

CUDB Notifications

Facility Description

Copyright

© Ericsson AB 2016-2018. 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	Document Purpose and Scope	1
1.2	Revision Information	1
1.3	Target Groups	2
1.4	Prerequisites	2
1.5	Typographic Conventions	3
2	Overview	4
2.1	Prerequisites	4
2.2	Architecture	4
2.3	Description	5
2.4	Dependencies and Interactions	7
3	Operation and Maintenance	10
3.1	Activation	10
3.2	Configuration	10
3.3	Fault Management	15
3.4	Performance Management	16
3.5	Security	16
3.6	Logging	16
	Glossary	17
	Reference List	18





1 Introduction

This document provides a description of the optional notifications feature in the Ericsson Centralized User Database (CUDB).

1.1 Document Purpose and Scope

The purpose of this document is to describe the optional notifications feature from a functional and operational point of view.

1.2 Revision Information

Rev. A

This document is based on 9/15534-HDA10403/9 with the following changes:

- Updated hardware information.

Rev. B

Other than editorial changes, this document has been revised as follows:

- [Architecture](#) on page 4: Added information on sets of DEs.
- [Description](#) on page 5: Added information on regular expression pattern.
- [Notifications Parameters](#) on page 11: Updated information on `CudbNotificationObjectClass` and the `monitor` type.
- [Configuration Examples](#) on page 14: Added new section with examples of configuring new CUDB Notification Events using regular expressions.

Rev. C

Other than editorial changes, this document has been revised as follows:

- [Notifications Parameters](#) on page 11: Updated to reflect support of extended POSIX regular expressions.
- **Section 4.2.3.2 Using Regular Expressions:** Updated to reflect support of POSIX regular expressions.

Rev. D

Other than editorial changes, this document has been revised as follows:



- [Architecture](#) on page 4: Updated description.
- [Description](#) on page 5: Updated description of the **Monitored** type.
- [Notifications Parameters](#) on page 11: Updated section.
- [Configuration Examples](#) on page 14: Updated section.
- **Section 4.2.3.1 Creating a New Notification Event:** Section contents moved to the Creating a New Notification Event section of [CUDB System Administrator Guide](#).
- **Section 4.2.3.2 Using Regular Expressions:** Section contents revised and moved to [Configuration Examples](#) on page 14.

Rev. E

Other than editorial changes, this document has been revised as follows:

- [Preconfigured Notifications](#) on page 15: Updated MS Purged in SGSN, International Mobile Subscriber Identity (IMSI), and Collision Detection Counter (CDC).
- [Fault Management](#) on page 15: Added list of triggered alarms.

Rev. F

Other than editorial changes, this document has been revised as follows:

- [Dependencies and Interactions](#) on page 7: Added new interaction.

Rev. G

Other than editorial changes, this document has been revised as follows:

- [Dependencies and Interactions](#) on page 7: Updated the method of calculating the number of needed TCP connections (numOfConnections).

1.3 Target Groups

As this document provides overall information about the notifications feature, it is intended for system developers, testers, administrators, or operators.

1.4 Prerequisites

Users of this document must have an overall knowledge of the CUDB system basics.



1.5 Typographic Conventions

Typographic Conventions can be found in the following document:

- Typographic Conventions



2 Overview

The notifications feature is used to inform Front End (FE) applications about modifications in data they store in a CUDB system.

2.1 Prerequisites

This section is not applicable to this feature.

2.2 Architecture

The notifications feature works on a Distribution Entry (DE) basis to provide information about changes in the attribute data stored in entries below the DEs in the Lightweight Directory Access Protocol (LDAP) tree. In other words, it reports modifications of data stored in the Data Store (DS) layer of a certain DE or set of DEs.

Figure 1 shows the notifications module, its components, and relationships with other CUDB modules and external FEs.

