

# Function Specification Resource Activation

## Ericsson Dynamic Activation 1

---

### FUNCTION SPECIFICATION

**Copyright**

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

**Disclaimer**

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

**Trademark List**

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



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Purpose and Scope	1
1.2	Target Groups	1
1.3	Typographic Conventions	1
1.4	Prerequisites	1
<b>2</b>	<b>Function Overview</b>	<b>2</b>
<b>3</b>	<b>Inbound Interfaces</b>	<b>2</b>
3.1	CAI and MML	3
3.2	CAI3G and Async CAI3G	3
3.3	CLI	4
3.4	EDIFACT	4
<b>4</b>	<b>Business Logic</b>	<b>5</b>
4.1	Standard Provisioning Business Logic	5
4.2	Customer Adaptation for Provisioning Business Logic	6
4.3	Standard Cluster Strategy	6
4.3.1	ActiveActive	6
4.3.2	ActiveActive-BestEffort	7
4.3.3	PrimaryBackup	9
4.3.4	ActiveActiveQueue	9
4.3.5	ActivePassive	12
4.3.6	MultipleAIR	13
4.4	Customer Adaptation for Cluster Strategy	15
<b>5</b>	<b>Outbound Connectivity</b>	<b>15</b>
5.1	Generic Protocol Support	15
5.2	Network Element Management	16
5.3	Connection Robustness	18
<b>6</b>	<b>Authorization and Access Control</b>	<b>24</b>
6.1	Provisioning Access Control	24
<b>7</b>	<b>Fault Tolerance</b>	<b>26</b>
7.1	Operations without Automatic Rollback Functionality	28
7.2	Operations without Fault Tolerance	28



<b>8</b>	<b>Prevent Concurrent Subscriber Provisioning</b>	<b>29</b>
<b>9</b>	<b>Replay Operation</b>	<b>30</b>
9.1	Automatic Replay Operation	31
9.1.1	Automatic Provisioning Block	35
9.1.2	RESTful Notification Interface	37
9.2	Manual Replay Operation	38
<b>10</b>	<b>Configurable Loose Error Handling</b>	<b>39</b>
<b>11</b>	<b>CAI3G Distributed Configuration</b>	<b>39</b>
11.1	Request Distribution	40
11.1.1	Distributing a Northbound Synchronous CAI3G Request	40
11.1.2	Distributing a Synchronous Request through SV	40
11.2	Routing Configuration	41
11.3	Faults or Errors	42
<b>12</b>	<b>License Management</b>	<b>42</b>
12.1	Enforcement of Subscriber Licensing	42
12.2	Expiration Notifications	43
12.3	Blocked Provisioning	43
	<b>Reference List</b>	<b>45</b>



# 1 Introduction

This section is an introduction to this document. It contains information about the prerequisites, purpose, scope, and target group for the document. This section also contains explanations of typographic conventions used in this document.

## 1.1 Purpose and Scope

The purpose of this document is to give detailed description about the Resource Activation functions in Ericsson™ Dynamic Activation (EDA) and to provide an entry of related documentation. Descriptions of functional behavior for the features hosted by Dynamic Activation can be found in the respective Function Specification.

For the full list and description of Dynamic Activation documents, see *Library Overview*, Reference [1].

## 1.2 Target Groups

The target groups for this document are as follows:

- System Administrator
- Application Administrator
- Any other personnel who intend to learn about Dynamic Activation

For more information about different target groups, see *Library Overview*, Reference [1].

## 1.3 Typographic Conventions

Typographic conventions are described in the document *Library Overview*, Reference [1].

## 1.4 Prerequisites

It is assumed that the readers have basic product knowledge of Dynamic Activation. The document recommended for such information is *Product Overview*, Reference [2].

## 2 Function Overview

Dynamic Activation can provide **Resource Activation** services, such as provisioning 2G/3G voice & data, LTE, IMS/VoLTE, TV, and other applications for subscribers. Figure 1 shows the conceptual components of Dynamic Activation.

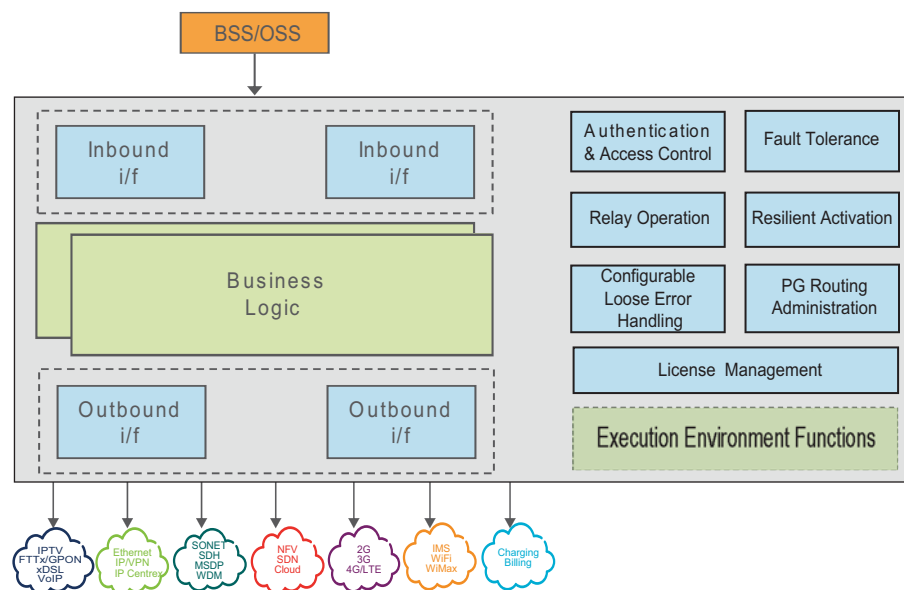


Figure 1 Dynamic Activation Logical Architecture

The common function provides access control and routing of requests to the deployed Business Logic Modules, as well as handles threading and queuing of requests.

The common functions of the platform are described in the *Function Specification Dynamic Activation Execution Environment*, Reference [3].

## 3 Inbound Interfaces

The inbound interfaces layer supports a number of default provisioning interfaces. It handles the connectivity with northbound systems and internally takes the responsibility to convert an inbound interface to the internal data representation supported by the business logic modules.



These protocols operate on a Managed Object (MO), that is, the functional representation of the provisioning target. An MO can represent the complete information for a southbound application, for example Number Portability. An MO can also reflect a subset of the information for an application, for example Close User Group of the Home Location Register (HLR).

The physical implementation of the networks, protocols, and interactions among southbound systems is hidden by an MO. It provides the operators with a uniform interface to all southbound systems.

The basic operations supported for most of the MOs are `Create`, `Set`, `Get`, and `Delete`.

A provisioning request coming from a northbound system that operates on an MO is called a Customer Service Order (CSO). One CSO can generate one or many Network Service Orders (NSO) to southbound systems.

If communication errors to Network Elements (NEs) are detected, a fault code is returned to the Business Support System (BSS). The current BSS session is not ended.

In case the northbound system prefers another interface than the default ones, it is possible to tailor a customer solution as a service.

The default provisioning interfaces are described in the coming sessions.

## 3.1 CAI and MML

Customer Administration Interface (CAI) and Man-Machine Language (MML) interfaces provide backward compatibility between classic HLRs and layered UDC in order to simplify the migration.

For more information about the functionality provided through the CAI and the MML interfaces, refer to *Function Specification Layered HLR*, Reference [4].

## 3.2 CAI3G and Async CAI3G

Customer Administration Interface Third Generation (CAI3G) is aimed to provide a uniform provisioning interface for NEs to contain the subscription data.

For more information about the functionality provided through the CAI3G and Async CAI3G interfaces, refer to *Function Specification Dynamic Activation Execution Environment*, Reference [3].

For the functionality provided through the CAI3G interface, see the following documents:

- *Function Specification Layered HLR*, Reference [4]
- *Function Specification Layered IMS*, Reference [5]



- *Function Specification Layered LTE EPC*, Reference [6]
- *Function Specification Layered EIR*, Reference [7]
- *Function Specification Layered IPWorks/AAA*, Reference [8]
- *Function Specification BCE*, Reference [18]
- *Function Specification PGM*, Reference [19]
- *Function Specification IPWorks/ENUM*, Reference [20]
- *Function Specification MTAS*, Reference [21]
- *Function Specification SAPC*, Reference [27]

### 3.3 CLI

Command-Line Interface (CLI) is used for long lasting asynchronous provisioning activities such as massive update and location procedure. CLI is also used for multi-destination provisioning activities such as updating service associated data in all HLR Front Ends.

The CLI is accessed by using telnet clients or direct socket connections. The number of concurrent logons are 10 per PL node. There is no time-out defined for CLI.

For security purposes, the CLI can be protected by SSL.

The supported character set in a CLI interface is ASCII.

For more information about the CLI, see *Introduction to CLI for Layered Applications*, Reference [9].

For the functionality provided through the CLI, see the following documents:

- *Function Specification Layered HLR*, Reference [4]
- *Function Specification Layered LTE EPC*, Reference [6]
- *Function Specification Layered EIR*, Reference [7]
- *Function Specification Layered IPWorks/AAA*, Reference [8]
- *Function Specification SAPC*, Reference [27]

### 3.4 EDIFACT

The Generic Mediation Device (GMD) EDIFACT service provides an asynchronous provisioning feature to support a new EDIFACT northbound





protocol for the charging and billing in one (called convergent charging in previous version) solution in Dynamic Activation.

EDIFACT is the interface between the Business Support and Control System (BSCS)/GMD and Dynamic Activation in the Charging and Billing in One (CBiO) solution. Dynamic Activation receives the hardware independent EDIFACT messages from GMD, and then generates the messages required by NEs.

Dynamic Activation provides EDIFACT as northbound interface for CBiO, which allows system integrators to develop Customer Adaptation (CA) according to the customer requirements for EDIFACT interface.

For more information about the EDIFACT, see *Generic EDIFACT Interface Specification*, Reference [22].

For the functionality provided through the EDIFACT interface, see *Solution Description Charging and CBiO*, Reference [23].

## 4 Business Logic

### 4.1 Standard Provisioning Business Logic

The provisioning business logics are runtime deployable Java™ code that provides specific business features.

For details of a default feature, refer to the respective Function Specification of the feature.

The business logic is managed in the following Dynamic Activation GUI. For more information, refer to *User Guide for Resource Activation*, Reference [10].

#### **Activation Logic Management**

- View available activation logics properties, including configuration data and the static target data.
- Edit configuration data for an activation logic: some business logics come with different flavor of provisioning behavior, this function allows user to define the wanted behavior in runtime.
- Back up configuration data for an activation logic: before changing configuration data, this function can be used to save a copy of the current settings. Multiple backups can be saved in the system, each with a distinguished name.



- Restore configuration data for an activation logic: this function restores the configuration settings from one of the backups.

## 4.2 Customer Adaptation for Provisioning Business Logic

Dynamic Activation also supports to customize provisioning Business Logic (BL) with Java Data Views (JDVs), outbound connectors. For more information about JDVs and SV, refer to *Customer Adaptation Development Guide for Resource Activation*, Reference [14].

## 4.3 Standard Cluster Strategy

The cluster strategy is a business logic to orchestrate the provisioning behavior of the clustered NEs according to the designated relationship between the NEs.

**Note:** Cluster strategies feature are not applicable for Multimedia Telephony Application Server (MTAS) and NEs in the User Data Consolidation (UDC) solution.

