

Integration in User Data Consolidation

Ericsson Service-Aware Policy Controller

USER GUIDE

Copyright

© Ericsson España, S.A. 2017, 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	Integration in User Data Consolidation Introduction	1
2	Integration in User Data Consolidation Solution Description	3
2.1	Solution Overview	3
2.2	SAPC Data Model for UDC	4
2.2.1	SAPC LDAP Directory Information Tree	4
2.2.2	SAPC Entries	6
2.3	Connection Handling	11
2.3.1	Clean up of Connections	12
2.4	CUDB Geographical Redundancy (1+1+1) Support	12
2.4.1	Failover	13
2.4.1.1	CUDB Failure Detection	14
2.4.1.1.1	Error Handling with CUDB	14
2.4.1.1.2	LDAP Operations Time-Out	16
2.4.1.2	Failover Execution	19
2.4.2	Failback	19
2.4.3	Connectivity Lost	22
2.5	Failure in LDAP Operations	23
2.6	Overload Protection on IP BB Delayed Responses	24
2.7	CUDB Overload	24
3	Interfaces Description	25
3.1	LDAP Interface	26
3.1.1	LDAP Messages between the SAPC and CUDB	26
3.1.2	LDAP between Provisioning System and CUDB	26
3.2	SOAP Interface	26
4	Operation and Maintenance	29
4.1	Configure Database Access in the SAPC	29
4.1.1	Subscriber	29
4.1.2	Detailed Values of Entity Data Sources to access data from CUDB	30
4.1.2.1	External Repository	30
4.1.2.2	Subscriber	31
4.1.2.3	Subscriber and SubscriberIdentity when Subscribers are provisioned with several Identifiers	37
4.1.2.4	Emergency Subscriber	39
4.2	Configure SOAP Notifications in CUDB	40
4.3	Subscriber Data Extension in CUDB	40
4.4	LDAP Operations for Provisioning System	41



4.4.1	Creation of Entries	41
4.4.2	Deletion of Entries	43
4.4.3	Addition/Removal of Traffic Identifiers	43
4.5	Alarms	43
4.6	Logging	43
5	Capabilities	45
	Reference List	47



1 Integration in User Data Consolidation Introduction

This document contains the information required to integrate the SAPC application into Ericsson UDC solution. The document can be also used as reference solution for other LDAP external repositories different to CUDB.

It provides the Ericsson reference DIT to access the SAPC subscriber data in CUDB.





2 Integration in User Data Consolidation Solution Description

2.1 Solution Overview

One of the use cases for the SAPC related to external database is to access to a common User Data Repository (UDR) where the different applications consolidate their user data (single instance of common data).

UDC is the solution based on Data Layered Architecture (DLA) defined in 3GPP (see User Data Convergence (UDC)). DLA implies the separation of application logic from user data storage, simplifying the network topologies and providing more efficient use of network resources. DLA enables independent scalability of storage and processing capacity. Network applications are implemented as data-less applications that access user data stored in a centralized and efficient Back-End (BE) database.

Ericsson UDC solution complies with 3GPP UDC. In Ericsson UDC, the back end database is implemented by the Centralized User Database (CUDB). For more information about Ericsson UDC solution, see User Data Consolidation (UDC) , Technical Product Description.

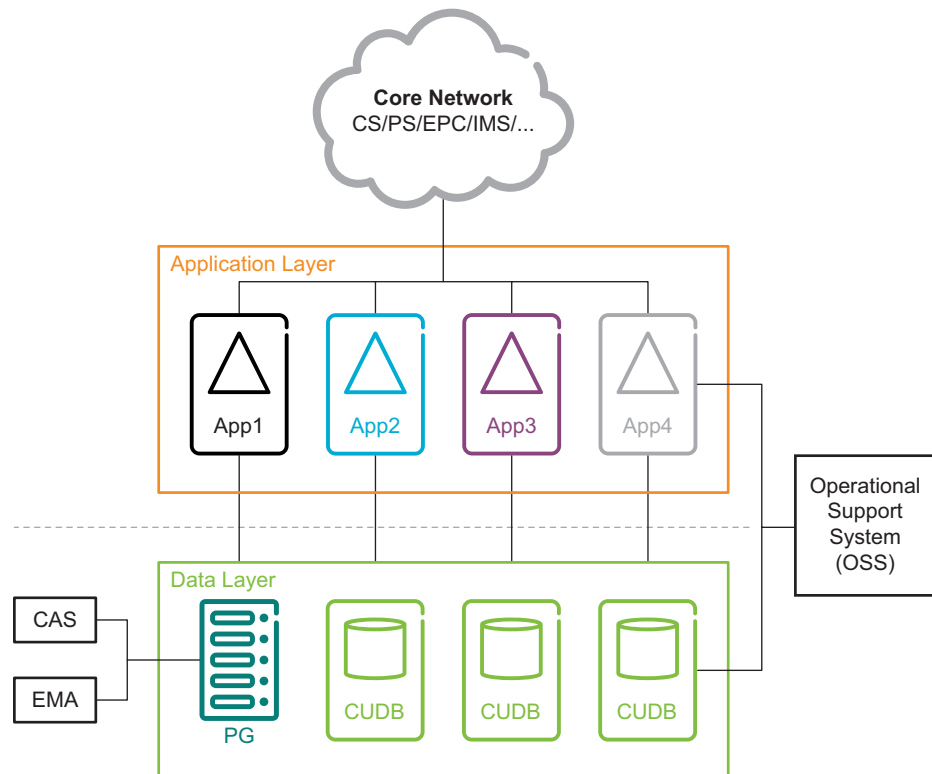


Figure 1 Ericsson UDC Reference Architecture



Inside this architecture, the SAPC is an application that stores data in CUDB.

2.2 SAPC Data Model for UDC

The SAPC handles an internal data model when the application data is stored in its internal database. This data model is described in [Provisioning REST API](#).

In UDC, CUDB is in charge of storing the application data. CUDB is built as an LDAP directory server, containing the needed entries and attributes according to the schema defined for the different applications. Data is accessible by LDAP, and so the data is modeled according to LDAP specification. From an external LDAP client, data is displayed as a tree-like structured, named DIT. Each entry in the DIT is identified by a name, Relative Distinguished Name (RDN), being the path from the tree root the so-called DN.

When integrating with UDC solution, the SAPC stores **subscriber** data in CUDB. Other data, like Services, Groups or configuration data are kept stored in the SAPC. In the next chapters, it is explained the reference data model for the SAPC subscriber data stored in CUDB.

2.2.1 SAPC LDAP Directory Information Tree

CUDB offers an LDAP Directory Information Tree (DIT) that can be extended to accommodate the data of the different applications. For more information about the meaning of each entry, see CUDB LDAP Interwork Description. Figure 2 shows how CUDB DIT is extended to host the SAPC application data.

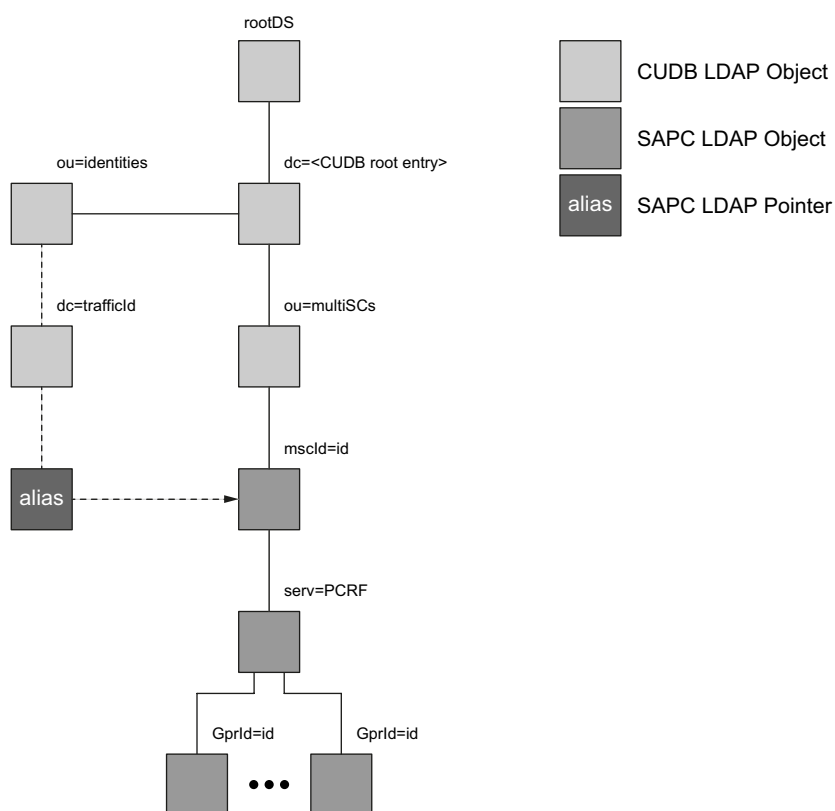


Figure 2 The SAPC Directory Information Tree

The SAPC subscriber data is placed under `mscId = <multiservice consumer identifier>`. Under this level, the SAPC subscriber data is placed under `serv=PCRF`.

The `ou=identities` branch contains an alias to the corresponding `ou=multiSCs` branch entry. The Distinguished Name (DN) of the associated subscriber identifier entry is obtained by dereferencing the alias.

The SAPC can access subscriber data with two keys:

- **mscId**, that corresponds to the administrative subscriber identity in the SAPC. It is used when the value of the `mscId` is the same than the value of the subscriber identifier received within the traffic request. For example, if the SAPC receives an IMSI in the traffic request, the value of the `mscId` for that subscriber is the IMSI value. Besides, the SAPC uses this key when reading the subscriber data after receiving a notification from CUDB. In both cases, the SAPC uses the following DN to get access to the SAPC subscriber data:

`serv=PCRF, mscId=<mscId>,ou=multiSCs,<CUDB root entry>`
- **trafficId**, that corresponds to the identifier received by the SAPC in the traffic query. It is used when the identifier included in the traffic request can be different to the `mscId`. An example is the case when there is Fixed Mobile



Convergence scenario, where the subscriber profile both for fixed and mobile is hosted under the same mscId entry. The traffic request can be received with an IMSI in the mobile scenario or a NAI in the fixed one.

