

CUDB Optimized Subtree Searches

User Guide

Copyright

© Ericsson AB 2018. 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	Target Groups	1
1.2	Revision Information	1
1.3	Typographic Conventions	1
2	Overview	2
3	Optimized Subtree Searches Configuration	4
3.1	Configure an Optimized Subtree Search	5
3.2	Delete an Optimized Subtree Search	5
3.3	Update an Optimized Subtree Search	6
3.4	Creating Optimized Subtree Search in the Data Model	6
3.5	Assigning Optimized Subtree Search to User	7
3.6	Removing Optimized Subtree Search from User	7
3.7	Deleting Optimized Subtree Search	7
3.8	Modifying Optimized Subtree Search configuration	8
	Glossary	9
	Reference List	10





1 Introduction

This document describes the concepts, configurations, and procedures of the Optimized Subtree Searches function in the Ericsson Centralized User Database (CUDB) system.

1.1 Target Groups

The intended audience of this document is the same audience as of the [CUDB System Administrator Guide](#): Ericsson personnel and operators. A general knowledge of the CUDB system is assumed.

1.2 Revision Information

Rev. A

Initial release.

1.3 Typographic Conventions

Typographic Conventions can be found in the following document:

- [Typographic Conventions](#)

2 Overview

In regular LDAP search operations, a query with scope set to 'wholeSubtree' requires to fully scan the complete subtree of descendant entries before applying the filter expression to those child objects. Those kind of queries usually have a performance impact as the size of the subtree gets bigger (more time and resources for drilling down all the populated descendants).

The Optimized Subtree Searches function can be used to selectively specify and refine the original request to be performed over a given set of descendant entries of the baseObject or branches instead of scanning everything below the search base.

Optimized subtree queries are configured individually for those application frontends/LDAP users who would benefit from them.

This feature could also help to reduce number of queries issued to retrieve a group of entries by using a single subtree operations but restricted to look up a subset of the entries in the subtree instead.

Figure 1 describes how subtree search works: A subtree query starting at msclId level would require to scan and evaluate the filter against all the entries in the orange box, even if the wanted entries are just the ones in the blue boxes.

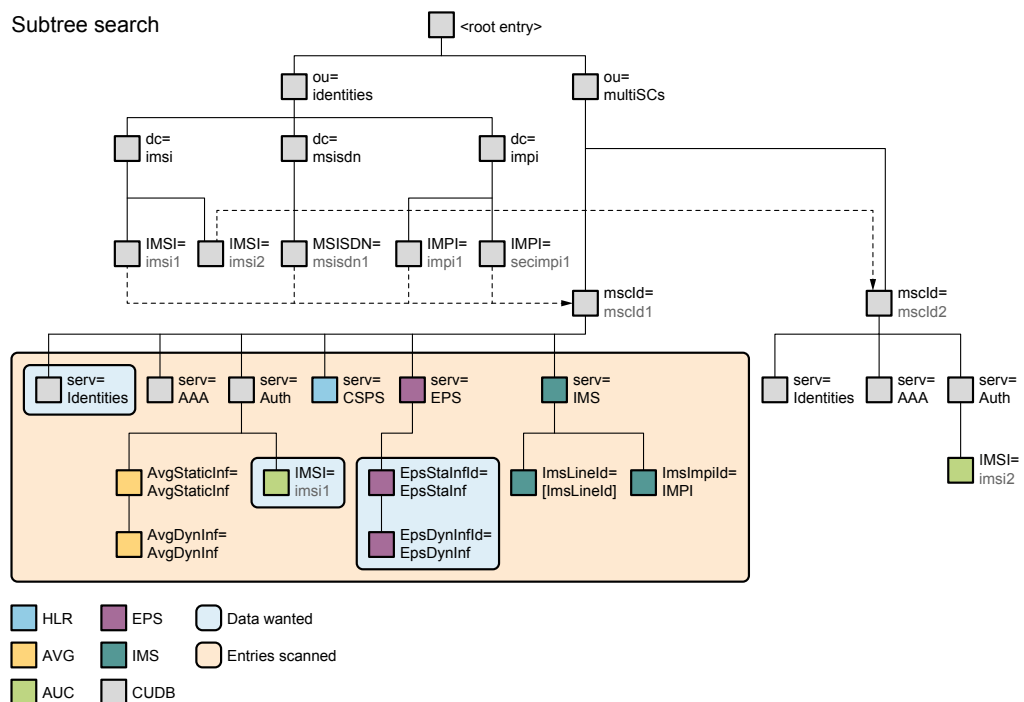


Figure 1 Subtree search



With the Optimized Subtree searches it is possible to configure what entries evaluate instead of scanning the whole subtree, therefore performing the operation more efficiently.