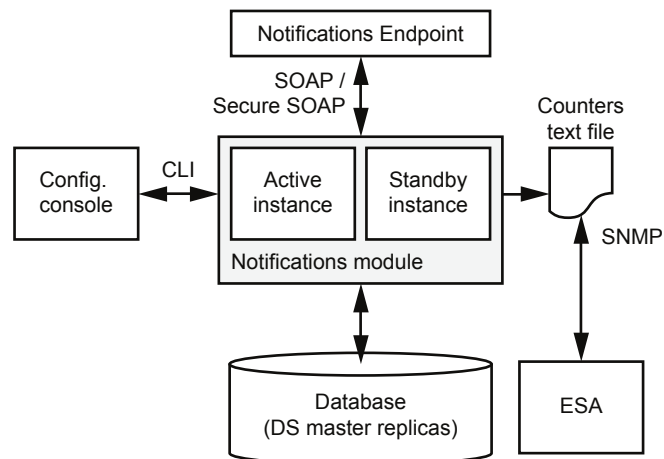


Figure 1 Notifications Module

The notifications module subscribes to the database events to monitor attribute changes, check for certain attribute values, and retrieve attribute contents to build the notifications.

Notifications to applications are sent as Simple Object Access Protocol (SOAP) messages over Hypertext Transfer Protocol (HTTP). CUDB acts as the SOAP



client and the applications FEs act as the SOAP servers (referred to as endpoints from now on). For more information on the CUDB SOAP interface, refer to [CUDB SOAP Interwork Description](#).

Use of Transport Layer Security (TLS) key and certificate is also possible in the SOAP interface. For more information on security in CUDB, refer to [CUDB Security and Privacy Management](#).

The notifications feature has the following configuration options:

- It is possible to configure the data to be monitored for change, the data whose value must be checked before sending notifications, as well as the data sent into a notification.
- It is possible to configure several different notification types in parallel for the same or different applications FEs.
- It is possible to configure some feature behavior parameters, such as the destination of each notification message to be sent.

The parameters above are configured through the CUDB configuration Command Line Interface (CLI).

The notifications module is deployed in all payloads. However, only two instances run in active-standby model. The rest of the instances are waiting to enter in standby mode in case one of the active-standby instances fail. The active entity monitors the database events to be notified by subscribing to the master replicas of the DSGs hosted by the CUDB node. When the active entity fails, the standby entity becomes active and monitors the events instead of the active. For more information on high availability, refer to [CUDB High Availability](#).

The module gathers internal statistics on failed and successful operations. These statistics are downloaded to counters, which are managed by the Ericsson Simple Network Management Protocol (SNMP) Agent (ESA). Additionally, the module produces logging information about the performed operations and the operation errors.

2.3 Description

The notifications feature has the following three types of attributes:

- **Monitored:** triggers the processing of a notification when changes are made in tracked data in the database.

A notification will be sent in the following cases:

1. There is a change in the value of at least one monitored attribute in an entry that matches the specified dn/rdn in the Notification Event configuration.
2. After the Checked and Related configuration is evaluated successfully.



The way rdn/dn information is specified in the configuration allows monitoring single entries (fixed rdn) or a group of entries that matches a dn pattern regular expression (full dn with Portable Operating System Interface (POSIX) regular expressions).

Note: To trigger the processing of the notification, all of the configured monitored attributes must refer to the same LDAP entry or set of LDAP entries matching the regular expression which was specified in the CUDB Notification Event configuration.

Moreover, it is possible to monitor the creation or deletion of whole entries by using the `monitorAll` option (see [Notifications Parameters](#) on page 11 for further information).

The following database events are considered as attribute changes:

- Changing an attribute value to other than the previous one.
 - In non-multi-valued attributes, changing from no-value (`null` in the database) to a specific value.
 - In non-multi-valued attributes not using the `monitorAll` option, changing from a specific value to no-value (`null` in the database), except if the object class containing the attribute is not removed. If the object class was deleted, this event is not considered as an attribute change.
 - In non-multi-valued attributes using the `monitorAll` option, changing from a specific value to no-value (`null` in the database).
 - In multi-valued attributes, when at least one value is added to or deleted from the attribute, or when all the values from the attribute are replaced with a new value.
- **Checked:** A notification is sent if at least one of the attributes has a certain and configured value, which must be different from no-value. Multi-valued attributes are not allowed to be checked.
- **Related:** The attributes to be included in the notification message. The notification is sent if the attributes are defined in the database and have a value (that is, if they are not null). When a multi-valued attribute is included in a notification, all the values currently held in the attribute are sent in it.