In this case, the SAPC does an extra query to obtain the administrative subscriber identity with the traffic identifier as key. Then the SAPC retrieves the subscriber profile using the administrative identity as key. The DN for the extra query is:

```
TRAFFICID=<trafficIdNumber>,dc=trafficId,ou=identities,<CUDB  
root entry>
```

Note: The SAPC supports to convert an IP address to an IMSI list. For more information, refer to [Session Context Exposure User Guide](#).

2.2.2

SAPC Entries

This chapter specifies the entries for the SAPC subscriber data. AttributeType and object classes for the SAPC are defined in SAPC LDAP Schema for UDC.

The SAPC data stored in CUDB complies with the following **Conventions**:

- All attributes are of type Directory String.
- Almost all attributes are read-only, except **Accum**, **OngoingSession** and **ClosedSession** (read and write).
- All attributes are optional.

The following tables specify the object classes and their attributes for the SAPC subscriber data. These tables contain the following information:

Table 1 The SAPC Subscriber Data

object class name		
Name of the object class.		
Attribute	Corresponding attribute in SAPC internal data model	Remarks
Attribute name. "*" indicates key attributes, that is attributes being part of the DN.	Name of the corresponding attribute in the internal repository data model specified in Provisioning REST API or Analytics REST API. Refer to this document for attribute description and examples.	This column shows exceptions to the general conventions (see Page 6) and differences with the explanations in the Provisioning REST API or Analytics REST API. It also includes examples to clarify some attributes.



The OID is used as unique identifier for attributes and object classes. The OIDs for the object classes and attributes of the SAPC when the data are stored in CUDB are the following ones:

Object Classes OID

1.3.6.1.4.1.193.202.1.(x)

Attributes OID

1.3.6.1.4.1.193.202.2.(y)

Next tables cover the object classes that contain the SAPC subscriber data attributes.

Table 2 SAPC Object Class

objectClass SAPC		
Attribute	Corresponding Attribute in SAPC Internal Data Model (Referred to Provisioning REST API)	Remarks
*serv This attribute identifies the subscriber entry for the SAPC data. Defined in CUDB, see CUDB LDAP Interwork Description	-	Type: IA5 String Value range 1–32 characters Value = PCRF (forced value) Required: mandatory
SSubId	subscriberId	Required: mandatory Attribute size: 32 characters Single-value
TrafId	trafficIds	Multiple-value
WIServ	subscribedContents	Multiple-value
BIServ	deniedContents	Multiple-value
FamId	sharedDataplan	Attribute size: 32 characters Single-value
GrpId	dataplan	Multiple-value
Sms	smsDestinations	Multiple-value Example: +34600123456
SOpInf	operatorSpecificInfos	Multiple-value



Table 2 SAPC Object Class

objectClass SAPC		
Attribute	Corresponding Attribute in SAPC Internal Data Model (Referred to Provisioning REST API)	Remarks
Accum	usageAccumulators	Visibility: read-write. Provisioning system must not modify this attribute. Attribute type: octet string Single-value
CharS	onlineChargingSystemProfileId	Single-value Example: OcsWestMadrid
MpsP	mpsProfileId	Single-value Example: MpsProfileEpsBearerPri o2
SevTrig	eventTriggers	Multiple-value Attribute type: integer (positive numbers only) Example: 48
SspId	spid	Single-value Attribute type: integer Example: 145
SpdnGwName	pdnGwListName	Single-value Example: pdngListEricsson
SpresenceAreaName	presenceReportingAreaNames	Single-value Example: Butarque



Table 3 SAPC1 Object Class

objectClass SAPC1		
Attribute	Corresponding Attribute in SAPC Internal Data Model (Referred to Provisioning REST API)	Remarks
Slimit	usageLimits	Single-value Attribute size: 2680 characters When creating Reporting Groups in this attribute, subscriptionDate + resetPeriod must represent a date in the future.

Table 4 SAPC2 Object Class

objectClass SAPC2		
Attribute	Corresponding Attribute in SAPC Internal Data Model (Referred to Provisioning REST API)	Remarks
ContId	contentFiltering	Single-value Example: 222
CustId	customerId	Single-value Example: KKJ876JHL98
CharP	subscriberChargingProfileId	Single-value Example: SubsChargingProfile1
MaxBQosP	maxBearerQosProfileId	Single-value Example: QosProfileHigh
MinBQosP	minBearerQosProfileId	Single-value Example: QosProfileLow
SrvToRed	subscribedContents	Single-value Example: Service1, Service2

SAPC3 object class contains the data to associate a group (service offering) to a SAPC subscriber.

Note: The subscriber group profiles in itself are provisioned in the SAPC.



Table 5 SAPC3 Object Class

objectClass SAPC3		
Attribute	Corresponding Attribute in SAPC Internal Data Model (Referred to Provisioning REST API)	Remarks
* GrpId This attribute identifies the group to be associated to a subscriber or a family.	dataplanName	Required: mandatory Example: gold
Gprio	priority	Single-value Attribute type: integer Example: 1
StartD	startDate	Single-value Example: 01-01-2025T22:00:00
EndD	stopDate	Single-value Example: 02-01-2025
Duration	durations	Multiple-value Example: 01-09-2020, 30-09-2020

SAPC5 object class contains the elements of ongoing and closed session contexts. For more information about the session context collection, see [Session Context Exposure User Guide](#).



Table 6 SAPC5 Object Class

objectClass SAPC5		
Attribute	Corresponding Attribute in SAPC Internal Data Model (Referred to Analytics REST API)	Remarks
OngoingSession	ongoingSession	Visibility: read-write. Provisioning system must not modify this attribute. Attribute type: octet string Single-value
ClosedSession	closedSession	Visibility: read-write. Provisioning system must not modify this attribute. Attribute type: octet string Single-value

2.3 Connection Handling

The SAPC connects only to CUDB through the CUDB-FE VIP addresses, used to send and receive LDAP traffic.

The SAPC establishes a pool of LDAP sessions (TCP connections) towards the active CUDB from each traffic processor.

The SAPC establishes the LDAP sessions when receives traffic from the network (for example diameter queries) until the maximum number is reached. At that point, the SAPC does not open more LDAP sessions but uses the already established.

The SAPC handles the LDAP sessions in an asynchronous mode, resulting in several LDAP queries sent in parallel through the same LDAP session.

Figure 3 shows the flow for successful connection establishment when the SAPC receives a traffic request.

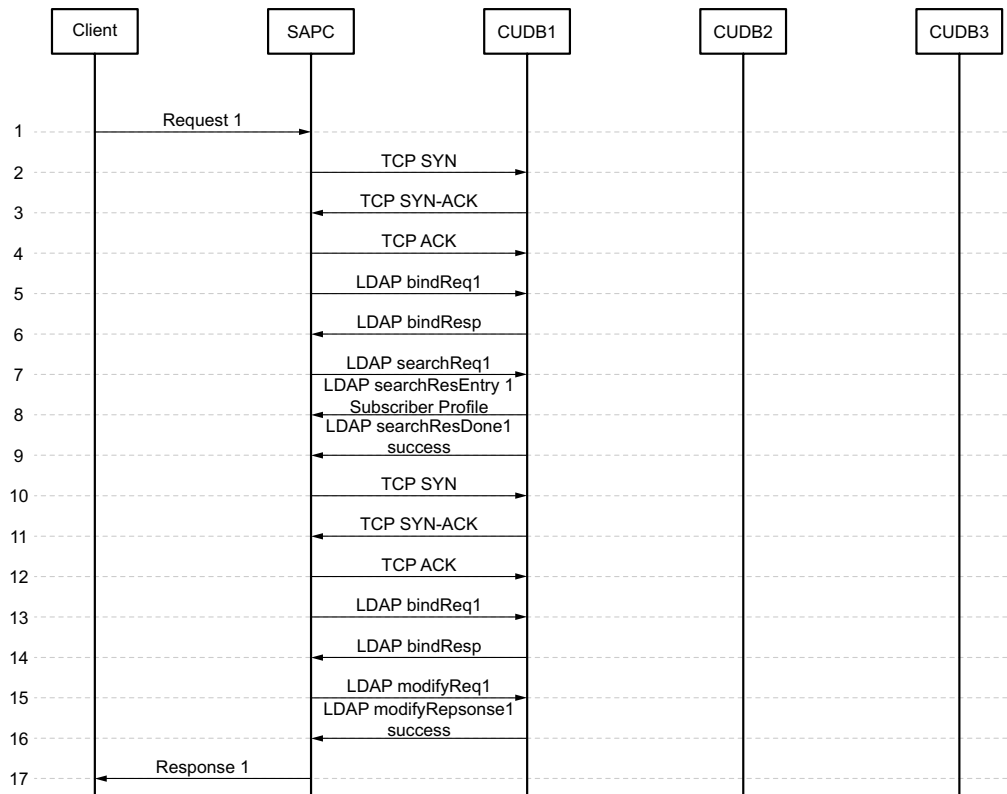


Figure 3 LDAP sessions establishment upon traffic request

2.3.1 Clean up of Connections

The SAPC handles an inactivity timer to close the LDAP sessions from stand-by CUDB VIP addresses that are not used during a period.

The SAPC computes the inactivity timer when traffic is served in the traffic processor. This means that the SAPC closes the connections from stand-by CUDB VIP addresses that do not serve any traffic, while the active one is serving traffic.

The connections from active CUDB, are not closed even if they are not in use. This saves time avoiding new connection establishments when traffic is served again.

When an VIP address of the CUDB is removed from the SAPC configuration, the SAPC removes the TCP connections established towards the removed VIP address when the inactivity timer expires.

2.4 CUDB Geographical Redundancy (1+1+1) Support

CUDB provides a redundancy configuration where the data can be replicated in three geographically distributed sites (1+1+1 geographical redundancy). In this configuration, the SAPC allows to define up to three different points of access



towards the CUDB system, each one towards different site. Each point of access is identified by a different VIP address.

The SAPC supports this for the LDAP interface. The operator can configure in the SAPC a pool of three VIP addresses in priority order to identify the CUDB sites that the SAPC can connect to. Only one VIP address is considered as active at a time, while the rest is stand by. The active VIP address is the VIP address with highest priority that is reachable and available to serve traffic.

Figure 4 shows the scenario in which the SAPC uses 1+1+1 redundancy in CUDB.