3 Optimized Subtree Searches Configuration

Optimize Subtree Searches function is defined in configuration YAML file with following structure which follows subtree request from [Figure 1](#):

```
hints:
  - hint: HINT-1
    BaseDnPattern: mscId=*,ou=multiscs,*
    Scope: sub
    Filter: (|(|(objectClass=EpsStaticInf)(objectC →
lass=EpsDynInf))(objectClass=mscIdentities)(objectClass=AU1))
    staticTargets:
      - srdn: serv=identities
      - srdn: EpsStaInfId=EpsStaInf,serv=EPS
      - srdn: EpsDynInfId=EpsDynInf,EpsStaInfId=EpsS →
taInf,serv=eps
      - srdn: IMSI=*,serv=Auth
  - hint: HINT-2
    .
    .
    .
```

Configuration is defined with the following fields:

- There can be only one **hints** section with multiple **hint** sections. Each **hint** has a unique name inside its file, for example HINT-1.
- **BaseDnPattern** has value that matches baseObject dn in search request and it must not include alias entries. It supports wildcards for variable rdn value that are defined with '*' character e.g. mscid=*. Root dn of incoming request is likely to be different for different operators, so it may be a good idea to have **BaseDnPattern** ending with a wildcard also like in example above. That way the optimized subtree search configuration can be used in any operator.
- **Scope** represents the search scope of the incoming request .
- **Filter** represents the filter of the incoming request. It can contain a wildcard, but it will only work if the application sends a filter with a wildcard. It cannot match a fixed value with a wildcard like in case of **BaseDnPattern**.
- Section **staticTargets** contains **srdn** keys which are the relative RDNs starting from the baseObject in the searchRequest (or the BaseDnPattern here) where the entries of interest are, that is the entries where the wanted data is. Only entries specified in this section will be returned, thus the base object will not be returned even if it fullfils filter. Attribute name in RDN must be known in advance, but RDN value can be unknown. In that case wildcard can be used for RDN value e.g. IMSI=*,serv=Auth. Such definition will return all entries one level below serv=Auth that have rdn attribute name IMSI.



Optimization will be applied under the following conditions:

1. Ldap user is configured with a subtree optimization.
2. Scope and filter of the incoming request are matching one of the hint definitions in the configuration file connected with the user.
3. The baseObject (possibly dereferenced) in the SearchRequest, matches BaseDnPattern of hint found in condition 2.

Standard behavior is applied if one of above conditions is not fulfilled.

Optimization is only applied to entries stored in DS, thus the srdn keys can only contain entries stored in DS. In case the wanted entry is non-distributed data (data stored in PLDB) accessed through DSG alias entry, then srdn section must contain DSG alias entry itself.

Note: BaseDnPattern started with serv=auth results in specific behavior described in *Base DN Transformation in LDAP operations over Authentication Data Operations (serv=Auth)* in CUDB LDAP Interwork Description. In this case BaseDnPattern can be defined on DE level including serv=auth in srdn section.

```
BaseDnPattern: mscId=*,ou=multiscs,*
- srdn:          AvgStaticInfId=AvgStaticInf,serv=Auth
- srdn:          AvgDynInfId=AvgDynInf, AvgStaticInfId=
AvgStaticInf,serv=Auth →
```

3.1 Configure an Optimized Subtree Search

To configure an Optimized Subtree Search, perform the following steps:

Steps

1. Create the Optimized Subtree Search, see [Creating Optimized Subtree Search in the Data Model](#) on page 6. Repeat this procedure on every CUDB node.
2. Assign the Optimized Subtree Search to the user, see [Assigning Optimized Subtree Search to User](#) on page 7.

3.2 Delete an Optimized Subtree Search

To delete an Optimized Subtree Search, perform the following steps:

Steps

1. Unassign the Optimized Subtree Search from user, see [Removing Optimized Subtree Search from User](#) on page 7.



2. Delete the Optimized Subtree Search, see [Deleting Optimized Subtree Search](#) on page 7. Repeat this procedure on every CUDB node.

3.3 Update an Optimized Subtree Search

To update an Optimized Subtree Search, perform the following steps:

Steps

1. Update Optimized Subtree Search config, see [Modifying Optimized Subtree Search configuration](#) on page 8. Repeat this procedure on every CUDB node.
2. Reassign the Optimized Subtree Search to the user, see [Assigning Optimized Subtree Search to User](#) on page 7.