It is possible to define as many notifications as needed, but each one of them must have at least one monitored attribute. A notification is triggered if a monitored attribute changes and if the checked attributes (if defined) fulfills the comparison condition. All the attributes included in the notification are related to the same DE, as the feature is DE-based. [Preconfigured Notifications](#) on page 15 shows an example of a notification definition.

A list of notification endpoints is also defined per notification. Each endpoint is characterized by a certain weight, which determines how often it receives



notifications. Endpoints having a weight different from zero are included in a Weighted Round Robin (WRR) algorithm. This means, for example, if three endpoints (EP1, EP2 and EP3) are defined, each having weights 4, 2 and 1 respectively, then EP1 receives twice the number of notifications than EP2, and four times the number of notifications than EP3. On the other hand, if an endpoint has weight=0 set, it is excluded from the WRR algorithm and receives all of the notifications. Therefore, all the endpoints with weight=0 receive notifications in a broadcast. The order of WRR endpoints where notifications are being sent to is equal with the configuration order of the WRR endpoints in the configuration model.

2.4 Dependencies and Interactions

The notifications module works in an autonomous way in the CUDB system and does not interfere with the LDAP traffic operations performed by the system up to the characterized performance limits. However, the following limitations in the feature and interactions with other features must be considered:

- In general, only binary content up to 4 KB is supported. Only string and integer attributes are supported in notifications (for more information about mapping LDAP to internal database types, refer to the related table in [CUDB Application Integration Guide](#)). `OctetString` attributes are supported with the following limitations:
 - For monitor and related attributes, `OctetString` binary attributes up to 4 KB are supported.
 - For check attributes, `OctetString` binary attributes up to 1530 bytes are supported.
 - `OctetString` attributes greater than 4 KB are specifically not supported in any case.
- Depending on the feature configuration, unexpected notifications can be received during CUDB maintenance tasks, such as backup and restore operations and reallocation and reconciliation procedures because of deletion and creation of DEs in the database.
- PL data is not supported. The feature is limited to work with data stored in the DS only.
- In monitored attributes, changing from no-value (*null* in the database) to a specific value is considered a change.
- In monitored attributes, changing from a specific value to no-value is considered a change, except when the object class containing the attribute is also removed.
- Multi-valued attributes are not allowed to be included as checked values in the notifications.



- CUDB sends one notification message per LDAP operation that cause a change of any of the monitored attributes. However, consecutive LDAP modify operations received by CUDB within a very short period of time (about 1 second) over the same LDAP entry can be managed as only one LDAP operation according to the notifications feature. This sends just one notification message for all the modifications instead one per LDAP operation. In this case, the data in such notification message can contain inaccurate information in the sent multi-valued attributes.
- Operational attributes (`entryDS`, `structuralObjectClass`, `createTimestamp`, `modifyTimestamp`) are not supported in notifications.
- It is possible that the notification-related software processes show a high memory consumption. However, this situation is very unlikely, as it requires certain conditions to happen simultaneously, such as many notification events configured, high notifications rate and several notification endpoints showing a slow response time.
- The maximum number of available TCP connections towards the notifications endpoints is 1000. This must be taken into account as an upper limit when configuring the Notifications feature. The number of needed TCP connections (`numOfConnections`) is calculated as follows:

`numOfConnections = numConnectionsEvent1 + numConnectionsEvent2 + numConnectionsEvent3 + ...`

where `numConnectionsEvent<X>` is the number of TCP connections needed per each defined `CudbNotificationEvent`.

Also, `numConnectionsEvent<X>` is calculated as follows:

`numConnectionsEvent<X> = numberOfEndpoints<X> * numberOfSoapThreads<X>`

where `numberOfSoapThreads<X>` is the number of SOAP threads and `numberOfEndpoints<X>` is the number of endpoints defined for `CudbNotificationEvent<X>`. For information on both multiplication factors, for each `CudbNotificationEvent`, refer to [CUDB Node Configuration Data Model Description](#).