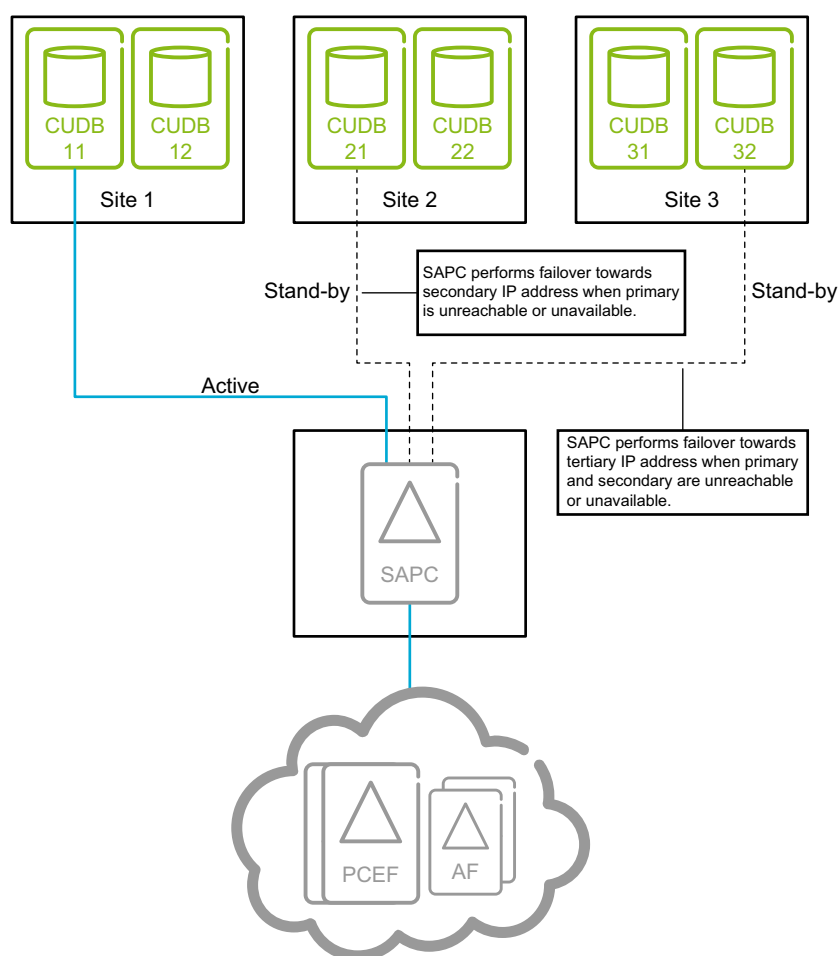


Figure 4 The SAPC and CUDB 1+1+1 geographical redundancy

2.4.1 Failover

Failover is the mechanism that allows the SAPC to route the LDAP traffic towards a lower priority VIP address. The failover takes place when the active VIP address is unreachable or unavailable to serve traffic.

The SAPC always tries to use as active VIP address the highest priority VIP address that is available. When the active VIP address experiences a failure, a failover is performed towards the next VIP address in priority order. The SAPC raises an alarm reporting the faulty VIP address.

The SAPC controls and executes the failover per processor and per pool of LDAP sessions.

2.4.1.1 CUDB Failure Detection

The SAPC detects that a CUDB is down, when receives specific errors from CUDB or when does not receive any answer (LDAP operations time-out). The failover criteria is reached when the number of consecutive LDAP failed queries is higher than the 30% of the pool size of LDAP connections. When the CUDB is down, the SAPC does the following actions:

- raises an alarm reporting the failing CUDB
- closes the LDAP sessions towards the failing CUDB
- and executes failover (see details in Section 2.4.1.2 on page 18).

If an LDAP query reaches the maximum number of reattempts before the failover criteria is fulfilled (the CUDB is not considered down yet), the SAPC considers that the LDAP operation has failed. See Section 2.5 Failure in LDAP Operations on page 23.

2.4.1.1.1 Error Handling with CUDB

The SAPC considers the following errors received from CUDB:

- An LDAP Error Code 52 as response to an LDAP query. CUDB reports with this error code that is unavailable to serve traffic. The SAPC executes failover immediately.
- A TCP_RST message to terminate the TCP connections..

Figure 5 shows the case of Error Code 52.

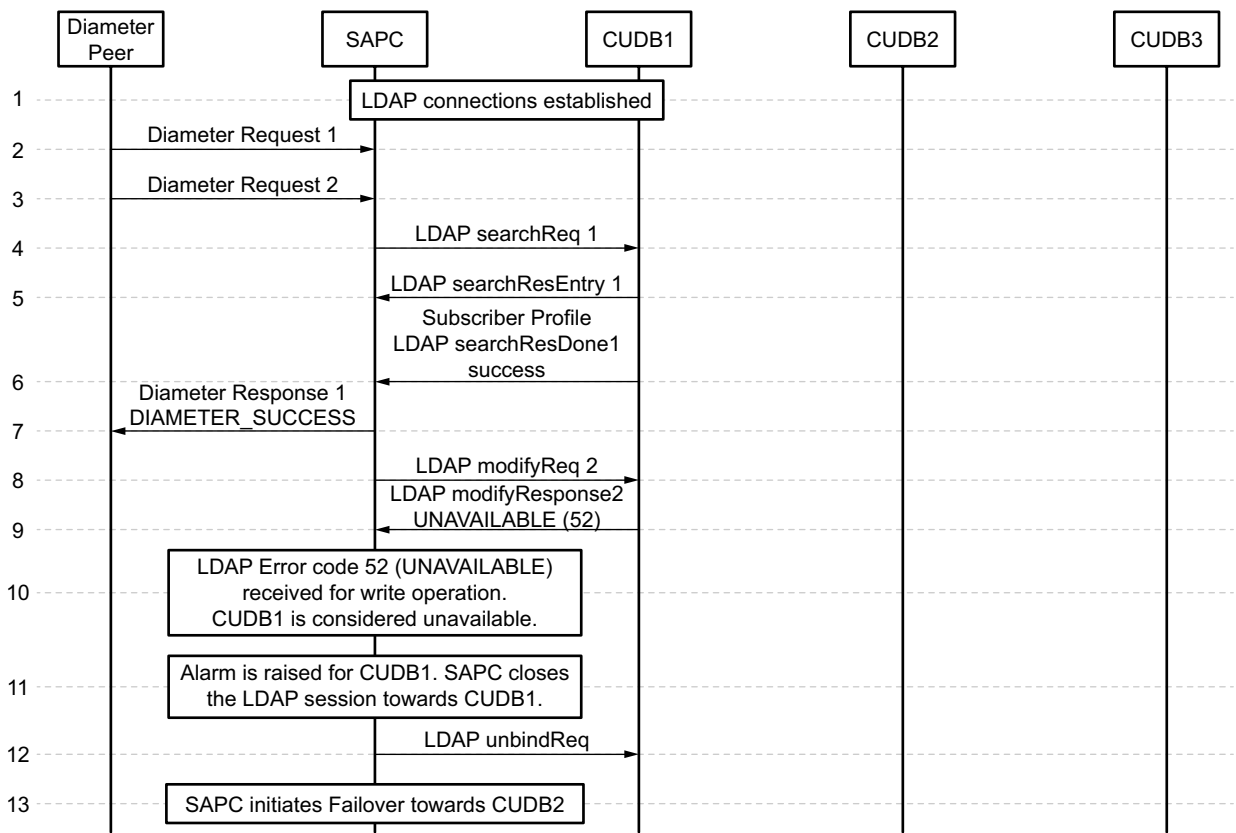


Figure 5 The SAPC detects CUDB failure at reception of LDAP error code 52

Figure 6 shows the case for TCP_RST.

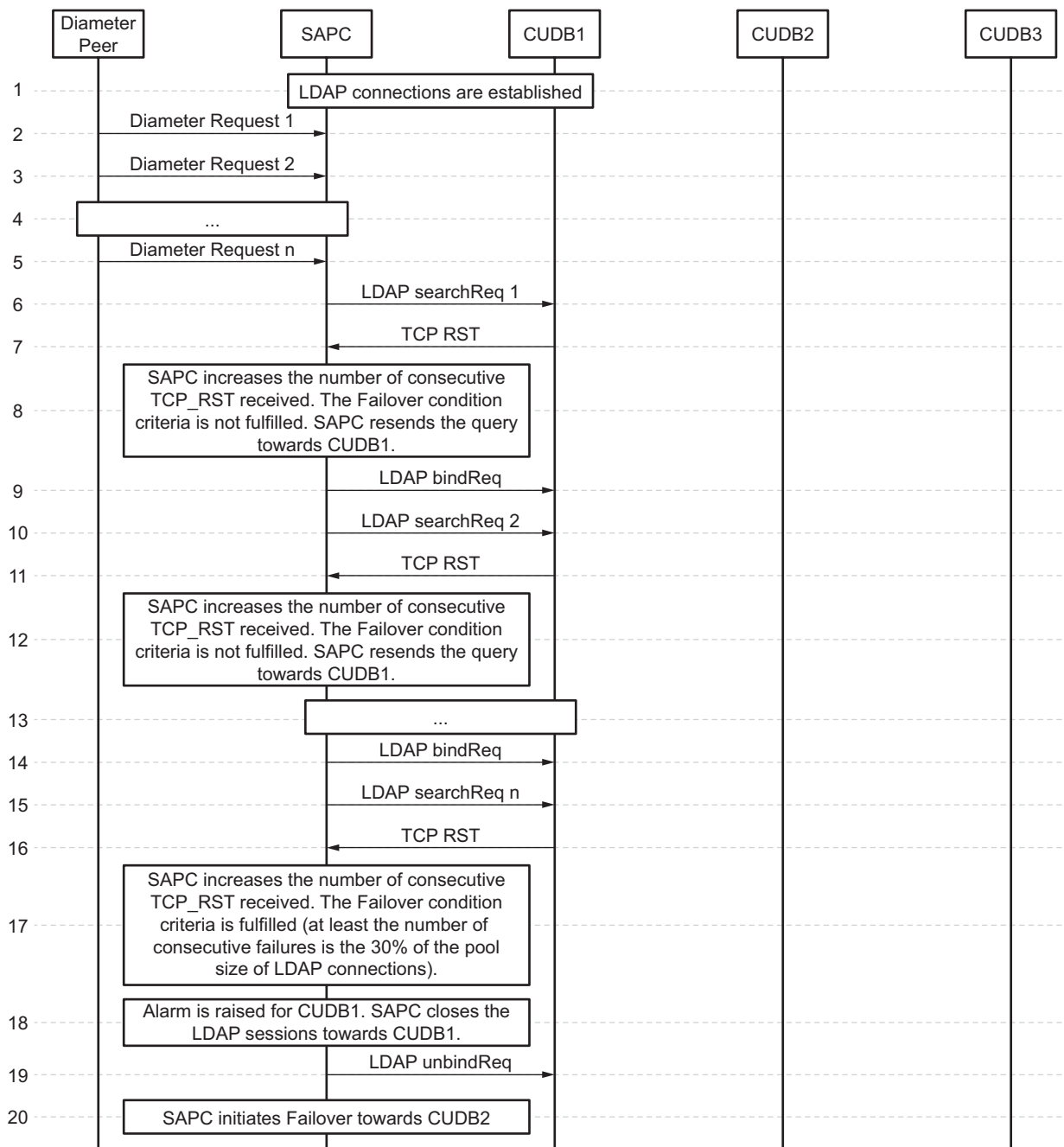


Figure 6 The SAPC detects CUDB failure on TCP RST and failover criteria is fulfilled

2.4.1.1.2 LDAP Operations Time-Out

The SAPC monitors the LDAP queries sent to the CUDB. When a time-out occurs in one LDAP session, the SAPC reattempts the LDAP query (according to the number of configured query reattempts), towards the same CUDB.