For how to manage the Cluster strategy instance, refer to *User Guide for Resource Activation*, Reference [10].

When a Cluster strategy is used, system returns failure message if a request is not successful in the entire cluster, which means that some NEs in the cluster are provisioned successfully, while failures can occur on the other NEs. In such a case, if rollback is required on the successful NEs, the rollback must be defined in the BL in the Network Abstraction layer.

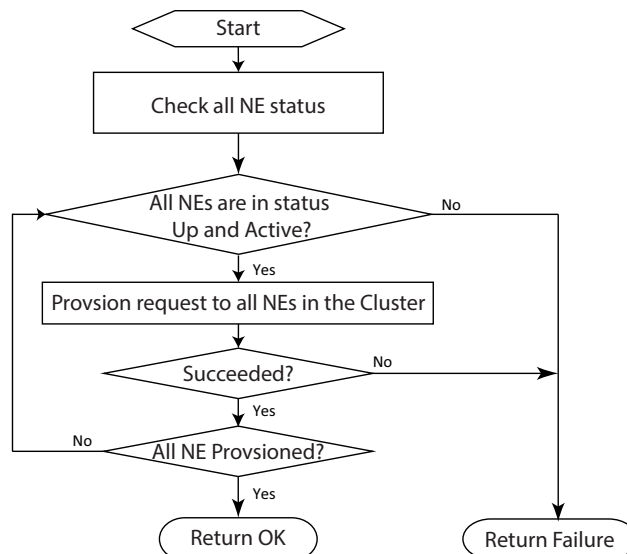
### 4.3.1 ActiveActive

The ActiveActive Cluster Strategy is used for NEs that are provisioned with identical data. For Create, Set or Delete operations, all NEs in the cluster have to be provisioned correctly before declaring the request as successful. For Get operation, it only needs to get data from one NE that is in status `Up` and `Active`, regardless of the state and status of the rest of the NEs in the cluster.

It is possible to manually set the state of the NE to `Inactive` for maintenance purpose. But if one or more NEs in the cluster are blacklisted or in state `Inactive`, the Create, Set or Delete operation does not provision to any NE in the cluster.

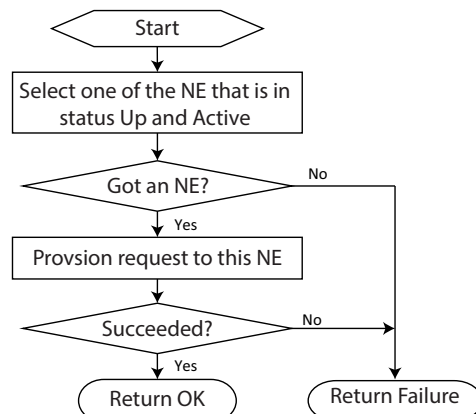
Errors that occur during the provisioning process are handled by the activation logics.

The following figure shows the workflow of ActiveActive strategy for CREATE/SET/DELETE operation:



**Figure 2** ActiveActive Strategy – CREATE/SET/DELETE Operation

The following figure shows the workflow of ActiveActive strategy for GET operation:



**Figure 3** ActiveActive Strategy – GET Operation

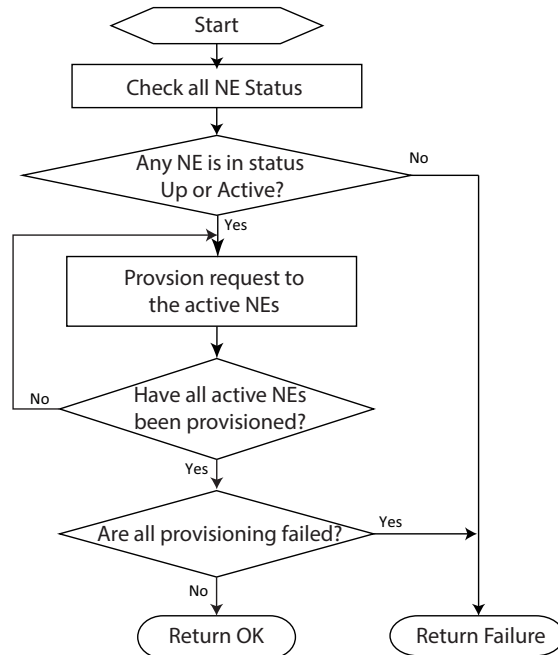
### 4.3.2

### ActiveActive-BestEffort

In ActiveActive-BestEffort Cluster strategy, the request is provisioned to all active NEs in the cluster, but system only returns failure when requests fail on all active NEs.

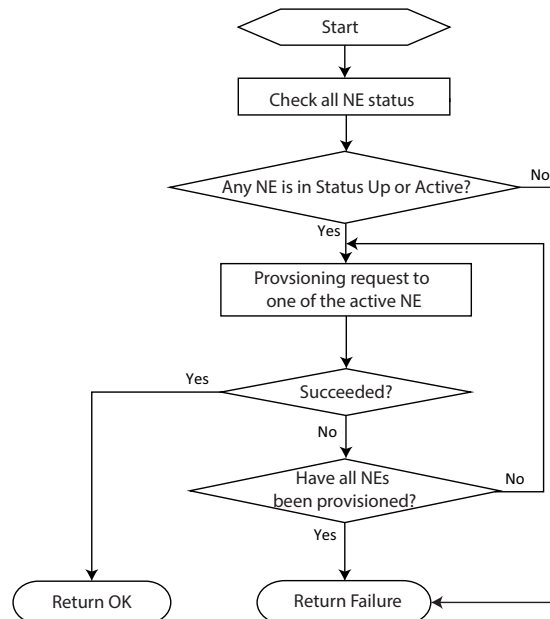
The GET operation is different from other operations (Create, Set or Delete). For Get operations, it retrieves data from the first NE with status Up and Active. If error occurs, it tries the next available NE until data is retrieved or NE fails.

The following figure shows the workflow of ActiveActive-BestEffort strategy for CREATE/SET/DELETE operation:



**Figure 4** ActiveActive-BestEffort Strategy – CREATE/SET/DELETE Operation

The following figure shows the workflow of ActiveActive-BestEffort strategy for GET operation:



**Figure 5** ActiveActive-BestEffort Strategy– GET Operation

### 4.3.3 PrimaryBackup

The PrimaryBackup strategy is used for the cluster that has one primary NE, and one or more backup NEs.

Dynamic Activation first sends the request to the primary NE. If the request has been successfully provisioned to the primary NE, the same request is sent to all backup NEs. Otherwise, Dynamic Activation does not provision this request to other NEs and system returns failure. If the request has been successfully provisioned to primary NE, but the request fails on the backup NEs, the provisioning operation will be stored in processing queue and system returns OK.

The following figure shows the workflow of PrimaryBackup strategy with one primary and one backup NE:

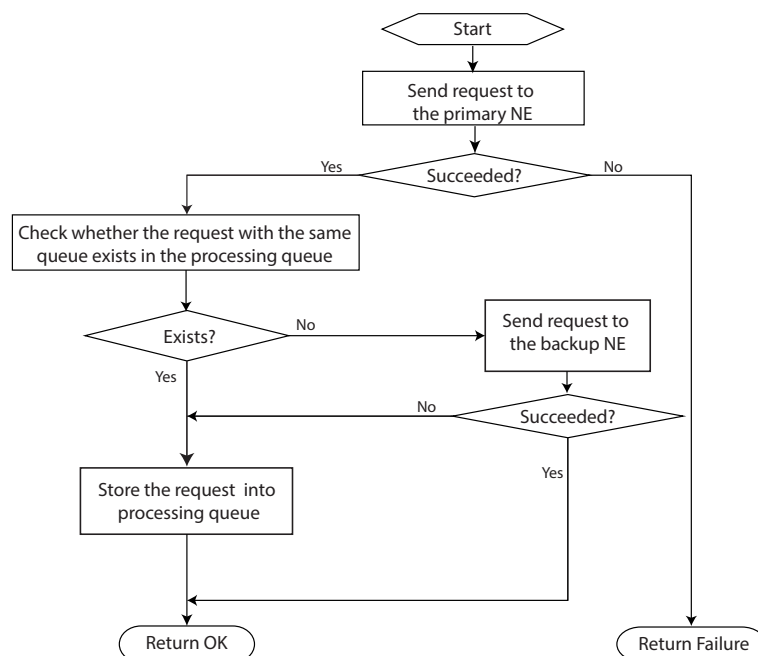


Figure 6 PrimaryBackup Strategy with One Primary and One Backup NE

### 4.3.4 ActiveActiveQueue

In ActiveActiveQueue strategy, the request is provisioned to all NEs in the cluster. If the provisioning fails on an NE, the failed command and its queue id are stored into a temporary list. Then system provisions the request to the next NE. After all NEs are provisioned and at least one provisioning succeeds, the data in temporary list is stored into processing queue. If the provisioning fails on all NEs, the complete provisioning operation fails. In this case, the data in temporary list is not stored in processing queue, and an error will be returned.

The GET operation is different from other operations (Create, Set or Delete). For Get operation, it only needs to successfully get data from one of the NE,

and the data in temporary list is only stored into processing queue when the provisioning operation succeeds.

The following figure shows the workflow of ActiveActiveQueue strategy of CREATE/SET/DELETE operation:

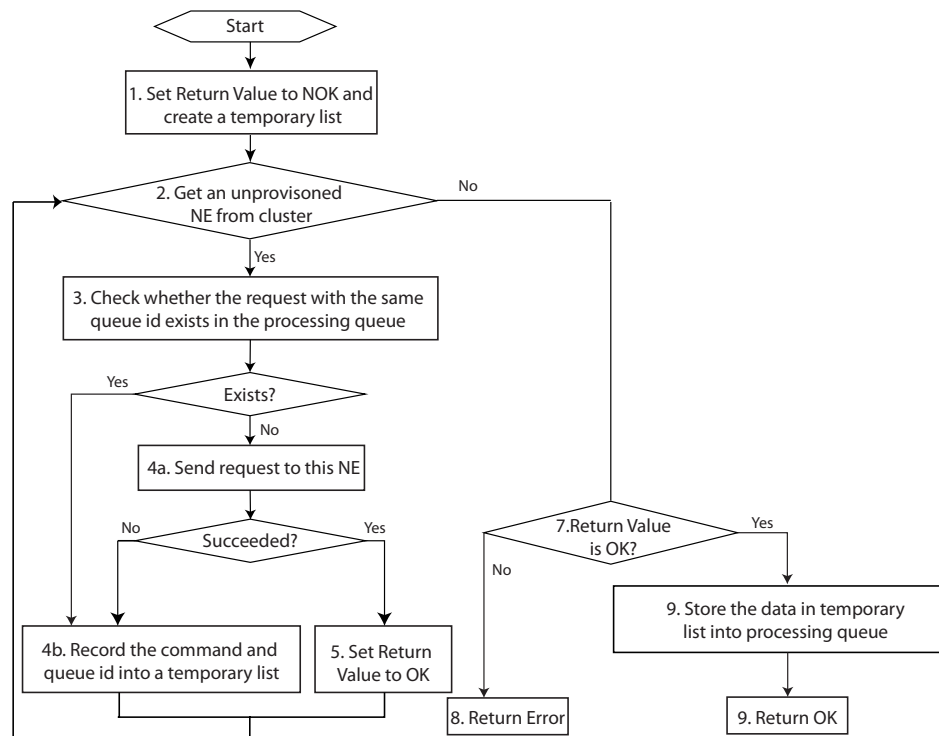


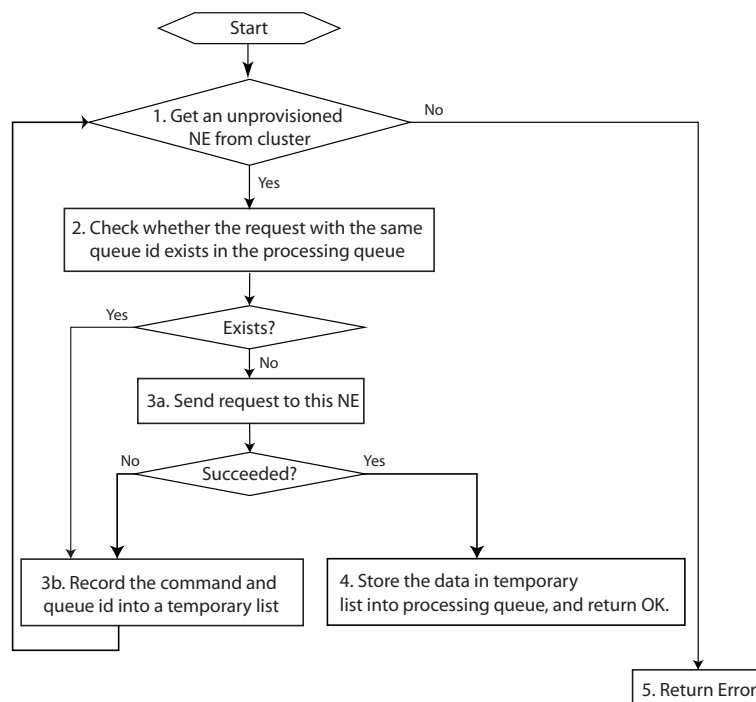
Figure 7 ActiveActiveQueue Strategy – CREATE/SET/DELETE Operation

- 1 System initializes the return value to "NOK" and creates a temporary list to record the failed commands and queue ids.
- 2 Get an unprovisioned NE from the cluster.
- 3 Check whether the request with the same queue id exists in the processing queue or not.
- 4 If request does not exist in the queue, system sends the request to this NE. Otherwise, system records this command and its queue id into temporary list.
- 5 If the request successfully provisioned, the return value is updated from "NOK" to "OK". Otherwise, the failed command and its queue id are recorded into the temporary list.
- 6 Repeat Step 2 to Step 5 for the next NE from the cluster until all the NEs in the cluster have been tried.
- 7 After all NEs are provisioned, system checks the return value.



- 8 If the return value is "NOK", it means the request fails on all NEs and an error will be returned. The provisioning operation fails.
- 9 If the return value is "OK", system stores the data recorded in the temporary list into processing queue. The provisioning operation succeeds .

The following figure shows the workflow of ActiveActiveQueue strategy of GET operation:



*Figure 8 ActiveActiveQueue Strategy – GET Operation*

- 1 Get an unprovisioned NE from the cluster.
- 2 Check whether the request with the same queue id exists in the processing queue or not.
- 3 If request does not exist in the queue, system sends the request to this NE. Otherwise, system records this command and its queue id into temporary list.
- 4 If the request succeeds, system stores the data recorded in the temporary list into processing queue and returns OK. Otherwise, system stores the failed command and its queue id into temporary list and the request is sent to the next unprovisioned NE.
- 5 If the request fails on all NEs, an error will be returned.



### 4.3.5 ActivePassive

The ActivePassive strategy is used to support redundant NE deployment scenarios which has one NE with provisioning state `On-Active` and one or more NEs with provisioning state `On-Passive/Off` in the cluster.

Three NE states are supported in this cluster strategy:

- `On-Active`, indicates that the NE is ready to receive provisioning requests.
- `On-Passive`, indicates that the NE is stand-by.
- `Off`, indicates that provisioning is not sent to this NE.

All the requests are sent to the NE with provisioning state `On-Active`. If the provisioning fails on the NE with provisioning state `On-Active` due to the link error or match the error code for detecting passive status of NE, this NE turns into provisioning state `On-Passive`. Then, Dynamic Activation provisions the request to the NEs with provisioning state `On-Passive` in the cluster one at a time until an NE is qualified to turn to provisioning state `On-Active`.

The NE is qualified in the following situations:

- the provisioning succeeds on this NE.
- the provisioning fails on this NE but the response is neither link error nor the error code configured in the cluster strategy GUI.

If no `On-Passive` NE is qualified to set to `On-Active`, the original NE with provisioning state `On-Active` remains active.

The following figure shows the workflow of the ActivePassive strategy:



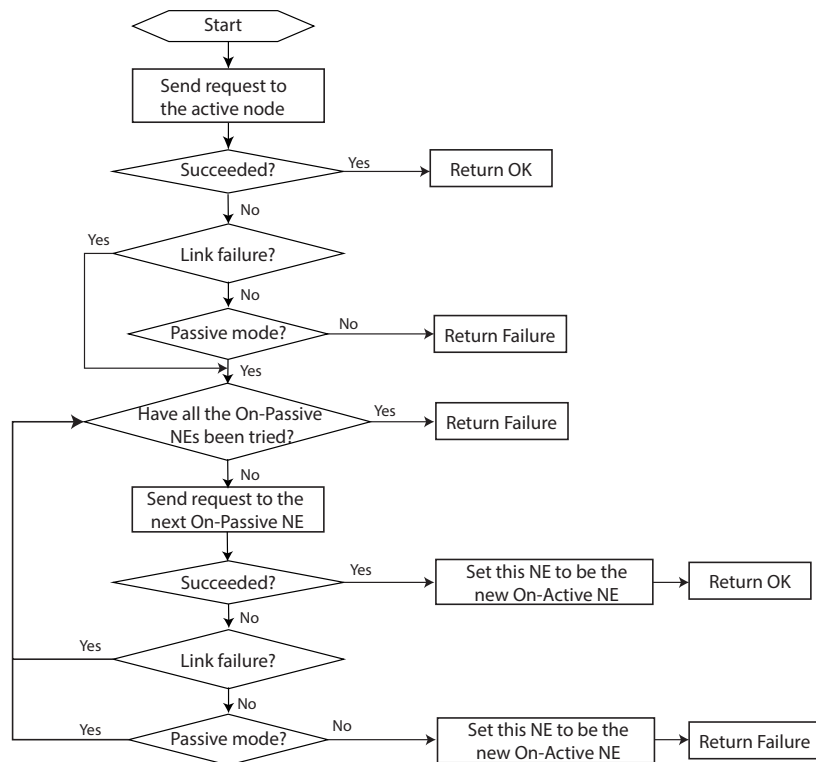


Figure 9 ActivePassive Strategy

#### 4.3.6

### MultipleAIR

MultipleAIR strategy supports the cluster that has the following relationship between NEs:

- One primary NE and one or more secondary NEs in a cluster.
- From the subscriber data point of view, the subscriber data are automatically synchronized among all AIRs. Therefore, Dynamic Activation only needs to successfully provision one of NE in a cluster.
- From the operation point of view, the CREATE operation is different from other operations (GET/SET/DELETE).

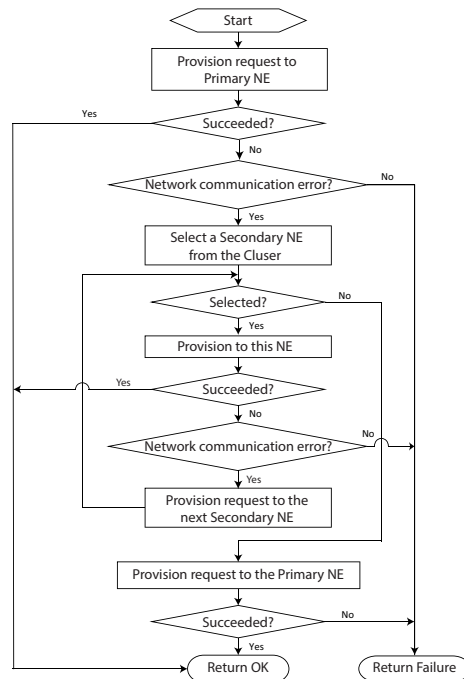
#### — CREATE Operation

For CREATE operation, the Primary NE is tried first, then the rest of NEs are tried in case of failure.

Dynamic Activation sends a CREATE request to the Primary NE for CREATE in the cluster. If this provision succeeds, the whole provision succeeds. Otherwise, if network communication error happens, Dynamic Activation will try to provision the secondary NEs one by one in the cluster until it succeeds. During the process, a functional error (such as syntax error, faulty data) will end the whole process

with a failure response. If all NEs have been tried without success, the Primary NE will be retried one more time. If it still fails, the whole process will be ended with failure.

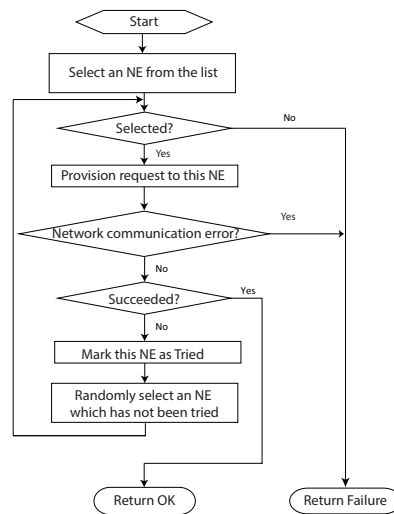
The following figure shows the MultipleAIR Strategy for CREATE operation:



#### — GET/SET/DELETE Operation

Dynamic Activation randomly chooses an NE from the cluster to provision. If it succeeds, the whole provision succeeds. Otherwise, if network communication error happens, Dynamic Activation will try to provision the rest NEs in the cluster in a random way until it succeeds. During the process, a functional error will end the whole process with a failure response. If all NEs have been tried without success, the whole process will be ended with failure.

The following figure shows the MultipleAIR strategy for GET/SET/DELETE operation:



## 4.4 Customer Adaptation for Cluster Strategy

Dynamic Activation also supports to customize cluster strategy. For more information, refer to *Customer Adaptation Development Guide for Resource Activation* Reference [14].

# 5 Outbound Connectivity

Dynamic Activation provides unified outbound connection management solution that can be configured in runtime to serve business logics.

## 5.1 Generic Protocol Support

By default, Dynamic Activation supports Lightweight Directory Access Protocol (LDAP), Telnet, SSH, Provisioning Notification (SOAP), CAI3G (HTTP/HTTPS) and HTTP as outbound protocols. New protocols can be introduced as a system integration service.

These protocols can be associated to a target NE. The protocol behavior for a specific NE can be configured in the GUI when defining the NE.

For information on how to define a protocol for an NE, see *User Guide for Resource Activation*, Reference [10].

## 5.2 Network Element Management

NE management is implemented in two layers, the service abstraction layer and the connection management layer. The solid arrows in the Figure 10 show the use flow while the dotted arrows show where the entities belong to.