See the examples below for three notification events defined with the following parameters:

- `CudbNotificationEvent<1>`: 20 SOAP threads and 4 endpoints
- `CudbNotificationEvent<2>`: 30 SOAP threads and 2 endpoints
- `CudbNotificationEvent<3>`: 40 SOAP threads and 1 endpoint

`numOfConnections = 20*4 + 30*2 + 40*1 = 80 + 60 + 40 = 180 TCP connections`



- When Data Repair is invoked, it may perform modifications to LDAP entries. In such case, notifications will be sent just like as if the modification had been performed by a node external to CUDB, such as by PG or any application FE.

Furthermore, there are some limitations about the lost notifications when the active instance dies. This loss is due to:

- AMF Switchover time.
- Time to initialize the process and the NDB subscription.
- Internal queues of the process.
- Consecutive LDAP insert and delete operations, received by CUDB within a very short period of time and over the same LDAP entry that is also being monitored, can result in getting an error for the delete LDAP operation.



3 Operation and Maintenance

This section describes the Operation and Maintenance of the notifications feature.

For information on troubleshooting this feature, refer to [CUDB Troubleshooting Guide](#).

3.1 Activation

This section describes the activation of the notifications feature.

3.1.1 Prerequisites

The notifications feature is activated through the CUDB configuration model. The necessary prerequisites and permissions are listed in [CUDB Node Configuration Data Model Description](#).

3.1.2 Activating Notifications

The notifications feature activation status is determined by attribute `enabled` in class `CudbNotifications` of the CUDB configuration model. If this boolean attribute is set to *false*, the sending notification is avoided.

3.2 Configuration

The notifications module is configured by a set of classes and attributes defined in the CUDB configuration model to provide the feature configuration options.

3.2.1 Configuring Notifications

The types of configuration parameters related to the notifications module are as follows:

- **Module configuration parameters:** applicable for the whole module. They can be modified at runtime, taking immediate effect after they are validated, without the need of disabling the feature and then enabling it again. Module configuration parameters are as follows:
 - Attribute `enabled`, class `CudbNotifications`: Enables or disables the notifications feature.
 - Attribute `maxReattempts`, class `CudbNotifications`: Sets the number of reattempts to resend a notification in case of error.



- Attribute `retryTime`, class `CudbNotifications`: Sets the time between the attempts.
 - **Notification related parameters:** applicable for each defined notification, not to the whole feature. Notification related parameters are as follows:
 - Endpoint definition parameters, detailed in class `CudbNotificationEndPoint`. Endpoint address and weight are defined here. These parameters can be modified at runtime, taking immediate effect after they are validated, without the need of disabling the feature and then enabling it again.
 - Monitored, checked and related attributes of the notification, defined in classes `CudbNotificationObjectClass` and `CudbNotificationAttr`. These parameters can be modified at runtime, but they need disabling and then enabling the feature again in order to take effect, which results in that notifications are not sent during this period.
- See [Notifications Parameters](#) on page 11 for further details on the parameters and classes, as well as the relationship and logic among them.
- **Security related parameters:** It is possible to configure secure endpoints by specifying TLS key and certificate as configuration parameters for the notifications. For more information, refer to [CUDB Security and Privacy Management](#).

3.2.2 Notifications Parameters

`CudbNotifications` is the root class hosting all the configuration data related to the notifications feature. `CudbNotifications` contains the module configuration parameters, and all notifications defined in the CUDB system hang on it.

Notifications are defined by the `CudbNotificationEvent` class. This class has the same number of instances as the number of notifications created in the system and each instance contains the configuration data related to each notification, which is divided into the following two parts:

- Endpoint definition parameters are detailed in the `CudbNotificationEndPoint` class. The class has the same number of instances as the number of the notification endpoints, and each instance contains the related data for each endpoint.
- Notification object classes are detailed in the `CudbNotificationObjectClass` class. One instance of this class has to be defined per one LDAP entry, or per set of the LDAP entries that can be defined with one POSIX extended regular expression, and object classes that contain LDAP attributes involved in the notification. Instances of these classes can be the following types:



- **monitor**: LDAP attributes in `CudbNotificationObjectClass` instances of the `monitor` type are monitored in the notification. Monitored attributes can be configured for a specific entry or for a set of entries.

The `name` attribute contains the object class of these LDAP attributes.

