary of Terms and Acronyms*
- [4] *CUDB Application Integration Guide*

### **Other Documents and Online References**

- [5] *The LDAP Data Interchange Format (LDIF) - Technical Specification, RFC 2849* <http://www.rfc-editor.org/rfc/rfc2849.txt>
- [6] *LDAP: String Representation of Search Filters, RFC 4515* <http://www.rfc-editor.org/rfc/rfc4515.txt>