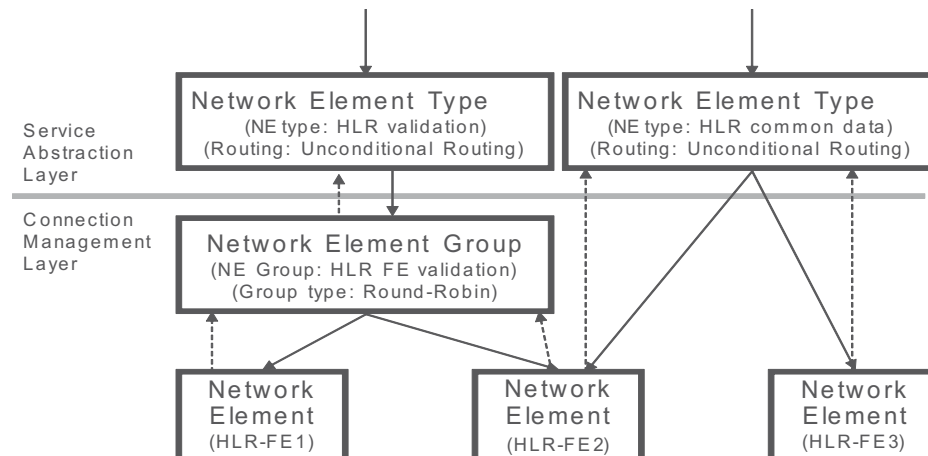


Figure 10 NE Management with Example

NE types are defined by the business logic. The default NE types can be loaded to the system at system initial configuration. Routing method can then be associated with the NE type via the GUI. One NE type can only be associated with one routing method. The available routing methods are:

- **Unconditional**  
There is only one target destination and that must be always used.
- **Regular expression**  
The business logic provides the destination information
- **Number range/series routing**  
The business logic provides an identifier which is used by the NE type to locate the destination.

The connection management layer manages the actual connectivity towards the physical element management. Each physical NE must be defined in this layer with its respective communication particulars. One NE can serve one or more purposes in the provisioning process, thus can be associated to one or more NE types.

An NE's availability is indicated by a status indicator. The indicator can be viewed from the GUI. The NE Group is using this indicator to decide if this NE is available for traffic.

The Status indicator is automatically controlled by monitoring the NE responses and periodically sending a heartbeat request. Status is set to **Down** as soon as



a communication error is received on a request or a heartbeat. Status is set to **Up** again by the first heartbeat that does not result in a communication error.

If there are multiple instances of NEs that are providing the same function in the provisioning process, an NE group entity is introduced to manage these resources. The NE group keeps track of the availability of NEs in the group and distributes traffic based on the defined distribution algorithm.

The available distribution algorithms are:

- Failover

The NEs in the group are listed with priority. All traffics are always sent to the available NE with the highest priority. The NE group manages the fallback when NE with the highest priority is back in service.

- Round robin

All available NEs in the group are used for traffic. Round robin can be configured with weight or without weight. The weight factor is expressed as integer between 1 and 100.

For example, if there are three NEs in the group with weight factors 1, 2 and 3, the load balancing picture is:

- 16.7% (1/6) for the first NE
- 33.3% (2/6) for the second NE
- 50% (3/6) for the third NE

**Note:** In the current release, the Failover and Round robin distribution algorithms are only available for southbound adapters based on Java Connector Architecture (JCA), not for that based on Connector Framework (CF).

When an NSO needs to be sent out, the following takes place:

- The business logic invokes the corresponding NE Type.
- The NE Type applies the configured routing method for this NSO and forwards it to the corresponding NE Group (alternatively direct to NE).
- NE group decides this NSO should be sent to network.

When configuring NE management for business logic, the following needs to be performed:

- Configure routing method for the pre-loaded NE types.
- Configure NE groups if necessary.
- Configure NEs.



- Associate NEs with the group.
- Associate NEs with the NE type.

Default NE groups and its routing method are provided as XML files, which can be loaded to the system after the initial system installation. It is possible to modify the routing methods via the GUI. Such configuration is preserved in the system and will not be affected by future system updates and upgrades.

For more information about what to configure for each business logics, see *Configuration Manual for Resource Activation*, Reference [11].

## 5.3 Connection Robustness

The LDAP, Telnet, SSH, and Provisioning Notification (SOAP for HSS-FE) protocols are used for critical missions by the business logics hosted in Dynamic Activation, thus extra robustness function are introduced for these protocols.

A specific NE that is considered down from Dynamic Activation perspective is moved to a blacklist in the NE Group. Blacklisted NEs are not used in the provisioning, that is, the NE group will not use them. Dynamic Activation periodically sends heartbeats towards each NE to check its status (up/down). NEs are removed from the blacklist when a successful heartbeat discovers that it is up again. When the NE is removed from the blacklist, it can be used in the provisioning again.

Table 1 describes the robustness behavior for Telnet, and SSH protocol. It applies to NEs with protocol name `Telnet` and `SSH`.

*Table 1 Robustness Behavior for Telnet, and SSH protocol*

Error Case	Error Description	Network Element Behavior	Network Element Group Behavior
Function Busy	The NE is unable to process the client request, currently, but if the client waits and resubmits the request, the NE could be able to process it.	Retry the operation on the same connection.  Retry is done configurable number of times. The delay between retries is configurable with parameter <code>Retry Time Interval</code> .  If all retries fail, the NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.
Broken Connection	Temporary or recoverable connection faults towards the NE: <ul style="list-style-type: none"><li>• The connection is dropped because of firewall idles time-out.</li><li>• The connection is dropped because of NE restart.</li></ul>	One retry is performed to establish the same connection after a delay. The delay is configurable with parameter <code>Long Retry Time Interval</code> .  If it fails, the NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.



Error Case	Error Description	Network Element Behavior	Network Element Group Behavior
Response time-out	NE does not respond to the client within the expected duration.  Another NE in the group could answer the request.	No retry on NE level, the NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.
Other Connection Faults	Other persistent connection-related fault towards the NE.  Another NE in the group could answer the request.	No retry on NE level, the NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.
Persistent Faults	Persistent fault with the NE, such as Central Processor Not Obtainable.  Another NE in the group could answer the request.	No retry on NE level, the NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.

Table 2 describes the robustness behavior for SSH protocol, based on Connector Framework. It applies to NEs with protocol name CF-SSH.

*Table 2 Robustness Behavior for SSH Protocol based on Connector Framework*

Error Case	Error Description	Network Element Behavior	Network Element Group Behavior
Broken Connection	Temporary or recoverable connection faults towards the NE: <ul style="list-style-type: none"> <li>The connection is dropped because of firewall idles time-out.</li> <li>The connection is dropped because of NE restart</li> </ul>	If it fails, the NE is blacklisted.	Not supported
Connection time-out	NE does not respond to the client within the expected duration.  Connection is still alive.	If it fails, the NE is blacklisted	Not supported

Table 3 describes the robustness behavior for LDAP.

*Table 3 Robustness Behavior for LDAP*

Error Case	Error Description	Network Element Behavior	Network Element Group Behavior
LDAP Timelimit Exceeded (3)	The operation's time limit specified by either the client or the server has been exceeded.	Retry the operation on the same connection.  Retry is done configurable number of times. The delay between retries is configurable with parameter Retry Time Interval.  If all retries fail, the NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.



Error Case	Error Description	Network Element Behavior	Network Element Group Behavior
Administrative limit exceeded (11)	The NE is unable to process the client's request, currently, because of a user or system specified limit, but if the client waits and resubmits the request, the NE might be able to process it.	Retry the operation on the same connection.  Retry is done configurable number of times. The delay between retries is configurable with parameter Retry Time Interval.  If all retries fail, the NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.
LDAP Busy (51)	The NE is unable to process the client's request, currently but if the client waits and resubmits the request, the NE might be able to process it.	Retry the operation on the same connection.  Retry is done configurable number of times. The delay between retries is configurable with parameter Retry Time Interval.  If all retries fail, the NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.
LDAP Unavailable (52)	The NE is unable to process the client's request, currently, because of a persistent error. Another NE in the group could answer the request.	No retry on NE level, the NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.
LDAP Unwilling to perform (53)	The NE is unable to process the client's request, currently, but if the client waits and resubmits the request, the NE might be able to process it. The retry must only be done for some of the suberrors.	Retry is done configurable number of times. The delay between retries is configurable with parameter Long Retry Time Interval.  The NE is never blacklisted.	No retry towards other NE in the group.
LDAP Other (80)	The NE is unable to process the client's request, currently, because of an internal error, but if the client waits and resubmits the request, the NE might be able to process it.	Retry the operation on the same connection.  Retry is done configurable number of times. The delay between retries is configurable with parameter Retry Time Interval.  If all retries fail, the NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.
Response time-out	NE does not respond to the client within the expected duration.  Another NE in the group could answer the request.	No retry on NE level, the NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.
Other Connection Faults	Other faults occurred because of connectivity error when communicating with the NE.  Another NE in the group could answer the request.	One retry is performed to establish the same connection immediately.  If it fails, the NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.





**Table 4 Robustness Behavior for Provisioning Notification (SOAP for HSS-FE)**

Error Case	Error Description	Network Element Behavior	Network Element Group Behavior
Communication Failure	The NE is unable to process the client request, such as time out fault, other web service faults, and so on. Currently, if the client waits and resubmits the request, the NE can process it.	Retry the operation on the same connection.  Retry is done within the times of a configurable number. The delay between retries is configurable with parameter <i>Retry Time Interval</i> .  If all retries fail, the NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.
Server Fault	SOAP server fault, such as Function Busy error.  Another NE in the group can answer the request.	No retry on NE level.  The NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.
Client Fault	SOAP client fault, such as the wrong format of SOAP notification request.	No retry on NE level.  No NE is blacklisted.	No retry on NE group level.
Other Connection Faults	Other persistent connection-related fault towards the NE.  Another NE in the group can answer the request.	No retry on NE level.  The NE is blacklisted.	The NE Group resends the same operation towards the next not blacklisted NE in the group.

For NE Retry, the following behavior can be defined via the GUI:

- Number of Retries (LDAP, Telnet, SSH, CF-SSH) / Retry Attempts (Provisioning Notification)

How many retry attempts are reasonable depends on what behavior the client prefers. If the client prefers to get response about the error as soon as possible, the number of retries are few or no retry at all. If the client prefers to let Dynamic Activation try to overcome the error, the number of retries are as many as possible. For CAI3G, the total retry duration should not exceed inbound interface time-out duration, which is three minutes. This is not applicable for MML and CLI.

- Retry Time Interval (LDAP) / Function Busy Timer (Telnet, SSH, CF-SSH) / Retry Interval (Provisioning Notification)

The Retry Time Interval / Function Busy Timer / Retry Interval should be configured so that most of the error instances are resolved before the first retry attempt. In the example below, Table 5 and Table 6, most function busy instances (for example, 80%) disappear within 200 ms. Thus, the first retry overcomes most of the error instances.

- Long Retry Time Interval (LDAP, Telnet, SSH, CF-SSH)

The Long Retry Time Interval is used for special situations when the NE needs longer time to recover.



- Retry Factor (LDAP, Telnet, SSH, CF-SSH)

The Retry Factor increases the Retry Time Interval / Function Busy Timer. The first retry takes place after the Retry Time Interval / Function Busy Timer duration is reached. For the second retry and onward, the delay duration is increased by the Retry Factor, expressed as:

$$\text{Present Delay} = \text{Previous Delay} \times \text{Retry Factor}$$

In the example below, Table 5 and Table 6, 10% of the function busy instances can be resolved within 400 ms. These operations are successful in the second attempt. The retry delay doubles for every new retry, allowing longer time for some difficult but rare function busy instances to recover. Some error instances might need the last attempt to overcome, which means rather long duration compared to no retry. Some error instances might not be able to recover after all the retry attempts.