This situation can be provoked by a communication link failure between the SAPC and CUDB, or by other failure in CUDB.

Figure 7 shows how the SAPC detects LDAP query time-out.

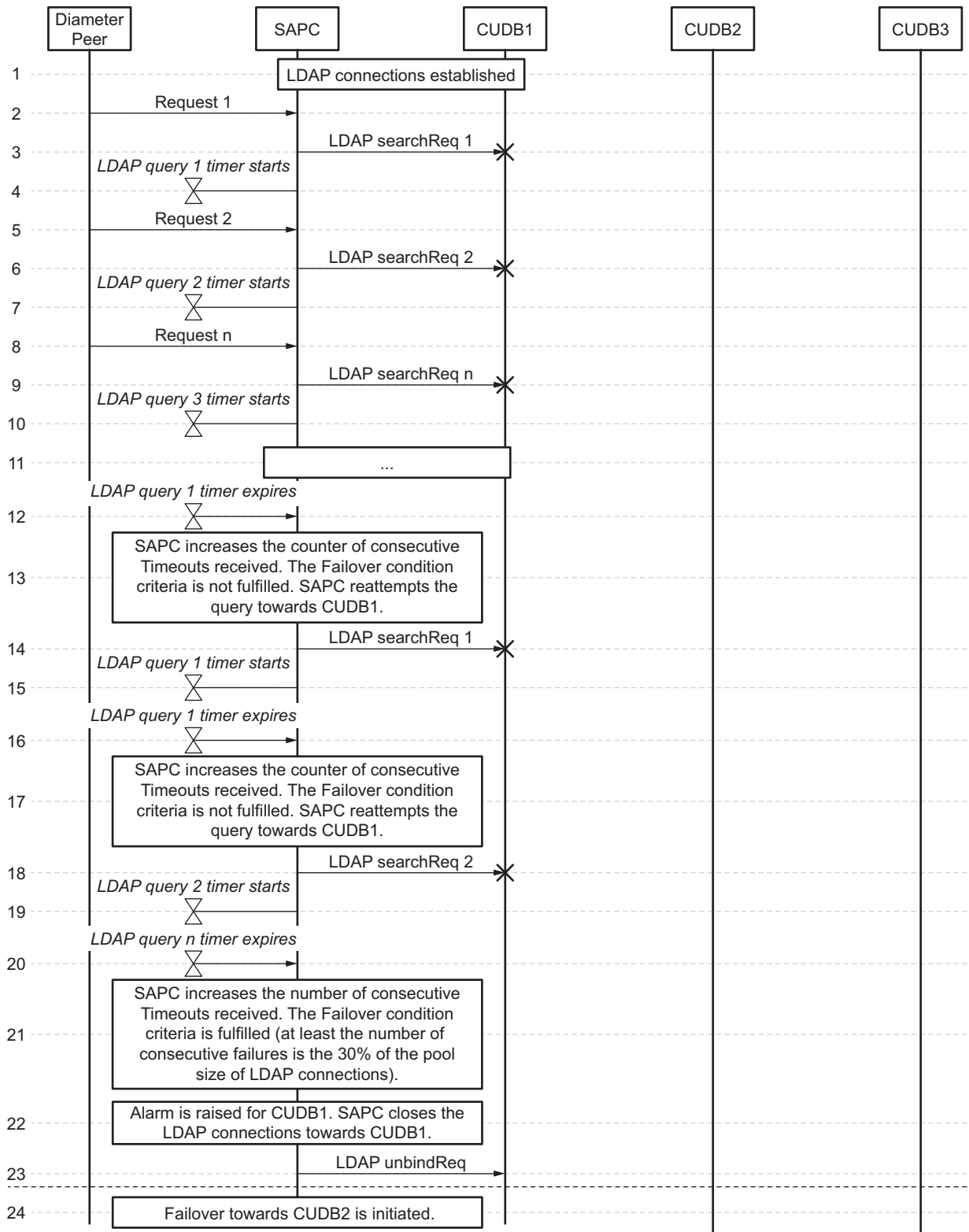


Figure 7 The SAPC detects CUIDB failure when no answer is received and failover criteria is fulfilled



2.4.1.2 Failover Execution

The SAPC initiates the failover when detects that the CUDB is down in the situations explained in Section 2.4.1.1 CUDB Failure Detection on page 14. During the failover, the SAPC establishes LDAP sessions to the next priority CUDB as explained in Section 2.3 on page 11. The SAPC does not lose any traffic during the failover. The queries that do not reach the maximum number of reattempts before the failover criteria is fulfilled, are retried to the new CUDB. The queries that reach the maximum number of reattempts before the failover criteria is fulfilled, are considered as failed.

When the SAPC executes the failover, it initiates a timer to perform failback when any VIP address with higher priority is available as explained in Section 2.4.2 Failback on page 19.

Figure 8 shows the failover process in the SAPC.

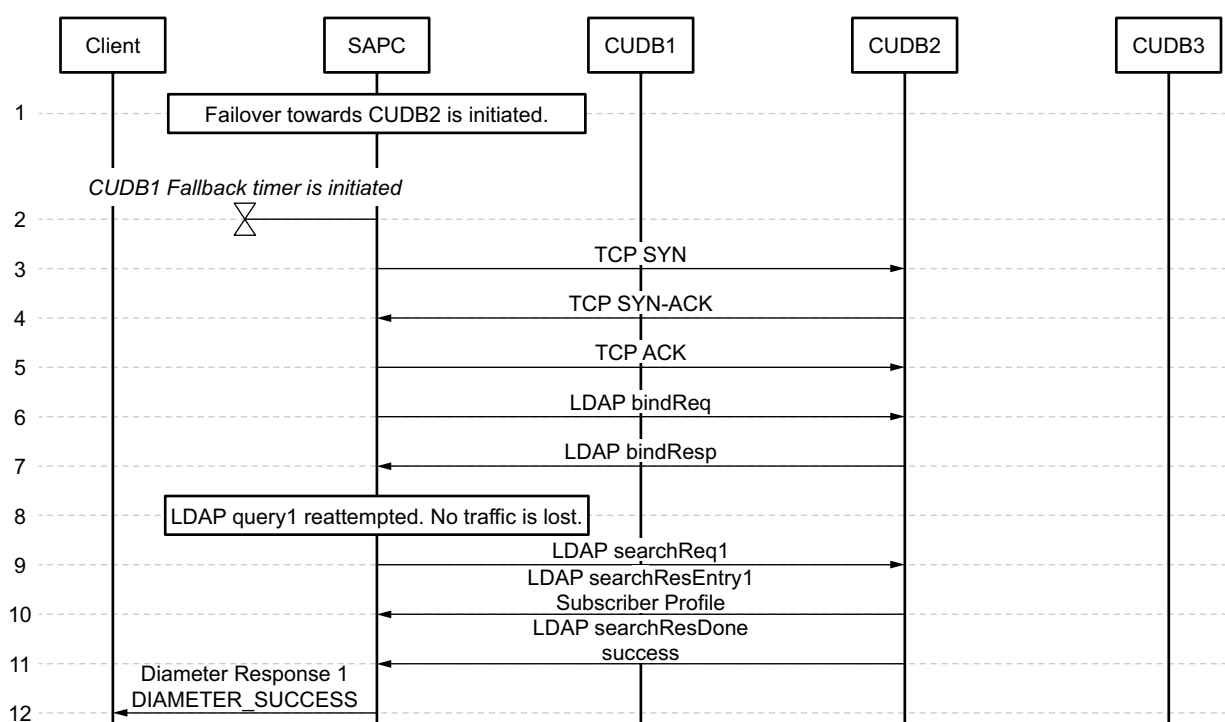


Figure 8 Failover to next priority CUDB.

2.4.2 Failback

After a failover, the SAPC monitors the state of the VIP addresses with higher priority than the active VIP address. To detect that an VIP address with higher priority is available, the SAPC triggers the failback procedure. To do that, the SAPC uses a failback timer, and performs the following actions when the timer expires:



- tries to establish an LDAP session and sends an LDAP test query towards the higher priority VIP address. If the LDAP operation succeeds, the SAPC marks that VIP address as the new active.
- initiates connection towards the new active VIP Address (connection establishment is detailed in Section 2.3 on page 11).
- ceases the alarm corresponding to the new active VIP.
- closes the LDAP Sessions towards the old active VIP address, once answered all the pending LDAP queries.
- ceases the alarms corresponding to lower priority VIPs, if any.

Note: The SAPC executes the failback per processor.

No traffic is lost during the failback procedure.

Figure 9 shows a failback when the SAPC is connected to the secondary VIP address.