3.4 Creating Optimized Subtree Search in the Data Model

Prepare and upload the configuration file with .yaml extension, for example TestHint.yaml, in the /home/cudb/dataAccess/ldapAccess/ldapFe/config/subtreeSearchConfig.

To create an Optimized Subtree Search, add an instance of CudbSubtreeSearchConfig class in the configuration model. For more information about the model, refer to [CUDB Node Configuration Data Model Description](#).

Example 1 Optimized Subtree Search configuration

```
ManagedElement=1,CudbSystem=1,CudbLocalNode=<nodeId>,CudbLdapAccess=1,CudbSubtreeSearchMgmt=1 →
(CudbSubtreeSearchMgmt=1)>show
CudbSubtreeSearchMgmt=1
  CudbSubtreeSearchConfig=1
  CudbSubtreeSearchConfig=2
(CudbSubtreeSearchMgmt=1)>configure
(config-CudbSubtreeSearchMgmt=1)>CudbSubtreeSearchConfig=3
(config-CudbSubtreeSearchConfig=3)>subtreeSearchConfigName=TestHint
(config-CudbSubtreeSearchConfig=3)>commit
(CudbSubtreeSearchConfig=3)>up
(CudbSubtreeSearchMgmt=1)>show
CudbSubtreeSearchMgmt=1
  CudbSubtreeSearchConfig=1
  CudbSubtreeSearchConfig=2
  CudbSubtreeSearchConfig=3
(CudbSubtreeSearchMgmt=1)>
```

Configuration file name is unique and subtreeSearchConfigName attribute must match that name, but without extension, like in the example above.

After This Task

Refer to the Object Model Modification Procedure in [CUDB Node Configuration Data Model Description](#) for more information on all the steps required to modify the object model.



3.5 Assigning Optimized Subtree Search to User

To assign an Optimized Subtree Search to the user, set the `subtreeSearchConfigName` attribute of the already created instance of the `CudbLdapUser` class belonging to LDAP user whose subtree searches are intended to be optimized. The value of the `subtreeSearchConfigName` attribute is the identifier of the Optimized Subtree Search connected to the user, and its value must match the `subtreeSearchConfigName` attribute value of the `CudbSubtreeSearchConfig` class. For more information, refer to [CUDB Node Configuration Data Model Description](#) CUDB Node Configuration Data Model Description.

Refer to the Object Model Modification Procedure in [CUDB Node Configuration Data Model Description](#) for more information on all the steps required to modify the object model.

Update the information of the LDAP users on every CUDB node apart from the one selected in this section by following the procedure described in [CUDB System Administrator Guide](#).

Note: The same Optimized Subtree Search can be assigned to several LDAP users. An LDAP user can have only one Optimized Subtree Search assigned.

3.6 Removing Optimized Subtree Search from User

To remove the Optimized Subtree Search from the user, remove the value of the `subtreeSearchConfigName` attribute of the instance of the `CudbLdapUser` class belonging to LDAP user whose subtree searches are optimized. For more information, refer to [CUDB Node Configuration Data Model Description](#).

Refer to the Object Model Modification Procedure in [CUDB Node Configuration Data Model Description](#) for more information on all the steps required to modify the object model.

Update the information of the LDAP users on every CUDB node apart from the one selected in this section by following the procedure described in [CUDB System Administrator Guide](#).

3.7 Deleting Optimized Subtree Search

To delete an Optimized Subtree Search, remove the instance of the `CudbSubtreeSearchConfig` class from the data model. For more information, refer to [CUDB Node Configuration Data Model Description](#).

Refer to the Object Model Modification Procedure in [CUDB Node Configuration Data Model Description](#) for more information on all the steps required to modify the object.



Note: Only Optimized Subtree Searches that are not connected to any LDAP user can be deleted.

After This Task

After an Optimized Subtree Search is deleted through the data model, also delete configuration file in the `/home/cudb/dataAccess/ldapAccess/ldapFe/config/subtreeSearchConfig/directory`.

3.8 Modifying Optimized Subtree Search configuration

An existing Optimized Subtree Search can also be modified. It is necessary to add changes in configuration file and then rename it to for example TestHint01. After the file has been updated, it is necessary to change subtreeSearchConfigName of CudbSubtreeSearchConfig class to this new name.

Refer to the Object Model Modification Procedure in [CUDB Node Configuration Data Model Description](#) for more information on all the steps required to modify the object model.



Glossary

For the terms, definitions, acronyms and abbreviations used in this document, refer to CUDB Glossary of Terms and Acronyms



Reference List

CUDB Documents

1. CUDB System Administrator Guide
2. CUDB Node Configuration Data Model Description
3. CUDB Troubleshooting Guide
4. CUDB Glossary of Terms and Acronyms