Table 5 and Table 6 are examples of how the retry behavior works for an NE configured as LDAP protocol.

*Table 5 Retry Behavior for an NE*

NE Retry Configuration:	
Number of retries: 3	
Retry Time Interval: 1000 ms	
Retry Factor: 2	
First retry delay	1000 ms
Second retry delay	2000 ms
Third retry delay	4000 ms

In practice, the client might experience the following latency:

*Table 6 Latency*

Percentage of Provisioning Request	Response Time	Description
99%	50 ms	The round trip time for a normal operation is assumed to be 50 ms.
0.8%	1050 ms	Function busy instances recovered before first retry.
0.1%	3050 ms	Function busy instances recovered before second retry.
0.08%	7050 ms	Function busy instances recovered before third retry.
0.02%	7050 ms	Function busy instances cannot be recovered, the operation fails.

If an operation fails on one NE, the NE Group performs retry towards another not blacklisted NE in the group. If that also fails after retries on NE level, the NE Group will retry on the next NE until the operation gets through or MAX\_SEND\_ATTEMPTS are reached.



For NE Group retry, the following behavior can be defined via updating of system properties:

- `RetryDelay`

This configuration is used as the retry delay before the first retry attempt to failover to another NE. The value should be configured so that most operations are successful with this retry.

- `RetryFactor`

For the second retry and onward, the delay duration is increased by the `RetryFactor`, expressed as:

$$\text{Present Delay} = \text{Previous Delay} \times \text{RetryFactor}$$

The `RetryFactor` is powerful especially when the network, in which the NE Group resides, experiences turbulence. It gives the network a chance to become stabilized and, thus, improve the retry successful rate.

- `MAX_SEND_ATTEMPTS`

The `MAX_SEND_ATTEMPTS` defines the maximum number of send attempts on NE Group level.

If all NEs are blacklisted, the NE Group level still applies the `RetryDelay` and `RetryFactor`, and checks if one of the NEs has got whitelisted during the delay. The total number of retries on NE Group level, either towards whitelisted NEs or towards checkups when all NEs are blacklisted, is defined by the `MAX_SEND_ATTEMPTS`.

Table 7 is an example of how the retry behavior and the `MAX_SEND_ATTEMPTS` work for NE Group.

**Table 7** *Example of Retry Behavior*

NE Group Retry Configuration: Number of NEs in the group: 3 <code>MAX_SEND_ATTEMPTS</code> : 6 Retry delay: 500 ms Retry factor: 2		
NE 1 fails and gets blacklisted First retry delay (towards NE 2)	500 ms	Number of send attempts 1
NE 2 fails and gets blacklisted Second retry delay (towards NE 3)	1000 ms	Number of send attempts 2
NE 3 fails and gets blacklisted Third retry delay (before checking for whitelisted NE)	2000 ms	Number of send attempts 3



All NEs blacklisted Fourth retry delay (before checking for whitelisted NE)	4000 ms	Number of send attempts 4
All NEs blacklisted Fifth retry delay (before checking for whitelisted NE)	8000 ms	Number of send attempts 5
All NEs blacklisted No more retries, MAX_SEND_ATTEMPTS reached		Number of send attempts 6

Dynamic Activation executes provisioning requests in parallel. Each of the executions detects and blacklist unhealthy NEs, which means that situations when retries on all NEs are required would rarely happen.

NE Group retry works even if direct routing is used. This means that if routing directly towards a NE instead of a group, second-level retry behavior is still triggered. It is possible to disable second-level retry but this is done globally for all configured NEs. For more information on how to configure the retry behavior and second-level retry, refer to *Configuration Manual for Resource Activation*, Reference [11].

## 6 Authorization and Access Control

Dynamic Activation provides authorization framework for defining user access to the application function as well as provisioning clients for the underlying network resources. The Access Control function enforces authorization when user uses the application function.

A GUI user can either have a `System User` or a `System Administrator` role. Both have access to all GUIs except the **User Management** GUI, which is only accessible by users with the `System Administrator` role.

For more information on how to define access control, refer to *User Guide for Resource Activation*, Reference [10].

### 6.1 Provisioning Access Control

Provisioning Access Control (PAC) covers all operations performed via all provisioning interfaces. A client, who is granted PAC authority, is called a provisioning client.

Authorization consists of one or more restriction rules. The restriction rules are the base for the PAC function.



Access Control allows only operations within the conditions specified in the client's restriction rules. Operations concerning conditions not specified in the restriction rules are not allowed.

By default, no condition is set for a client which means the client can perform all operations provided by the application. A GUI user have the permission to perform such restriction rule settings.

A provisioning client is granted PAC authority on two levels:

- On MO and operation level

This is performed by specifying one or more MOs a client can work with and the operation types for each MO the client is entitled to perform.

- On attribute level

Restriction Rule PAC authorities are set to limit what attributes and attribute values a client can work with.

When defining restriction rule for an attribute, the allowed value of the attribute is specified. For example, `>0 AND <10` means that values from 1 to 9 are allowed, and all other values are denied. `>10` means that all values above 10 are allowed.

A restriction rule on attribute level can include one or several conditions. The logical operators `&` (AND) and `|` (OR) can be used between conditions and each condition is an expression including `<attribute>`, `<attribute operator>`, and `<value>`.

For example, a number range rule for the attribute MSISDN can be expressed as:

```
msisdn > 491900001000 & msisdn < 491900002000
```

There are three different types of attributes and the type of an attribute decides how the rule can be defined.

The attribute types are:

- NUMERIC

For an attribute of type NUMERIC, the attribute operators `<`, `>`, and `=` can be used. The value must be a number. If multiple conditions using the attribute operator `=` are defined on the same attribute, the attribute must occur only once in the inbound request. In this case, the system does not support multiple occurrences of the same attribute in a request. However, if a range is specified for an attribute, multiple occurrences of the same attribute in a request are supported.

- REGEXP (regular expression)



For an attribute of type `REGEXP`, the attribute operator `?` can be used. The regular expression syntax that can be used is defined by the *XACML 1.1* specification, see Reference [28]. The value is a string containing the regular expression.

- `NOVALUE`

For an attribute of type `NOVALUE`, the attribute operator `!` (NOT) can be used. Using the NOT operator means that the request is not authorized if the attribute occurs in an inbound request.

For information about what to configure for a client for specific business logic, see *Configuration Manual for Resource Activation*, Reference [11].

## 7 Fault Tolerance

Fault tolerant logic is used to prevent inconsistency of subscribers when provisioning is only partly succeeded.

It is crucial to assure completeness of data when sending multiple LDAP operations to provision a subscriber. Retry attempts are performed during the retry time, and might not be successful. In those cases, the result can lead to inconsistent data for the provisioned subscriber.

The `Create` operation is fault tolerant in the way that a rollback is always automatically performed if the `Create` was not successful. If a rollback is executed, one of the following error codes are returned depending on the rollback status.

**Table 8** Description of Rollback Error Codes

Error Code	Description	Action
12013	Operation failed, rollback has been performed successfully	Resend the <code>Create</code> command.
12014	Operation failed, rollback was unsuccessful	First send the <code>Delete</code> command <sup>(1)</sup> , then send the <code>Create</code> command. <sup>(2)</sup>

(1) To make it possible to delete a partially created subscriber, one identity is used in the `Create` operation as master identity. This master identity must be used in the `Delete` command. For example, `MSISDN` is the master identity for HLR subscribers. For detail of master identity for each operation, refer to the corresponding interface specification.

(2) An event can be sent about a failing delete CUDB operation, even though the command itself was not `Delete`.



The following table holds a description of the partly provisioning error code.

**Table 9** Description on Partly Provisioning Error Code

Error Code	Description	Events
1101	External Error, includes more detailed information about the error.	Error Deleting Entry: <ul style="list-style-type: none"> <li>• 4104</li> <li>• 4206</li> <li>• 4304</li> </ul>
		Error Creating Entry: <ul style="list-style-type: none"> <li>• 4103</li> <li>• 4205</li> <li>• 4303</li> </ul>
		Error Updating Entry: <ul style="list-style-type: none"> <li>• 4105</li> <li>• 4207</li> <li>• 4305</li> </ul>

**Note:** Partly provisioned subscribers in older releases cannot be deleted. Partly provisioned subscribers must therefore be repaired before an upgrade.

The `Delete` and `Set` operations have been enhanced so that if any of them fails because of a CUDB connection error, the same command can be executed again.

The following table shows what actions the operator need to do in case of faults in the operation:

**Table 10** Actions on Failed Operations

Commands	Description	Action
Create	If the <code>create</code> request was unsuccessful and rollback was successful.	Resend the <code>Create</code> command
	If the <code>create</code> request was unsuccessful and rollback was unsuccessful. (1)	Execute corresponding <code>Delete</code> command and then the <code>Create</code> command.
Set	If the <code>Set</code> request was unsuccessful.	Resend the <code>Set</code> command.
Delete	If the <code>Delete</code> request was unsuccessful.	Resend the <code>Delete</code> command.

(1) If a rollback is unsuccessful because of that CUDB stops responding, for example, if time out on a connection occurs, the CUDB modification can still be executed and the subscriber is removed. In this case, if performing a `Delete` command, this results in an error that the subscriber does not exist.



**Note:** For IMSI Changeover, there is no rollback and the same command that failed needs to be sent again.

The following subsections describe deviations from the Fault tolerant logic.

## 7.1 Operations without Automatic Rollback Functionality

The following operations do not have automatic rollback functionality in this release:

*Table 11 Operations without Automatic Rollback Functionality*

Service	Command	Operation
AAA	AAAUser	Create
	AAAGroup	Create
	AAAPolicy	Create
EIR	Equipment	Create
	Equipments	Create
	ClonedHandsetUser	Create
	MobileStation	Create
	SearchOrder	Create

If any of the operations fail during execution, a `Delete` command must be sent manually.

## 7.2 Operations without Fault Tolerance

The following commands do not have fault tolerance functionality in this release:

*Table 12 Commands without Fault Tolerance*

Service	Command	Operation
EPS	EPSMultiSC	Set <sup>(1)</sup>
AVG	AVGMultiSC	Set
IMS	IMSAssociation	Set <sup>(1)</sup>
AAA	AAAUser	Set
		Delete
	AAAGroup	Set
	AAAPolicy	Set





Service	Command	Operation
EIR	Equipment	Set
		Delete
	Equipments	Delete
	ClonedHandsetUser	Set
		Delete
	MobileStation	Delete
	SearchOrder	Set
		Delete
HLR	Subscription	Set <sup>(2)</sup>
		Delete

(1) Partial fault tolerance with support for Set-Delete commands.

(2) For HLR Components. Refer to *CAI3G Interface Specification for HLR Components*, Reference [26]

If any of the commands fail during execution, refer to *CUDB Subscription Repair and Remove Procedures*, Reference [13] for more information.

## 8 Prevent Concurrent Subscriber Provisioning

Prevent Concurrent Subscriber Provisioning is able to avoid the concurrent provisioning to the same subscriber. That means if more than one requests are received on same subscriber, Dynamic Activation provisions the request that first acquires the lock and rejects other requests during the provisioning with error code 12016. This function is only impacted the `Create`, `Delete` and `Set` operation.

This function is disabled by default, and should be kept disabled when there is no cooccurring provisioning. Because enabling this function will decrease the provisioning performance. For detail information, refer to Reference [25].