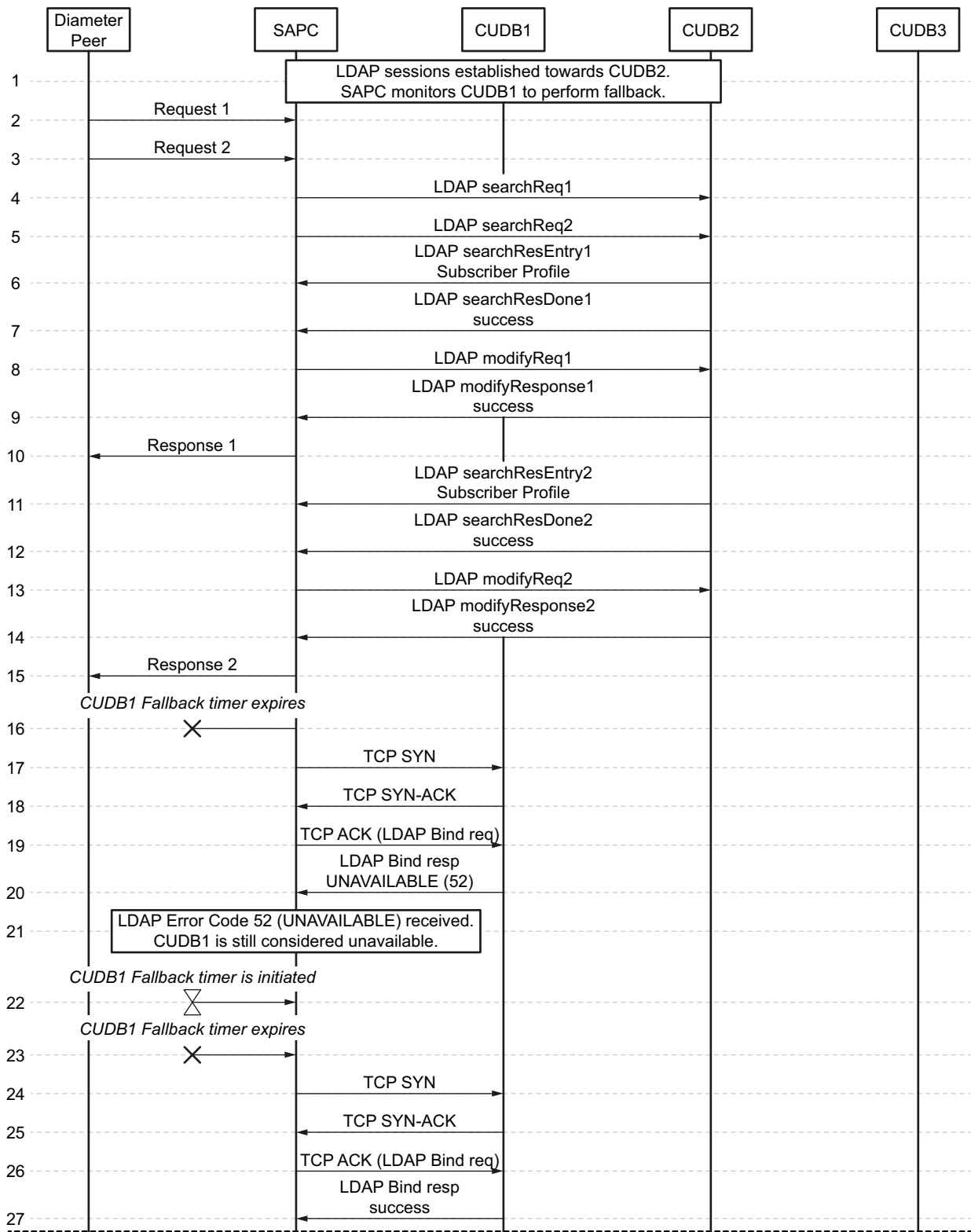


Figure 9 Fallback part 1

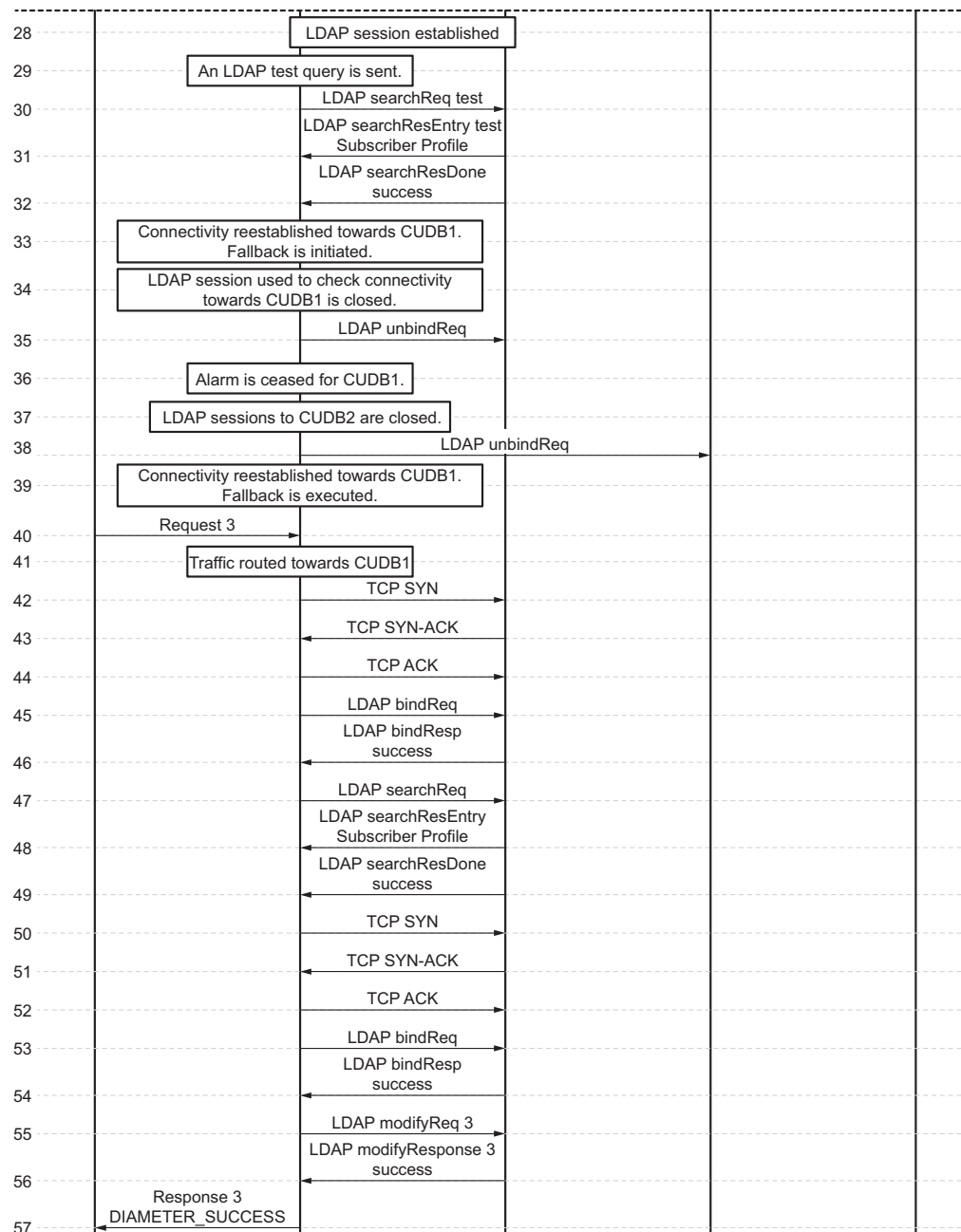


Figure 10 Failback part 2

2.4.3

Connectivity Lost

When all the configured VIP addresses are unavailable or unreachable, there is no active VIP address in the SAPC. In this situation, the SAPC considers that the LDAP operations have failed and does not serve any traffic unless the special unknown subscriber is provisioned in the SAPC as explained in [Database Access](#).



The SAPC monitors the status of all the VIP addresses to reestablish connectivity as soon as possible. When an LDAP session is successfully established towards any VIP address, the SAPC performs the following actions:

- considers that VIP address as active and processes traffic again.
- ceases the alarm of the active VIP address and of the VIP addresses with lower priority, if any.
- If active VIP address is other than the primary, initiates failback.

2.5 Failure in LDAP Operations

When an LDAP operation result is not successful, the SAPC handles the traffic depending on the failure. Next table shows all the failures covered and the SAPC actions.

Table 7 LDAP Failures

Failure	SAPC Action		
	Reattempt ⁽¹⁾ Operation	Failover criteria	Diameter Result-Code ⁽²⁾⁽³⁾
Operation Time-Out	Yes	Yes ⁽⁴⁾	DIAMETER_UNABLE_TO_C OMPLY DIAMETER_SUCCESS
TCP_RST	Yes	Yes ⁽⁴⁾	DIAMETER_UNABLE_TO_C OMPLY DIAMETER_SUCCESS
Error code 32 (noSuchObject)	No	No	DIAMETER_USER_UNKNO WN DIAMETER_SUCCESS
Error code 51(busy)	No	No	DIAMETER_TOO_BUSY DIAMETER_SUCCESS
Error code 52 (unavailable)	Yes	Immediate ⁽⁵⁾	DIAMETER_UNABLE_TO_C OMPLY DIAMETER_SUCCESS
Any other Error code	No	No	DIAMETER_UNABLE_TO_C OMPLY DIAMETER_SUCCESS

(1) The maximum number of reattempts is 2.

(2) If an LDAP query fails when the SAPC is processing a Gx CCR-T, the SAPC removes the session and answers with successful result code.

(3) When an LDAP search operation fails, the SAPC can use the especial "unknown" subscriber and answers with successful result code. The unknown subscriber must be provisioned in the SAPC as explained in **Database Access**

(4) The failover criteria consists of reaching at least the 30% of maximum LDAP connections on consecutive failures. It applies by processor independently.

(5) The error code 52 makes to fulfill the failover criteria immediately



2.6 Overload Protection on IP BB Delayed Responses

In case of congestion in the IP BB, the delay and or packet loss rate in the IP BB that connects the SAPC and CUDB can exceed the required values. So, the capacity in the SAPC can be dismissed owing to the throughput that the SAPC can obtain for the LDAP traffic is lower than the required to sustain the SAPC dimensioned capacity. During congestion, the response time for the LDAP operations is increased and the SAPC handles the received traffic until a limit determined by the maximum number of queries handled per LDAP connection. When this limit is exceeded, the SAPC considers the LDAP operations as failing queries and returns the corresponding error as explained in chapter Section 2.5 Failure in LDAP Operations on page 23. The SAPC follows same behavior if the response time is increased owing to congestion in the IP BB or in the CUDB.