The `dn` attribute must contain the Distinguished Name (DN), with or without the DE DN part, of the entry for which notification will be triggered. It can be configured in two different ways, as follows:

- For a specific entry, by setting the DN of the entry.
- For a set of entries, by replacing parts of the DN or even the whole DN with POSIX extended regular expressions matching the entries for which to receive notifications.

When the whole DN (including the DE DN part) is used to configure the `dn` attribute, POSIX extended regular expressions can be also used to replace exact subscribers IDs (or any other part of a DN).

All `dn` fields of all `CudbNotificationObjectClass` instances of the `monitor` type must contain the same value under the same instance of `CudbNotificationEvent`, so that all the monitored attributes refer to the same LDAP entry or set of LDAP entries.

See [Configuration Examples](#) on page 14 for specific examples about how to configure `CudbNotificationObjectClass` instances of the `monitor` type to monitor specific entries or a set of them.

One or several instances of class `CudbNotificationsAttr` can be created under each `CudbNotificationObjectClass` instance of the `monitor` type. All of them contain the attributes to be monitored within the object class. If any of them changes, the notification is subject to be sent, except when the attribute is single-valued and its object class is added or deleted in the same LDAP operation that changes the attribute value.

- **monitorAll**: This value has the same meaning and use as the `monitor` type, but it also triggers the sending notification when the attribute is single-valued and its object class is added or deleted in the same LDAP operation that changes the attribute value. Therefore, LDAP attributes contained in `CudbNotificationObjectClass` instances of the `monitorAll` type are monitored in the notification without exceptions. It monitors the creation or deletion of entries, not only the attributes. All `dn` fields of all `CudbNotificationObjectClass` instances of the `monitorAll` type must contain the same value under the same instance of `CudbNotificationEvent`. For more information, refer to [CUDB Node Configuration Data Model Description](#).
- **check**: LDAP attributes in `CudbNotificationObjectClass` instances of the `check` type are checked to have the specified value to send the



notification. The dn attribute must contain the DN of the entry without the DE DN part and the object class of these LDAP attributes.

One or several instances of class `CudbNotificationsAttr` can be created under each `CudbNotificationObjectClass` instance of the check type. All of them contain the attributes to be checked, and the value for the comparison.

If any of the check conditions is fulfilled the notification is sent. A condition is not fulfilled if the attribute to be checked has no-value.

- `related`: LDAP attributes contained in `CudbNotificationObjectClass` instances of the `related` type are included in the notification. The dn attribute must contain the DN of the entry without the DE DN part and the object class of these LDAP attributes.

One or several instances of class `CudbNotificationsAttr` can be created under each `CudbNotificationObjectClass` instance of the `related` type. Each instance contains the attributes to be sent in the notification.

In object classes of check and `related` type, regular expressions cannot be used. When there is a need to check or include attributes that belong to the same entry as the monitored one, regular expressions should not be used and there must be as many configured notification events as there are different entries for monitored object classes.

The attributes in the [CUDB Node Configuration Data Model Description](#) must be encoded as follows:

- Character strings for non-binary attributes.
- Base64-encoded strings for binary attributes.

At least one instance of the `CudbNotificationObjectClass`, `monitor`, or `monitorAll` types must be defined per notification. But it is possible not to define the check and `related` instance types. The CUDB system does not check at configuration time if the configured attributes belong to one of the supported types (see [Dependencies and Interactions](#) on page 7).

The `send` field of the defined instances of `CudbNotificationAttr` acts as a sending flag for the related attribute. If it is true, the attribute name and value are sent in the notification. For multi-valued attributes, the complete set of values is sent. In case the attributes are included in the `CudbNotificationObjectClass` instances of the `monitor` or `monitorAll` types, the old attribute values are also sent.

If any of the LDAP attributes and object classes in the notification (either `monitor`, `monitorAll`, `checked` or `included` attributes) can not be retrieved from the database, the notification is not sent, and an error is reported.



For more information about these parameters, and generic SAF configuration model parameter handling directives, refer to [CUDB Node Configuration Data Model Description](#).

3.2.3 Configuration Examples

This section provides different examples of configuring the dn attribute in `monitor` or `monitorAll` type `CudbNotificationObjectClass` classes.