*Table 13 Description of Concurrent subscriber provisioning Error Codes*

Error Code	Description	Action
12016	Concurrent subscriber modification not allowed. Please try again later.	Try again later.

Below business logic are supported by Prevent Concurrent Subscriber Provisioning feature.



- IPWorks/AAA Provisioning
- AUC Massive Provisioning
- AUC Subscriber Provisioning
- CUDB IMSI Changeover Provisioning
- CUDB MSISDN Changeover Provisioning
- HLR Massive Provisioning
- HLR Profile Provisioning
- HLR Subscriber Provisioning
- HSS Provisioning AVG
- HSS Provisioning EPS
- HSS Provisioning IMS
- SAPC Provisioning
- VoLTE provisioning

For how to configure Prevent Concurrent Subscriber Provisioning on GUI, refer to *User Guide for Resource Activation*, Reference [10].

## 9 Replay Operation

Dynamic Activation supports replay operations towards CUDB using northbound CSOs and southbound LDAP NSOs, recorded in the processing log. Start and end- time stamps are used to limit the replay time span of a replay operation, where also, a list of `RootLogId(s)` can be used as filter criteria for selective replay of individual CSOs.

Replay operations will replay CSOs with `SUCCESSFUL` and `FAILED` state, and its corresponding LDAP NSOs: `LDAP Add`, `Modify` and `Delete` operations. LDAP operations with result code `success(0)` are considered, otherwise omitted, where replay operation will continue with the next NSO in sequence.

To enable continuous replay on existing LDAP entries, the following LDAP error code responses will be ignored, and replay operation will continue with the next LDAP NSO in sequence, see Table 14.



**Table 14 Ignored LDAP Error Code Responses**

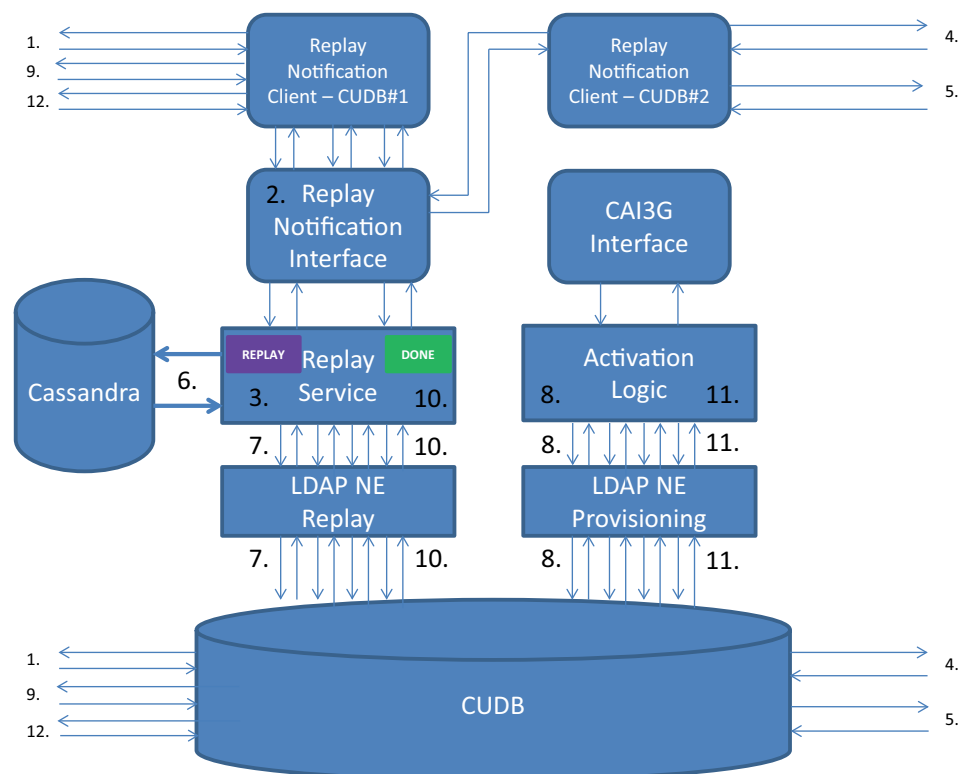
LDAP Operation	LDAP Error Code	Description
Add	68	Entry already exists.
Modify (Add)	19	A constraint violation.
Modify (Add)	20	An attribute or value already in use.
Modify (Delete)	16	No such attribute exists.
Delete	32	No such object exists.
	33	Alias problem.
	66	Subordinate objects must be deleted first

## 9.1

### Automatic Replay Operation

Automatic replay operations towards CUDB are supported by notifying Dynamic Activation with a time stamp, which indicates the starting point of a replay time span. Dynamic Activation implements an HTTP-based RESTful interface, using standard HTTP methods (POST) for notifying replay requests, and to poll (GET) the current state from Dynamic Activation during a replay operation.

Figure 11 gives an overview of an Automatic Replay operation.



**Figure 11 Overview, Automatic Replay Operation**



1. CUDB sends a replay request with a time stamp, indicating the starting point of a replay time span.
2. When a replay request is received, Dynamic Activation performs a sanity check of the time stamp provided, and also checks that a replay operation is not already ongoing, before finally sending an HTTP response header, and status code back to CUDB.

HTTP status codes defined for replay requests (HTTP POST) are listed in Table 15.

*Table 15 HTTP Status Codes, HTTP POST*

HTTP Status Code	Description
202 Accepted	Replay request accepted or indicating already received time stamp.
401 Not Authorized	Authentication failed.
403 Forbidden	Replay request rejected because of invalid time stamp or ranging too far back in time.
409 Conflict	Replay request rejected because of already ongoing replay operation.

3. If the replay request is accepted (202 Accepted), Dynamic Activation enters the `REPLAY` state, which also works as a lock for subsequent replay requests – where only one replay operation may be active at a time.
4. Subsequent replay requests, indicating the same time stamp as already received, will also be accepted, for example multiple CUDB nodes sending the same replay request, but will not be considered as a new replay operation.
5. New replay requests with a new timestamp will be rejected (409 Conflict), because of already ongoing replay operation.
6. Once in `REPLAY` state, Dynamic Activation will query the consolidated processing log, using the time stamp provided, for both southbound and northbound records, to determine if there are any provisioning operations to replay.
7. While in `REPLAY` state, Dynamic Activation will process all log records for the indicated timespan, up until the time stamp of replay request arrival.

A `REPLAY_TIME_OUT` parameter is defined for `REPLAY` state, which configures the time a replay operation may execute, before being ended, and reverted to replay state `DONE`. The `REPLAY_TIME_OUT` parameter is by default configured to 300 seconds – to provide enough head-room for a replay operation to complete, prior to time-out. This in case the provisioning rate during the indicated time span is higher than expected.

For more information about configuring Dynamic Activation for automatic replay operation, see *Configuration Manual UDC Data Durability*, Reference [15].



8. In the meantime, CUDB will block any LDAP operations originating from new or ongoing service orders.
9. At the same time, CUDB will periodically send replay poll state requests with default one second intervals – to determine when a replay operation is completed, and when to unblock normal provisioning.

HTTP status codes defined for poll state requests (HTTP GET), are listed in Table 16.

*Table 16 HTTP Status Codes, HTTP GET*

HTTP Status Code	Description
200 OK	OK.
401 Not Authorized	Authentication failed.

HTTP response header parameters and values defined for poll state requests (HTTP GET), are listed in Table 17.

*Table 17 HTTP Response Header Values, HTTP GET*

HTTP Response Header Parameter	Description
State: DONE	Default state during normal provisioning operation and when the replay operation has finished, failed, or timed out.
State: REPLAY	State during replay operation. In this state, the CUDB is blocking normal provisioning operation.

For more information about automatic provisioning block in CUDB, see Section 9.1.1 on page 35.

10. Once a replay operation is completed – either successfully, failed or timed out – Dynamic Activation returns to `DONE` state immediately, for CUDB to react and unblock the provisioning block as soon as possible For more information about Replay states, see Figure 12.

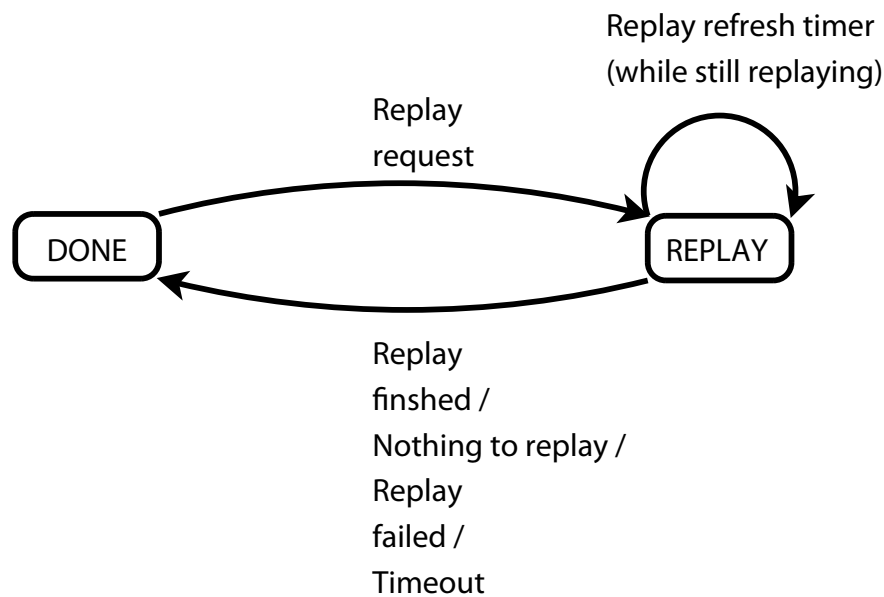


Figure 12 Replay State Diagram

11. In turn, Dynamic Activation will resume normal provisioning operation for both ongoing (retry) LDAP operations (NSOs) and new service orders (CSOs).
12. Replay requests with time stamps ranging too far back in time will be rejected immediately (403 Forbidden), whereas a trade-off between how long Dynamic Activation can keep ongoing provisioning requests pending, and replay performance must be considered. For example, how many provisioning requests were processed for the indicated time span and how long would it take to replay. In the interim, Dynamic Activation will remain in **REPLAY** state before starting the actual replay towards CUDB, or in case the replay request was deemed impossible to fulfill within a configured time span; Dynamic Activation will return to **DONE** state, and raise an alarm accordingly. Dynamic Activation will by default accept time stamps ranging 30 seconds back in time, based on the `MAX_REPLAY_TIME_SPAN` parameter.

In case of a rejected, failed or timed out replay operation – an alarm is raised – indicating the time span (start and end timestamp) that is subject to further analysis and possibly manual replay operation. An event will also be sent upon successfully completing a replay operation.

For additional details about a typical Automatic Replay sequence, see Figure 13.

For more information about Replay related alarms, see *Event and Alarm Handling*, Reference [16].

For more information about Manual Replay operation, see Section 9.2 on page 38.