2.7 CUDB Overload

When CUDB system is overloaded, CUDB rejects part of the LDAP traffic with LDAP error code busy (51). There are other overload situations in which CUDB responds with the LDAP error code other (80).

When CUDB rejects the SAPC LDAP traffic, the SAPC does not reattempt the LDAP queries and consider them as faulty. Therefore, the SAPC rejects the traffic as explained in chapter Section 2.5 Failure in LDAP Operations on page 23.

The SAPC does not perform any load regulation mechanism when CUDB is overloaded.

If CUDB rejects an important amount of LDAP traffic, during the CUDB overload situation, the SAPC rejects the incoming traffic as well, affecting to the service.

To prevent that CUDB overload situation affects to the network service, some actions can be done:

- Use local policies in the PCEFs as a fallback mechanism so the IP-CAN sessions can be established without Gx control.
- Use special unknown subscriber provisioned in the SAPC (See [Database Access](#)). In this case, the SAPC can provide service to the users according to the data provisioned in the unknown subscriber profile.

Warning!

Do not use unknown subscriber including Fair Usage Profile data when accumulation is done in the CUDB.



3 Interfaces Description

Figure 11 shows the interfaces between the SAPC, CUDB, and the Provisioning System.

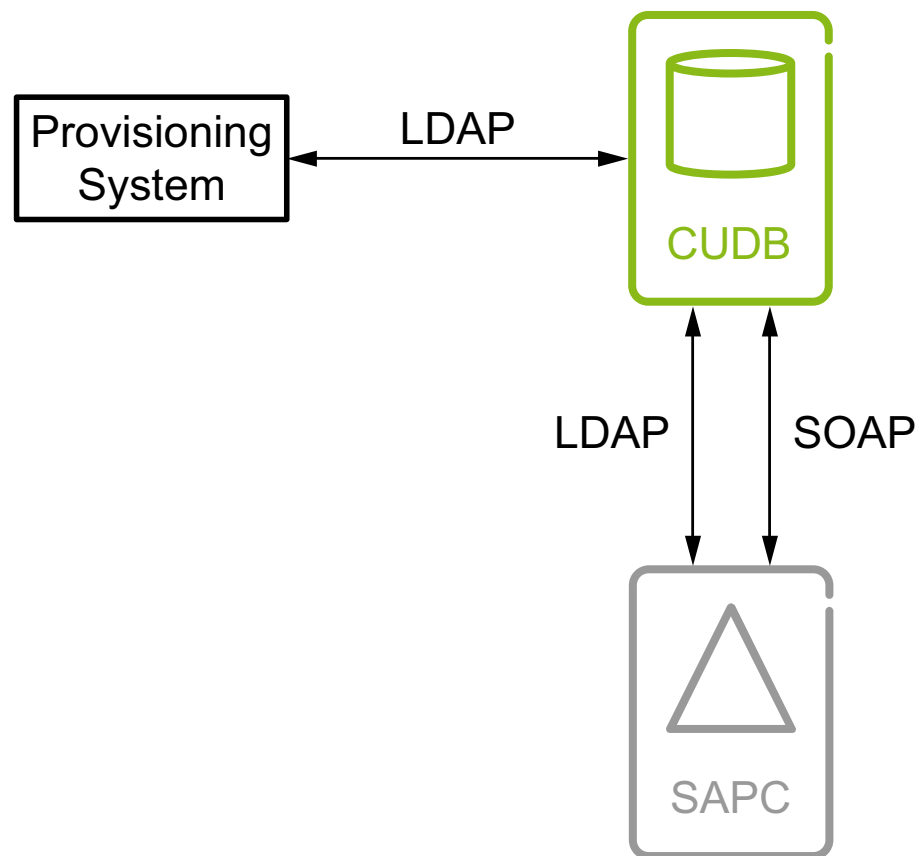


Figure 11 Interfaces

The communication between the SAPC and CUDB is performed with the following two protocols:

- LDAP v3
- SOAP

The communication between Provisioning System and CUDB is done using LDAP.



3.1 LDAP Interface

3.1.1 LDAP Messages between the SAPC and CUDB

The SAPC uses the following subset of LDAP operations:

- LDAP search
- LDAP modify
- LDAP bind
- LDAP unbind

These operations are described in Reference [13].

3.1.2 LDAP between Provisioning System and CUDB

The following operations can be used:

- LDAP search
- LDAP add
- LDAP modify
- LDAP delete
- LDAP bind
- LDAP unbind

The Provisioning System creates the SAPC entry. The **subscriber** data (including the Group entries if needed) and the traffic identities with their aliases as commented in Section 2.2.1 on page 4 are created. It can also modify the subscriber data. The Provisioning System neither creates nor modifies the Accum, OngoingSession and ClosedSession attributes, as they are created and modified by the SAPC.

LDAP Data Interchange Format (LDIF) can also be used for provisioning purpose.

3.2 SOAP Interface

SOAP is the protocol used between CUDB and the SAPC for notification purposes. As the SAPC data is stored in CUDB, CUDB notifies the SAPC about creation or changes in the subscriber profile that may affect the ongoing subscriber sessions.

When the SAPC receives the notification, it reads the subscriber data from CUDB (LDAP search) and for each active IP-CAN session of the subscriber, the SAPC executes a session reauthorization process.



A notification must be configured in CUDB to be sent to all the SAPC 's when subscriber profile is created or when specific subscriber data attributes change (see SOAP notifications configuration in UDC for configuration of attributes to monitor in CUDB).

Note: To configure the notifications for the attributes Gprio, StartD and EndD to trigger a notification when any of these attributes is modified for a group assigned to a subscriber, configure in the CUDB all the groups available in the SAPC 's as explained in SOAP notifications configuration in UDC. When a new group is added, follow the Configuration Modification Procedure in CUDB in CUDB Node Configuration Data Model Description.

CUDB sends the DN corresponding to the entry of the changed attribute (serv=PCRF, mscId=<mscId>,ou=multiSCs,<CUDB root entry> or GrpId=<groupId>,serv=PCRF, mscId=<mscId>,ou=multiSCs,<CUDB root entry>) as part of the SOAP message. The SAPC then uses the DN (no other attribute is needed for the SAPC) to identify if there is any session for the concerned subscriber. For each session, the SAPC performs an LDAP search to access the subscriber data and reauthorizes the session.

The SOAP message format is described in [SOAP Notification Interface Description](#).

CUDB does not send to the SAPC the operation XML attribute within modificationInformation XML element. To derive the type of change done on Subscriber data, the SAPC uses the following procedure:

- It looks if SSubId attribute is received in ldapAttributeName (within affectedLdapAttribute):
 - then, if oldLdapAttribute value is received and newLdapAttribute value is empty, the SAPC considers delete operation
 - or if newLdapAttributevalue is received and oldLdapAttribute value is empty, the SAPC considers create operation
- Otherwise, the SAPC considers modify operation.





4 Operation and Maintenance

4.1 Configure Database Access in the SAPC

The SAPC needs to know where the data is and what data to look for to map each attribute to its internal data model. This information is configured through the **Entity Data Sources**, see [Database Access](#).

Configure the following Entity Data Sources to access the subscriber data stored in CUDB. All Entity Data Sources are mandatory unless this document explicitly indicates that are optional:

4.1.1 Subscriber

To handle the subscriber profile stored in the CUDB, configure the following Entity Data Sources:

- **ExternalRepository**. This Entity Data Source is mandatory to use the geographical redundancy provided by the CUDB. It allows operator to configure up to 3 VIP addresses to identify the CUDB. The VIP addresses are considered as primary, secondary, and tertiary in terms of priority according to the position in the list. The VIP address in first position is the primary.

This Entity Data Source is also recommended even when no geographical redundancy is used. It minimizes to configure the VIP address and port number data in all the Entity Data Sources needed to handle the subscriber profile.

- **SubscriberIdentity**. This Entity Data Source is **optional**. It is only needed when the SAPC access the data with the traffic identity branch as explained in Section 2.2.1 on page 4. Otherwise, when subscribers are provisioned with a unique identifier in the CUDB which matches the traffic identifier, this entity is not configured.
- **Subscriber**.
- **GroupsToSubscriber**.
- **AccumulatedUsage**.
- **SessionInfo**.
- **SubscriberUnknown**. This Entity Data Source is **optional**. Configure it only to have a fallback mechanism as explained in [Database Access](#).
- **GroupsToSubscriberUnknown**. This Entity Data Source is **optional**. Configure it only to have a fallback mechanism as explained in [Database Access](#).



- SubscriberEmergency. This Entity Data Source is **optional**, but it must be configured if the emergency subscriber profile is stored in the CUDB.

4.1.2 Detailed Values of Entity Data Sources to access data from CUDB

Update the information included in the entities related to the fields url, entry and properties according to the real operator data model.

4.1.2.1 External Repository

The ExternalRepository Entity Data Source shows definition of three VIP addresses and port number to access CUDB. In the next example, the primary VIP address is 10.42.97.100, secondary VIP address is 10.42.97.101, and tertiary VIP address is 10.42.97.102. The port number is 389.

The external database VIP addresses have to represent CUDB-FE VIP addresses as explained in Section 2.3 on page 11.

```
<edit-config>
  <target>
    <running />
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <dnPrefix>dc=ManagedElement</dnPrefix>
      <networkManagedElementId>1</networkManagedElementId>
      <userLabel>Managed Element</userLabel>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
          <entityDataId>1</entityDataId>
          <EDSources xmlns="urn:com:ericsson:ecim:edsourcesmom">
            <eDSourcesId>1</eDSourcesId>
            <EDSource xmlns="urn:com:ericsson:ecim:edsourcemom" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
              <eDSourceId>ExternalRepository</eDSourceId>
              <definition>
                def ExternalRepository ()
                {
                  dataSource = {url = ""; query = ""; }
                  fieldDef = {
                    ips = "10.42.97.100;10.42.97.101;10.42.97.102";
                    port = "389";
                  }
                }
              </definition>
            </EDSource>
          </EDSources>
        </EntityData>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>
```

Example 1 External Repository

In the next example, the CUDB VIP addresses are represented in IPv6 format.