Example 1 Configuring dn without DE DN Part

In [Example 1](#), a notification will be triggered for any LDAP entry whose complete DN contains the configured value.

```
dn="EpsDynInfId=EpsDynInf,EpsStaInfId=EpsStaInf,serv=eps"
```

Example 2 Using Regular Expression to Replace a Subscriber ID in DN of LDAP Entry

In [Example 2](#), a notification will be triggered for any LDAP entry that matches the configured value, for example, for any `mscId` value.

```
dn="EpsDynInfId=EpsDynInf,EpsStaInfId=EpsStaInf,serv=eps,mscId=.*,ou=multiscs,dc=operator,dc=com"
```

Example 3 Using Regular Expressions to Configure dn without DE DN Part

In [Example 3](#), a notification will be triggered for any value of `grpId`.

```
dn="grpId=.*,serv=pcrf.*"
```

If this configuration must be restricted to only a few values of the `grpId`, use the OR operator: [Example 4](#).

Example 4 Using Regular Expressions with Logical Operators, First Method

```
dn="(grpId=id1|grpId=id2),serv=pcrf.*"
```

Or, to accomplish the same: [Example 5](#)

Example 5 Using Regular Expressions with Logical Operators, Second Method

```
dn="grpId=(id1|id2),serv=pcrf.*"
```

Example 6 Using Multiple Wildcards in dn Attribute

```
dn="recordId=.*,viewId=.*,fqdn=.*,ou=EnumDn.*,serv=enum,ou=servCommonData,dc=operator,dc=com"
```

Note: Any regular expression can be used to define the DN pattern. However, complex regular expressions should be used with caution to avoid unexpected notifications.



3.2.4 Preconfigured Notifications

An Evolved Packet Core (EPC) Mobility Notification is preconfigured in CUDB. It sends notifications to the Home Location Register (HLR) FE. EPC mobility support is an optional CUDB feature, as detailed in [CUDB Technical Product Description](#).

The following attributes are involved in the preconfigured notification:

- Monitored: EpsMobilityNotifInfo
- Checked: PSL0C, which must be equal to SGSN Known or MS Purged in SGSN.
- Related:
 - International Mobile Subscriber Identity (IMSI).
 - Serving GPRS Support Node (SGSN) node where the subscriber is located.
 - Collision Detection Counter (CDC).

3.2.5 External Networks Connectivity

As notifications are sent outside the CUDB, it is needed to configure the network connectivity facilities in the system for communication towards the endpoints. For more information, refer to [CUDB Node Network Description](#).

Additionally, eVIP routes for the endpoints must be added in the blades or Virtual Machines (VMs) where the notification processes are running, if they are not already established. Otherwise, the endpoints cannot be accessed. For more information, refer to [CUDB System Administrator Guide](#).

It is also possible to configure how TCP sockets towards the endpoints are initialized per notification. For more information about configuring socket initialization, refer to [CUDB SOAP Interwork Description](#).

3.3 Fault Management

This feature triggers the following alarms:

- SOAP Notifications, Discarded Notifications, refer to [SOAP Notifications, Discarded Notifications](#).
- SOAP Notifications, Endpoint Unreachable, refer to [SOAP Notifications, Endpoint Unreachable](#).



3.4 Performance Management

For more information on counters managed by the notifications module, refer to [CUDB Counters List](#).

3.5 Security

It is possible to configure a TLS key and certification in the SOAP interface. For detailed information about security related parameters and security configuration, refer to [CUDB Security and Privacy Management](#).

3.6 Logging

For more information on the log messages generated by the notifications module, refer to [CUDB Node Logging Events](#).



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 SOAP Interwork Description
2. CUDB Security and Privacy Management
3. CUDB High Availability
4. CUDB Application Integration Guide
5. CUDB Troubleshooting Guide
6. CUDB Node Configuration Data Model Description
7. CUDB Technical Product Description
8. CUDB Node Network Description
9. CUDB System Administrator Guide
10. SOAP Notifications, Discarded Notifications
11. SOAP Notifications, Endpoint Unreachable
12. CUDB Counters List
13. CUDB Node Logging Events
14. CUDB Glossary of Terms and Acronyms