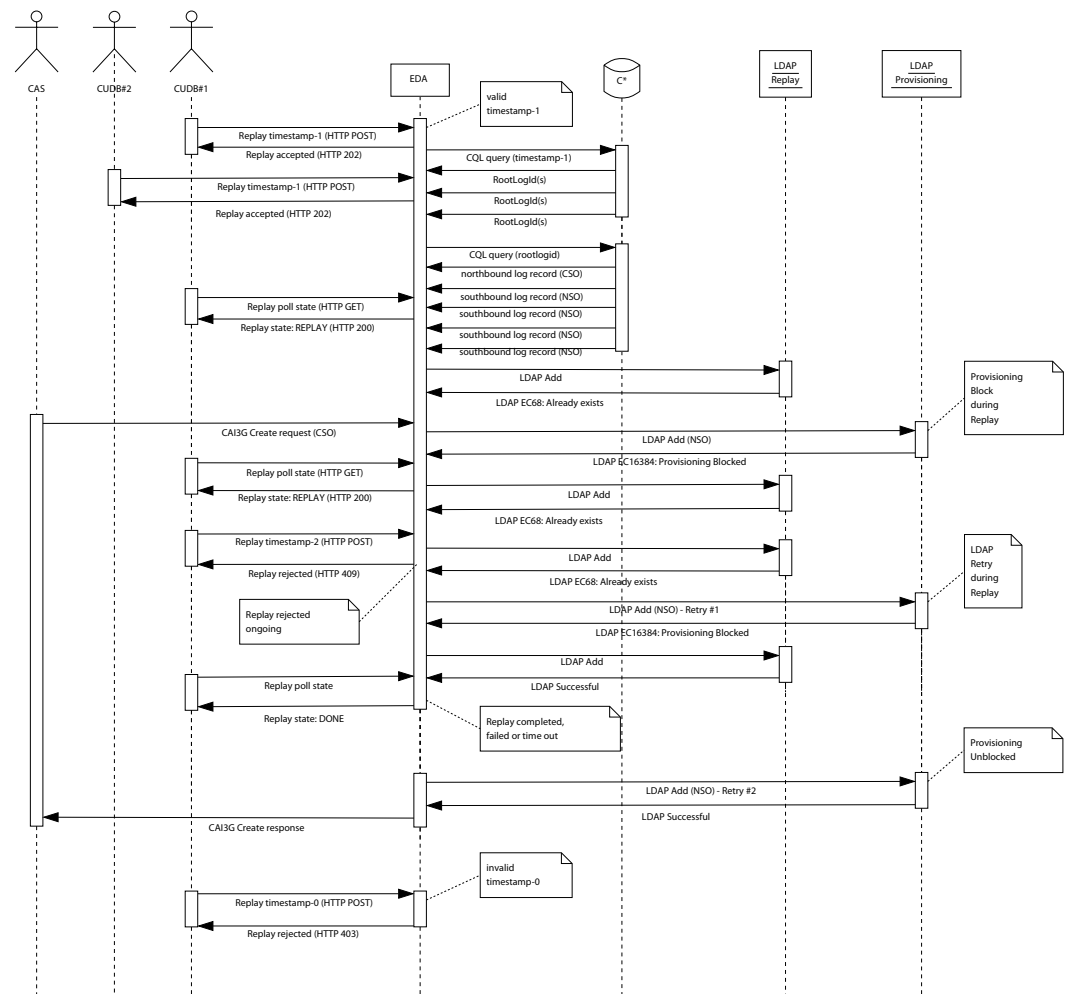


Figure 13 Automatic Replay Sequence

### 9.1.1

## Automatic Provisioning Block

Automatic replay operations are initiated and coordinated from CUDB, which upon notifying Dynamic Activation, also will block subsequent LDAP operations belonging to new and ongoing service orders. This to secure sequential ordering of LDAP operations originating from normal provisioning operations, and automatic replay operations, see Figure 14.

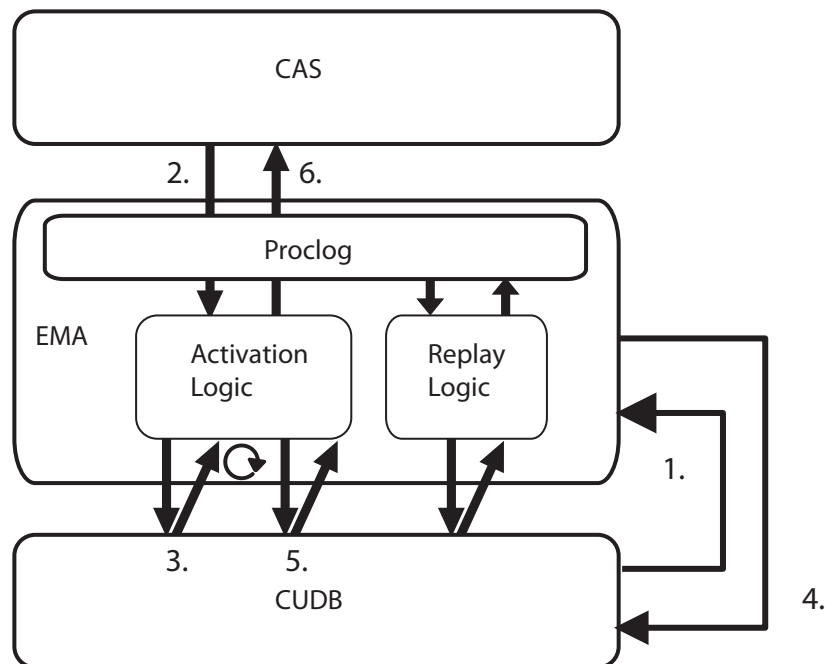


Figure 14 Automatic Replay Operations, Overview

- 1 Dynamic Activation receives REPLAY request and starts replaying from processing log.
- 2 New service orders from CAS are accepted.
- 3 LDAP operations originating from new and ongoing service orders are blocked in CUDB and resent until replay operation is completed, failed, or timed out.
- 4 Replay operation is completed, failed, or timed out.
- 5 New and ongoing service orders are unblocked in CUDB, and Dynamic Activation can resume normal provisioning operation.
- 6 New and ongoing service orders are responded back to CAS.

To distinguish LDAP operations originating from replay operations, a duplicate LDAP NE is configured for each CUDB node with a separate LDAP `Replay` user-dn. When performing an automatic replay operation, CUDB will reject LDAP operations originating from the LDAP `Provisioning` user-dn with a proprietary LDAP error code (16384) – providing exclusive access for the LDAP `Replay` user-dn.

LDAP operations rejected with LDAP error code 16384 will be resent `MAX_SEND_ATTEMPTS` number of times – before failing execution of an ongoing service order.

During normal provisioning operation, the LDAP `Replay` user-dn will be rejected with a proprietary LDAP error code (16385) – providing exclusive access for the LDAP `Provisioning` user-dn.





For more details regarding CUDB proprietary LDAP error codes, see Table 18.

*Table 18 CUDB Proprietary LDAP Error Codes*

LDAP User-dn	LDAP Error Code	Description
Provisioning	16384	Provisioning Blocking currently active. Write operations rejected for provisioning clients.
Replay	16385	Provisioning Blocking currently inactive. Write operations rejected for re-provisioning clients.

For more information about configuring Dynamic Activation for automatic replay operation, see *Configuration Manual UDC Data Durability*, Reference [15].

### 9.1.2

### RESTful Notification Interface

The Replay notification interface is an HTTPS-based RESTful implementation, which uses existing HTTP listeners for OAM (https port 8383), and also leverage on existing security (SSL/TLS), and authentication mechanisms.

HTTP Basic Authentication (BA) implementation is used for basic access authentication, where the Notification Client must provide a username and password (base64 encoded) when making a request (both for replay request and poll state requests).

The RESTful Notification Interface components are hosted by both SC nodes, where replay requests are distributed between the two in a round-robin fashion – using an IP load balancer. In turn, the replay request is sent to one of the PL nodes, that registers the Replay service, which then sets and holds a lock for Replay (distributed and synchronized with ZooKeeper), and triggers the replay operation. Replay operations may only be executed from one (single) PL node at a time. Subsequent replay requests may be distributed to any PL node, but will only be accepted if the previous replay operation is completed, or if indicating the same time stamp as for the ongoing replay operation (no action) – otherwise rejected (409 Conflict).

Poll state requests will also be distributed in a round-robin fashion, whereas the current state of a replay operation is distributed and synchronized for all nodes, and hence will respond with the same state regardless on which node the poll state request ends up with.

For more details regarding RESTful Notification Interface and client components, see Figure 15.

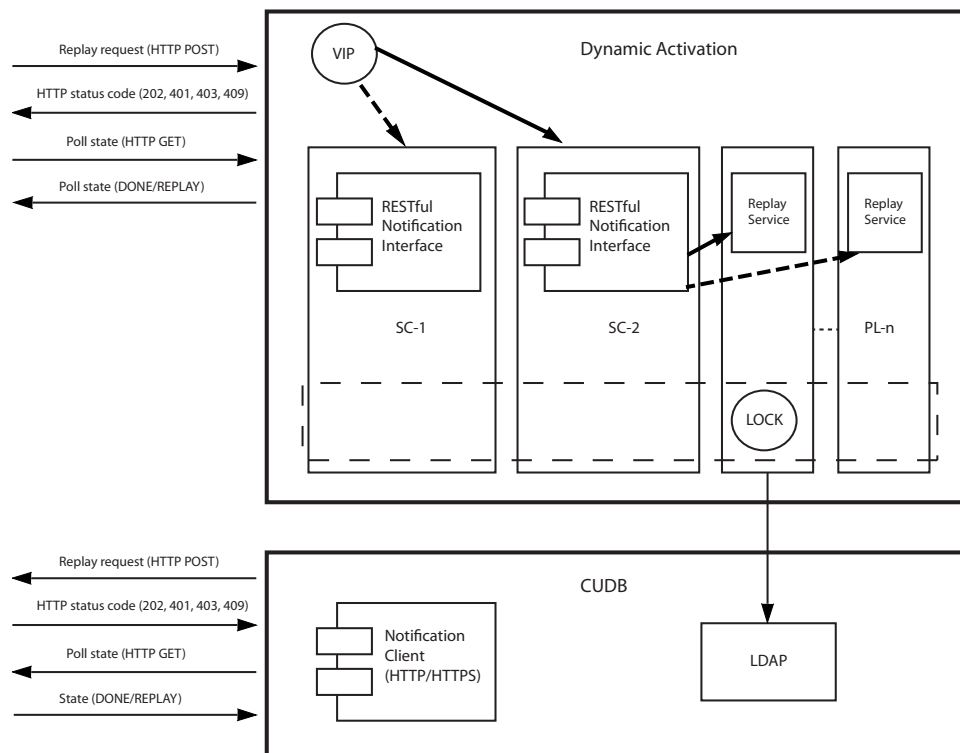


Figure 15 RESTful Notification Interface and Client Components

## 9.2 Manual Replay Operation

Manual replay operations towards CUDB are enabled with the `replay-admin-tool.sh` script, which supports replay operation based on:

- Time span using start- and end- time stamps.
- Selective replay, using list of RootLogIds

Manual replay operations are primarily intended as a means for handling failed automatic replay operations, whereas the time span that is subject for manual replay operation is indicated in alarms and events, defined for replay operations.

**Note:** Replay time-out does not apply to manual replay operations.

Provisioning operations will not be blocked in CUDB with a manual replay operation, whereas LDAP operations originating from a manual replay operation will interact with CUDB using LDAP `Provisioning` user-dn instead of LDAP `Replay` user-dn, used for automatic replay operations. Hence, it is recommended to apply northbound provisioning block on Dynamic Activation prior to a manual replay operation, to avoid race conditions between provisioning, and replay operations.

For more information about Manual Replay operation, see *System Administrators Guide for Native Deployment*, Reference [12].