Do!

Set the IP address in square brackets in case of IPv6 format.

```
<edit-config>
  <target>
    <running />
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <dnPrefix>dc=ManagedElement</dnPrefix>
      <networkManagedElementId>1</networkManagedElementId>
      <userLabel>Managed Element</userLabel>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
          <entityDataId>1</entityDataId>
          <EDSources xmlns="urn:com:ericsson:ecim:edsourcesmom">
            <eDSourcesId>1</eDSourcesId>
            <EDSource xmlns="urn:com:ericsson:ecim:edsourcemom" xmlns:nc="urn:ietf:params:xml:ns:netconf:base1.0">
              <eDSourceId>ExternalRepository</eDSourceId>
              <definition>
def ExternalRepository ()
{
  dataSource = {url = ""; query = ""; }
  fieldDef = {
    ips = "[FE80:0000:0000:0000:0202:B3FF:FE1E:8329];[2001:1b70:8294:1995::199];[2001:db8:85a3::8a2e:370:7334]"
    port = "389";
  }
}
              </definition>
            </EDSource>
          </EDSources>
        </EntityData>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>
```

Example 2 External Repository for IPv6

4.1.2.2 Subscriber

```
<edit-config>
  <target>
    <running />
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <dnPrefix>dc=ManagedElement</dnPrefix>
      <networkManagedElementId>1</networkManagedElementId>
      <userLabel>Managed Element</userLabel>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
          <entityDataId>1</entityDataId>
```



```
<EDSources xmlns="urn:com:ericsson:ecim:edsourcesmom">
  <eDSourcesId>1</eDSourcesId>
  <EDSource xmlns="urn:com:ericsson:ecim:edsourcemom" xmlns:nc="urn:com:ericsson:ecim:edsourcesmom:nc">
    <eDSourceId>Subscriber</eDSourceId>
    <definition>

def Subscriber (argId,
                argIps = 'ExternalRepository.ips',
                argPort='ExternalRepository.port') {
  dataSource = {
    url = "ldap://{argIps}:{argPort}/serv=PCRF,mscId={argId},
          ou=multiSCs,dc=operator,dc=com?";
    query = "?scope=sub?(objectclass = * ){alias=always;
            entry=serv=PCRF,mscId={argId},ou=multiSCs,dc=operator,dc=com;}";
    properties = {user = "cn=Manager,dc=operator,dc=com"; };
  }
  fieldDef = {
    id = dataSourceField("SSubId");
    trafficIds = dataSourceField("TrafId");
    groups = dataSourceField("GrpId");
    subscribedServices = dataSourceField("WlServ");
    blacklistServices = dataSourceField("BlServ");
    sharedDataplan = dataSourceField("FamId");
    presenceReportingAreaNames = dataSourceField("SpresenceAreaName");
    eventTriggers = dataSourceField("SevTrig");
    contentFilteringProfileId = dataSourceField("ContId");
    chargingProfile = SubsChargingProfile(dataSourceField("CharP"));
    chargingSystem = OnlineChargingSystemProfile(dataSourceField("CharS"));
    customerId = dataSourceField("CustId");
    maxBearerQosProfile = BearerQosProfile(dataSourceField("MaxBQosP"));
    minBearerQosProfile = BearerQosProfile(dataSourceField("MinBQosP"));
    servicesToRedirect = dataSourceField("SrvToRed");
    usageLimits = dataSourceField("Slimit");
    sms = dataSourceField("Sms");
    pdnGwListName = PdnGwListProfile(dataSourceField("SpdnGwName"));
    spid = dataSourceField("SspId");
    mpsProfile = MpsProfile(dataSourceField("MpsP"));
    operatorSpecificInfos = dataSourceField("SOpInf");
  }
}

    </definition>
    <dbPassword>
      <cleartext></cleartext>
      <password>passwd_ldap</password>
    </dbPassword>
  </EDSource>
</EDSources>
</EntityData>
</PolicyControlFunction>
</ManagedElement>
</config>
```



```
</edit-config>
```

Example 3 Subscriber

```
<edit-config>
  <target>
    <running />
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <dnPrefix>dc=ManagedElement</dnPrefix>
      <networkManagedElementId>1</networkManagedElementId>
      <userLabel>Managed Element</userLabel>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
          <entityDataId>1</entityDataId>
          <EDSources xmlns="urn:com:ericsson:ecim:edsourcesmom">
            <eDSourcesId>1</eDSourcesId>
            <EDSource xmlns="urn:com:ericsson:ecim:edsourcemom" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
              <eDSOURCEId>GroupsToSubscriber</eDSOURCEId>
              <definition>
def GroupsToSubscriber (subsId, groupId,
                        argIps = 'ExternalRepository.ips',
                        argPort='ExternalRepository.port'){
  dataSource = {
    url = "ldap://{argIps}:{argPort}/serv=PCRF,mscId={subsId},
          ou=multiSCs,dc=operator,dc=com?";
    query = "?scope=sub?(objectclass = * )?{alias=always;
    entry=GpId={groupId},serv=PCRF,mscId={subsId},ou=multiSCs,dc=operator;}";
    properties = {user = "cn=Manager,dc=operator,dc=com"; };
  }
  fieldDef = {
    priority = dataSourceField("Gprio");
    startDate = dataSourceField("StartD");
    endDate = dataSourceField("EndD");
    durations = dataSourceField("Duration");
  }
}

              </definition>
              <dbPassword>
                <cleartext></cleartext>
                <password>passwd_ldap</password>
              </dbPassword>
            </EDSource>
          </EDSources>
        </EntityData>
      </PolicyControlFunction>
    </ManagedElement>
```



```
</config>  
</edit-config>
```

Example 4 Groups to subscriber

Next is the configuration of the EDS AccumulatedUsage.



```

<edit-config>
  <target>
    <running />
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <dnPrefix>dc=ManagedElement</dnPrefix>
      <networkManagedElementId>1</networkManagedElementId>
      <userLabel>Managed Element</userLabel>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
          <entityDataId>1</entityDataId>
          <EDSources xmlns="urn:com:ericsson:ecim:edsourcesmom">
            <eDSourcesId>1</eDSourcesId>
            <EDSource xmlns="urn:com:ericsson:ecim:edsourcemom" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
              <eDSourceId>AccumulatedUsage</eDSourceId>
              <definition>
def AccumulatedUsage(argId,
                      argIps = 'ExternalRepository.ips',
                      argPort='ExternalRepository.port')
{
  dataSource = {
    url = "ldap://{argIps}:{argPort}/serv=PCRF,mscId={argId},
          ou=multiSCs,dc=operator,dc=com?";
    query = "?scope=sub?(objectclass = * )?{alias=always;
    entry=serv=PCRF,mscId={argId},ou=multiSCs,dc=operator,dc=com;}";
    properties = {user = "cn=Manager,dc=operator,dc=com";};
  }
  fieldDef = {
    id = dataSourceField("SSubId");
    data = dataSourceField("Accum");
  }
}

              </definition>
              <dbPassword>
                <cleartext></cleartext>
                <password>passwd_ldap</password>
              </dbPassword>
            </EDSource>
          </EDSources>
        </EntityData>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>

```

Example 5 Accumulated Usage



Next is the configuration of the EDS SessionInfo.

```
<edit-config>
  <target>
    <running />
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <dnPrefix>dc=ManagedElement</dnPrefix>
      <networkManagedElementId>1</networkManagedElementId>
      <userLabel>Managed Element</userLabel>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
          <entityDataId>1</entityDataId>
          <EDSources xmlns="urn:com:ericsson:ecim:edsourcesmom">
            <EDSourcesId>1</EDSourcesId>
            <EDSource xmlns="urn:com:ericsson:ecim:edsourcemom" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
              <EDSourceId>SessionInformation</EDSourceId>
              <definition>
def SessionInfo (argId,
                  argIps = 'ExternalRepository.ips',
                  argPort='ExternalRepository.port')
{
  dataSource = {
    url = "ldap://{argIps}:{argPort}/serv=PCRF,mscId={argId},
          ou=multiSCs,dc=operator,dc=com?";
    query = "?scope=sub?(objectclass = * )?{alias=always;
    entry=serv=PCRF,mscId={argId},ou=multiSCs,dc=operator,dc=com;}";
    properties = {user = "cn=Manager,dc=operator,dc=com";}
  }
  fieldDef = {
    id = dataSourceField("SSubId");
    ongoingSession = dataSourceField("OngoingSession");
    closedSession = dataSourceField("ClosedSession");
  }
}

              </definition>
              <dbPassword>
                <cleartext></cleartext>
                <password>passwd_ldap</password>
              </dbPassword>
            </EDSource>
          </EDSources>
        </EntityData>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>
```




Example 6 Session Information

4.1.2.3 **Subscriber and SubscriberIdentity when Subscribers are provisioned with several Identifiers**

When any subscriber is provisioned with a traffic identifier different than the administrative identifier, the SAPC accesses the data with the traffic identity branch as explained in Section 2.2.1 on page 4. In this scenario, configure the SubscriberIdentity EDS as explained in the following example:



```

<edit-config>
  <target>
    <running />
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <dnPrefix>dc=ManagedElement</dnPrefix>
      <networkManagedElementId>1</networkManagedElementId>
      <userLabel>Managed Element</userLabel>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
          <entityDataId>1</entityDataId>
          <EDSources xmlns="urn:com:ericsson:ecim:edsourcesmom">
            <eDSourcesId>1</eDSourcesId>
            <EDSource xmlns="urn:com:ericsson:ecim:edsourcemom" xmlns:nc=
              <eDSourceId>SubscriberIdentity</eDSourceId>
              <definition>
def SubscriberIdentity (argId,
                        argIps='ExternalRepository.ips',
                        argPort='ExternalRepository.port') {
  dataSource = {
    url = "ldap://{argIps}:{argPort}/TRAFFICID={argId},
          dc=trafficId,ou=identities,dc=operator,dc=com?";
    query = "?scope=base?(objectclass = * )?{alias=always;}}?";
    properties = {user = "cn=Manager,dc=operator,dc=com";};
  }
  fieldDef =
  {
    trafficId = arg("argId");
    adminId = dataSourceField("mscId");
  }
}
            </definition>
            <dbPassword>
              <cleartext></cleartext>
              <password>passwd_ldap</password>
            </dbPassword>
          </EDSource>
        </EDSources>
      </EntityData>
    </PolicyControlFunction>
  </ManagedElement>
</config>
</edit-config>

```

Example 7 Subscriber Identity



4.1.2.4 Emergency Subscriber

In the next example, it is included the EDS related to emergency subscriber.