## 10 Configurable Loose Error Handling

Loose Error Handling is used when an error is considered as acceptable and therefore can be ignored. This function provides the possibility to alter error handling behavior of a business logic in runtime. It is defined on per network element basis.

Loose Error Handling can be configured in GUI by the System Integrator who has in-depth knowledge of the business logic. When a loose error rule is configured, the error response that matches the rule will be considered as a successful response.

Loose Error Handling can be configured for the following standard activation logics:

- JDV-ENUM-Monolithic-Resource-Provisioning
- JDV-FNR-Monolithic-Subscriber-Provisioning
- JDV-HLR-Monolithic-Subscriber-Provisioning

For how to configure loose error rules, see *User Guide for Resource Activation*, Reference [10].

For how to specify looseable errors and pre-defined rules which are used to display on Loose Error Handling GUI, see *Customer Adaptation Development Guide for Resource Activation* Reference [14].

## 11 CAI3G Distributed Configuration

CAI3G Distributed Configuration (DC) manages the request distribution towards remote CAI3G server such as Dynamic Activation, Multi Activation, Classic Multi Activation or CAI3G compliant network elements. This feature enables Dynamic Activation to provide a unique northbound interface to BSS. When a request has been sent to Dynamic Activation through the inbound interface, this request is able to be distributed to remote CAI3G server according to the CAI3G DC Routing configuration.

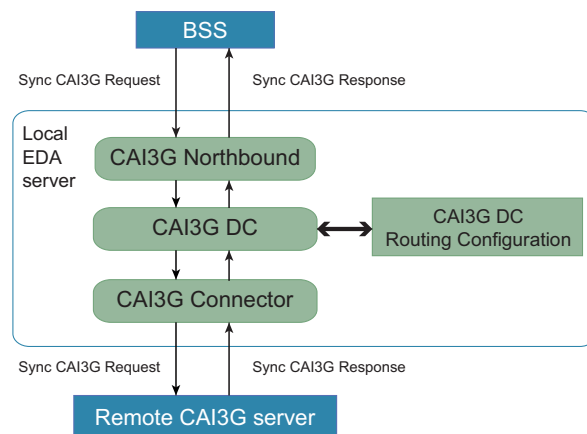
## 11.1 Request Distribution

The CAI3G DC supports requests sent by the following types of interface and each type has its own workflow:

- The northbound CAI3G request sent over the Synchronous CAI3G 1.2 interface, see Section 11.1.1 on page 40 for its workflow:
- The synchronous request sent over a synchronous interface which has the customized Subscriber View (SV) business logics deployed on the local Dynamic Activation server, such as the standard CAI3G, CAI or other supported NBIA requests, see Section 11.1.2 on page 40 for its workflow.

### 11.1.1 Distributing a Northbound Synchronous CAI3G Request

The Workflow of distributing a northbound synchronous CAI3G request is :



**Figure 16** Workflow of distributing a Northbound Synchronous CAI3G Request

1. When receiving a synchronous CAI3G request from northbound interface, Dynamic Activation converts the request into a CAI3G payload.
2. CAI3G DC looks up the destination of the CAI3G payload according to CAI3G DC routing configuration and sends this CAI3G payload to the target remote CAI3G server through CAI3G Connector.
3. Local Dynamic Activation server receives a response from remote CAI3G server and sends back to BSS through CAI3G DC.

### 11.1.2 Distributing a Synchronous Request through SV

The workflow of distributing a synchronous Request is:

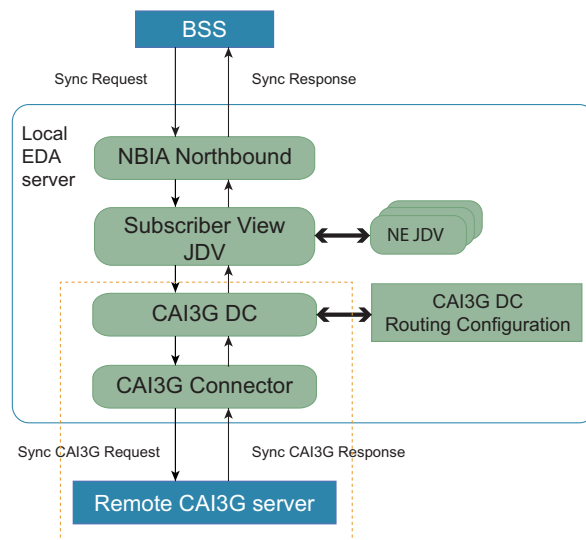


Figure 17 Workflow of distributing a synchronous Request

1. When Dynamic Activation receives a synchronous request from northbound interface, SV handles this request in Dynamic Activation.
2. SV generates a CAI3G payload which will be distributed to remote CAI3G server.
3. CAI3G DC looks up the destination of the CAI3G payload according to CAI3G DC routing configuration and sends this CAI3G payload to the target remote CAI3G server through CAI3G Connector.
4. Local Dynamic Activation server receives a response from remote CAI3G server and sends back to SV through CAI3G DC.
5. SV handles this response and generates a synchronous response back to BSS.

## 11.2 Routing Configuration

A Dynamic Activation CAI3G DC Routing Configuration contains :

- **MO Type** - The key parameter of a CAI3G DC routing to identify the request.
- **MO ID** - CAI3G DC Routing Methods are supported upon MO ID in the northbound CAI3G request.
- **Routing Method** - The supported routing methods are:
  - Unconditional
  - Number Range
  - Number Series
  - Regular Expression



- **Remote CAI3G server** - The remote CAI3G server is created as a regular Network Element with CAI3G protocol on GUI.

**Note:** NE Group, NE Cluster, NE Loose Error Handling are not supported.

For more detail information, refer to *User Guide for Resource Activation*, Reference [24].

## 11.3 Faults or Errors

CAI3G faults have the following two scenarios:

- For Synchronous CAI3G request distribution, the CAI3G fault response received from remote CAI3G server is sent back to BSS without any modification.

**Note:** In the local CAI3G request log such as Processing Log, operator can check the source server (either the local CAI3G server or the remote CAI3G server) of a CAI3G fault response.

- For Synchronous request distribution through SV, the CAI3G fault response received from remote CAI3G server is wrapped into a special `JavaDataViewException`. And this `JavaDataViewException` is handled in Subscriber View JDV. For detailed information about JDV and SV, refer to *Customer Adaptation Development Guide for Resource Activation*, Reference [14].

## 12 License Management

This section describes the Dynamic Activation license capacity, alerts, and notifications.

For information on the license architecture, refer to *Function Specification Dynamic Activation Execution Environment*, Reference [3].

### 12.1 Enforcement of Subscriber Licensing

This section contains information on how Dynamic Activation reacts when the system exceeds its defined capacity.

If the use of features exceeds the defined capacity for a value package, the system reacts.



If the subscriber capacity level is exceeded for a specified time, creation of new subscribers is disabled on all features within the appropriate value package.

The details of the functionality are as follows:

- When 80% of the licensed subscriber base is used, an event is triggered, an email is sent to the notification email address, and the GUI is updated.
- When 90% of the licensed subscriber base is used, an event is triggered, an email is sent to the notification email address, and the GUI is updated.
- When 100% of the licensed subscriber base is used, an alarm is triggered, an email is sent to the notification email address, and the GUI is updated.
- When 105% of the licensed capacity is used, duration time is measured. When the duration time exceeds 60 days, all `create` requests are stopped. Once a day, an event is triggered and an email is sent to the notification email address informing that 105% of capacity has been exceeded for more than 60 days.

If the capacity drops below 100%, the duration time measurement is stopped. If the used capacity, again, exceeds 105% of available capacity, the duration time measurement restarts from 0 days.

- When 125% of the licensed capacity is used, all `create` requests are stopped. Every hour an event is triggered and an email is sent to the notification email address informing that 125% of capacity has been exceeded.

For information on how to configure the notification email address, see *User Guide for Resource Activation*, Reference [10].

**Note:** The capacity event levels are not configurable.

## 12.2 Expiration Notifications

All licenses have their individual expiration date.

When the expiration time for a license reaches certain thresholds, an event is triggered and an email is sent to the notification email address. The email includes information of the time left before the license expires.

The threshold values are: 365, 270, 180, 90, 30, 15, 10, 5, 2 and 1 day left before license expiration.

## 12.3 Blocked Provisioning

In the service activation services, a new provisioning gets blocked in one of the following occasions:

- License used capacity is too high, see Section 12.1 on page 42.



- License has expired, see Section 12.2 on page 43.

When provisioning is blocked, it is not possible to create new subscribers.

If provisioning is blocked because of that too much capacity is used, a renewal of license capacity is needed or number of used capacity needs to be decreased. This can be done by deleting subscribers.

If provisioning is blocked because of expired license, new license needs to be ordered. See *System Administrators Guide for Native Deployment*, Reference [12].





## Reference List

### Ericsson Documents

- [1] *Library Overview*, 18/1553-CSH 109 628 Uen
- [2] *Product Overview*, 1550-CSH 109 628 Uen
- [3] *Function Specification Dynamic Activation Execution Environment*, 6/155 17-CSH 109 628 Uen
- [4] *Function Specification Layered HLR*, 4/155 17-CSH 109 628 Uen
- [5] *Function Specification Layered IMS*, 18/155 17-CSH 109 628 Uen
- [6] *Function Specification Layered LTE EPC*, 5/155 17-CSH 109 628 Uen
- [7] *Function Specification Layered EIR*, 7/155 17-CSH 109 628 Uen
- [8] *Function Specification Layered IPWorks/AAA*, 8/155 17-CSH 109 628 Uen
- [9] *Generic CLI Interface Specification*, 15/155 19-CSH 109 628 Uen
- [10] *User Guide for Resource Activation*, 1/1553-CSH 109 628 Uen
- [11] *Configuration Manual for Resource Activation*, 2/1543-CSH 109 628 Uen
- [12] *System Administrators Guide for Native Deployment*, 1/1543-CSH 109 628 Uen
- [13] *CUIDB Subscription Repair and Remove Procedures*, 4/1553-CSH 109 628 Uen
- [14] *Customer Adaptation Development Guide for Resource Activation*, 5/1553-CSH 109 628 Uen
- [15] *Configuration Manual UDC Data Durability*, 4/1543-2/CRH109 1438 Uen
- [16] *Event and Alarm Handling*, 3/1553-CSH 109 628 Uen
- [17] *DUP Customer Adaptation Migration Guide*, 19/1553-CSH 109 628 Uen
- [18] *Function Specification BCE*, 17/155 17-CSH 109 628 Uen
- [19] *Function Specification PGM*, 11/155 17-CSH 109 628 Uen
- [20] *Function Specification IPWorks/ENUM*, 20/155 17-CSH 109 628 Uen
- [21] *Function Specification MTAS*, 16/155 17-CSH 109 628 Uen



- [22] *Generic EDIFACT Interface Specification*, 3/155 19-CSH 109 628 Uen
- [23] *Solution Description Charging and CBiO*, 1/221 02-CSH 109 628 Uen
- [24] *User Guide for Resource Activation*, 1/1553-CSH 109 628 Uen
- [25] *Characteristics Description*, 3/192 02-FGC 101 3061 Uen
- [26] *CAI3G Interface Specification for HLR Components*, 25/155 19-CSH 109 628 Uen
- [27] *Function Specification SAPC*, 9/155 17-CSH 109 628 Uen

#### **Other References**

- [28] *XACML 1.1*, [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abrev=xacml](http://www.oasis-open.org/committees/tc_home.php?wg_abrev=xacml)