Note: Some fieldDefs whose function do not apply to the emergency subscriber, thus these fields are preconfigured to ""..

```
<edit-config>
  <target>
    <running />
  </target>
  <config>
    <ManagedElement xmlns="urn:com:ericsson:ecim:ComTop">
      <managedElementId>1</managedElementId>
      <dnPrefix>dc=ManagedElement</dnPrefix>
      <networkManagedElementId>1</networkManagedElementId>
      <userLabel>Managed Element</userLabel>
      <PolicyControlFunction xmlns="urn:com:ericsson:ecim:sapcmom">
        <policyControlFunctionId>1</policyControlFunctionId>
        <EntityData xmlns="urn:com:ericsson:ecim:entitydatamom">
          <entityDataId>1</entityDataId>
          <EDSources xmlns="urn:com:ericsson:ecim:edsourcesmom">
            <eDSourcesId>1</eDSourcesId>
            <EDSource xmlns="urn:com:ericsson:ecim:edsourcemom" xmlns:nc="urn:ietf:params:xml:ns:ietf:params:xml:ns:ietf:po">
              <eDSourceId>SubscriberEmergency</eDSourceId>
              <definition>
def SubscriberEmergency (argId,
                        argIps = 'ExternalRepository.ips',
                        argPort='ExternalRepository.port') {
  dataSource = {
    url = "ldap://{argIps}:{argPort}/serv=PCRF,mscId={argId},
          ou=multiSCs,dc=operator,dc=com?";
    query = "?scope=sub?(objectclass = * )?{alias=always;
    entry=serv=PCRF,mscId={argId},ou=multiSCs,dc=operator,dc=com;}";
    properties = {user = "cudbUser=PCRF,ou=admin,dc=operator,dc=com";};
  }
  fieldDef = {
    id = dataSourceField("SSubId");
    trafficIds = "";
    groups = "";
    subscribedServices = dataSourceField("WlServ");
    blacklistServices = dataSourceField("BlServ");
    sharedDataplan = "";
    presenceReportingAreaNames = "";
    eventTriggers = dataSourceField("SevTrig");
    contentFilteringProfileId = "";
    chargingProfile = SubsChargingProfile(dataSourceField("CharP"));
    chargingSystem = "";
    customerId = "";
    maxBearerQosProfile = BearerQosProfile(dataSourceField("MaxBQosP"));
    minBearerQosProfile = BearerQosProfile(dataSourceField("MinBQosP"));
  }
}
              </definition>
            </EDSource>
          </EDSources>
        </EntityData>
      </PolicyControlFunction>
    </ManagedElement>
  </config>
</edit-config>
```



```

        servicesToRedirect = "";
        usageLimits = "";
        sms = "";
        pdnGwListName = "";
        spid = "";
        mpsProfile = "";
    }
}

        </definition>
        <dbPassword>
            <cleartext></cleartext>
            <password>passwd_ldap</password>
        </dbPassword>
    </EDSource>
</EDSources>
</EntityData>
</PolicyControlFunction>
</ManagedElement>
</config>
</edit-config>

```

Example 8 Subscriber Emergency

4.2 Configure SOAP Notifications in CUDB

The commands to configure CUDB to send SOAP notifications to the SAPC's are described in SOAP notifications configuration in UDC.

4.3 Subscriber Data Extension in CUDB

As explained in [Database Access](#), operators may need to use their own extra data (not known in advanced by the SAPC) to be evaluated in the SAPC **policy conditions**. So, from the SAPC point of view, it is possible to extend `Subscriber` entry with extra attributes to the ones considered in Section 2.2.2 on page 6.

To do that, follow next steps (for each additional attribute in the subscriber profile):

1. Extend `sapc.schema` to include the new attribute. New object class is recommended for the new attributes. Follow instructions from CUDB application integration guide, see CUDB Application Integration Guide.

Example of new added LDAP attribute: `OpInf`

2. Reconfigure Subscriber EDS including the following line within `fieldDef` block:

```
extraData = dataSourceField("OpInf");
```



3. `Subscriber.extraData` is ready to be used within policy conditions.

4.4 LDAP Operations for Provisioning System

4.4.1 Creation of Entries

The procedure that the Provisioning System has to do to create a the SAPC subscriber on CUDB is the following:

1. Perform an LDAP add to create a `trafficId` entry with the value received as subscriber identity in the provisioning order, pointing (alias referencing) to the `mscId` entry.
2. LDAP add of `mscId` entry (CUDBMultiServiceConsumer object class).
3. LDAP add of `serv=PCRF` entry with at least the following object classes which are mandatory when a subscriber is created (the SAPC, top, and CUDBServiceAuxiliary object classes), under `mscId` entry.
4. Optionally, in case of several subscriber traffic identifiers associated to the administrative identifier, include `TrafId` attribute in the SAPC object class and perform an LDAP add of its corresponding `trafficId` entry pointing (alias referencing) to the `mscId` entry.
5. Optionally, if there is multiple service offering handling priority or temporary subscriptions, LDAP add of a `GrpId` entry (Group object class) per each value of `GrpId` attribute of the SAPC object class.

Here there is an example of how the SAPC entry with some subscriber data is created with LDIF including creation of `trafficId` identities with an alias pointing to the corresponding `mscId`:

```
#####
# TRAFFICID (alias) create. Subscriber identifier alias
# pointing mscId.
#####
dn: TRAFFICID=2233445566,dc=trafficId,ou=identities,dc=operator
changetype:add
objectClass: alias
objectClass: top
objectClass: TRAFFICID
TRAFFICID: 2233445566
aliasedObjectName: mscId=2233445566,ou=multiSCs,dc=operator

#####
# mscId create
#####

dn: mscId=2233445566,ou=multiSCs,dc=operator
changetype:add
objectClass: CUDBMultiServiceConsumer
```



```
objectClass: top
mscId: 2233445566

#####
# SAPC entry and subscriber data create
#####
dn: serv=PCRF,mscId=2233445566,ou=multiSCs,dc=operator
changetype:add
objectClass: SAPC
objectClass: SAPC2
objectClass: top
objectClass: CUDBServiceAuxiliary
serv:PCRF
SSubId: 2233445566
TrafId: 445783433
GrpId: group1
GrpId: group_QBAU_GGSN
GrpId: group_FixedAccess
CustId: aB111195000000cD

#####
# TRAFFICID (alias) create. An alias for the TrafficId (different
# from Subscriber identifier).
#####
dn: TRAFFICID=445783433,dc=trafficId,ou=identities,dc=operator
changetype:add
objectClass: alias
objectClass: top
objectClass: TRAFFICID
TRAFFICID: 445783433
aliasedObjectName: mscId=2233445566,ou=multiSCs,dc=operator

#####
# GroupId entry and Group data creation
#####
dn: GrpId=group_FixedAccess,serv=PCRF,
  mscId=2233445566,ou=multiSCs,dc=operator
changetype:add
objectClass: SAPC3
objectClass: top
GrpId:group_FixedAccess
Gprio: 1
StartD: 31-12-2000T23:59:59
EndD: 11-11-2013T13:59:59
```

Example 9 Subscriber Data provisioning



4.4.2 Deletion of Entries

The procedure that the Provisioning System has to do to remove the SAPC subscriber from CUDB is the following:

1. LDAP delete of TRAFFICID entries for the traffic identifiers different from the subscriber identity received in the provisioning order.
2. Optionally, if there is multiple service offering, LDAP delete for each of the GrpId entries under serv=PCRF entry.
3. LDAP delete of serv=PCRF entry.
4. LDAP delete of mscId entry.
5. LDAP delete of TRAFFICID entry for the traffic identifier that is the subscriber identity received in the provisioning order.

4.4.3 Addition/Removal of Traffic Identifiers

It is responsibility of the Provisioning System to keep coherence between traffic and administrative subscriber identifiers in CUDB.

4.5 Alarms

See [Database Access](#).

4.6 Logging

See [Database Access](#).





5 Capabilities

The following capabilities must be considered:

- Enough bandwidth. The link between the SAPC and CUDB is dimensioned to be able to handle the required bandwidth according to the traffic scenario and hardware configuration.
- The maximum One-Way Delay (OWD) of the link must be no more than 50 ms and the packet loss rate must be no more than 10^{-3} . The jitter added to the OWD must remain within the requirements for the maximum OWD.
- The SAPC establishes a maximum of 64 LDAP sessions to a CUDB-FE VIP address per traffic VM.
- The SAPC waits until the result is returned after an LDAP query for 2 seconds.
- The SAPC waits until the result is returned after an LDAP Bind for 4 seconds.
- The SAPC failback timer waits for 60 seconds before checking if a faulty high priority VIP address is available again to perform failback. On connectivity lost (see Section 2.4.3 Connectivity Lost on page 22), the failback timer waits for 1 second.
- The SAPC reattempts up to two times a query that has been answered with LDAP error code 52, timed out or received a TCP RST. After these reattempts, the SAPC considers the query as failed.
- The SAPC handles a maximum of 20 outstanding LDAP requests per LDAP session.
- The SAPC handles an inactivity timer for the TCP connections of 5 minutes.





Reference List

Ericsson Documents

- [1] CUDB Application Integration Guide - 7/1553-HDA 104 03/10 Uen
- [2] CUDB LDAP Interwork Description - 1/155 19-HDA 104 03/10 Uen
- [3] CUDB Node Configuration Data Model Description - 1/19202-CSH 109 067/10 Uen
- [4] Database Access
- [5] System Administrator Guide
- [6] SOAP Notification Interface Description
- [7] SOAP notifications configuration in UDC - 1/1553-CXP 902 0038/13 Uen
- [8] SAPC LDAP Schema for UDC - 1/1531-CXP 903 0138/7 Uen
- [9] User Data Consolidation (UDC) , Technical Product Description - 221 02-FGC 101 1571 Uen
- [10] Session Context Exposure User Guide
- [11] Analytics REST API

Standards

- [12] User Data Convergence (UDC) - 3GPP TS 23.335
- [13] Lightweight Directory Access Protocol (LDAP): The Protocol - RFC 4511