

VoLTE Provisioning Customer Adaptation Guide

Ericsson Dynamic Activation 1

USER GUIDE

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

1	Introduction	1
1.1	Purpose and Scope	1
1.2	Target Groups	1
1.3	Typographic Conventions	1
1.4	Prerequisites	1
2	VoLTE Service Model Overview	3
2.1	VoLTE Decision Service Model	3
2.2	VoLTE Decision Service Model Design Base Projects	3
2.3	VoLTE Execution Service Model	3
2.4	VoLTE Execution Service Model Design Base Project	5
2.5	VoLTE Basic Services Data Model	6
2.5.1	IMS	6
2.5.2	MMTel	9
2.5.3	ENUM	10
2.5.4	EPS	11
2.5.5	SAPC	11
2.5.6	HLR	12
3	Preparing for VoLTE Service Model Development	13
3.1	Preparing for CA Development Environment	13
3.2	VoLTE Decision Service Model Development Workflow	13
3.3	VoLTE Execution Service Model Development Workflow	14
4	VoLTE Service Model Development	15
4.1	Analyzing the VoLTE Service Model Application	15
4.2	Defining the VoLTE Service Model	15
4.3	Defining the VoLTE Service Model Access Control Model	16
4.4	Initializing the VoLTE Service Model Project	16
4.4.1	Importing the VoLTE Service Model Design Base Project	16
4.4.2	Renaming the VoLTE Service Model Design Base Project (Optional)	16
4.5	Handling the VoLTE Decision Service Model Business Operations	17
4.5.1	Handling CAI3G Data Validation	17
4.5.2	Handling VoLTE Auto Provisioning Criteria	17
4.5.3	Handling Concurrent Requests	17
4.5.4	Mapping Default Service Profiles	18
4.5.5	Mapping Error Codes	19



4.5.6	Integrating with UPG (Optional)	19
4.5.7	Notifying Results	19
4.6	Handling the VoLTE Execution Service Model Business Operations	21
4.6.1	Handling CAI3G Data Validation	21
4.6.2	Mapping Request Data	21
4.6.3	Handling Exceptions	22
4.6.4	Handling Data Inconsistency	22
5	Building and Deployment	25
5.1	Deploying DPHelper JDV	25
5.1.1	Preparing for CA Package	25
5.1.2	Installing DPHelper JDV	25
5.1.3	Uninstalling DPHelper JDV	26
5.2	Deploying and Undeploying a CA Service Model	26
6	Configuring Dynamic Activation for VoLTE Provisioning	27
6.1	Configuring NE in Dynamic Activation	27
6.2	Configuring Notification Endpoint in CUDB	27
7	Testing and Provisioning over CAI3G	29
7.1	Testing Tools for Sending Requests – SoapUI	29
	Reference List	31



1 Introduction

This document describes the processes of developing VoLTE Provisioning Customizations, by use of the VoLTE service model templates.

1.1 Purpose and Scope

This document is intended for the Customer Adaptation (CA) developers who develop service model Business Logic (BL) with Designer Studio, targeting VoLTE provisioning solution in Data Layered Architecture (DLA).

For information about the VoLTE provisioning solution in DLA, see *Solution Description VoLTE*, Reference [2].

1.2 Target Groups

The target groups for this document are as follows:

- CA developers
- Solution architects

1.3 Typographic Conventions

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

1.4 Prerequisites

The readers of this document must meet the following prerequisites:

- Knowledge of Dynamic Activation
- Knowledge of Linux
- Knowledge of Java, Eclipse, and Maven.





2 VoLTE Service Model Overview

2.1 VoLTE Decision Service Model

The VoLTE decision service model is a notification layer referring to the capabilities of notification handling, targeting VoLTE auto provisioning in DLA. It provides a single asynchronous CAI3G interface towards Core Network (CN) or any entities. The procedure of VoLTE auto provisioning subscriptions is as follows:

1. Make decisions.
2. Handle concurrent requests for the same subscriber.
3. Map the default service profiles to the execution service model.
4. Notify the results.

The VoLTE decision service model initiates each subscription with the default service profiles as template. For information about the default service profiles, see Section 2.5 on page 6.

2.2 VoLTE Decision Service Model Design Base Projects

The CA design base project, `VOLTEAutoProvisioning_DP-<version>.tar.gz`, supports the following features as a template for the VoLTE decision service model. Customers can follow this guide to extend the service model, for full support of the VoLTE auto provisioning criteria.

- Schema validation of the service model request.
- VoLTE auto provisioning criteria validation
- Concurrent request handling
- The default service profiles mapping to the execution service model
- Error code mapping
- Notification result
- Decision service model test code

2.3 VoLTE Execution Service Model

The VoLTE execution service model is an abstract layer referring to the capabilities of the underlying resource features of the product, targeting VoLTE

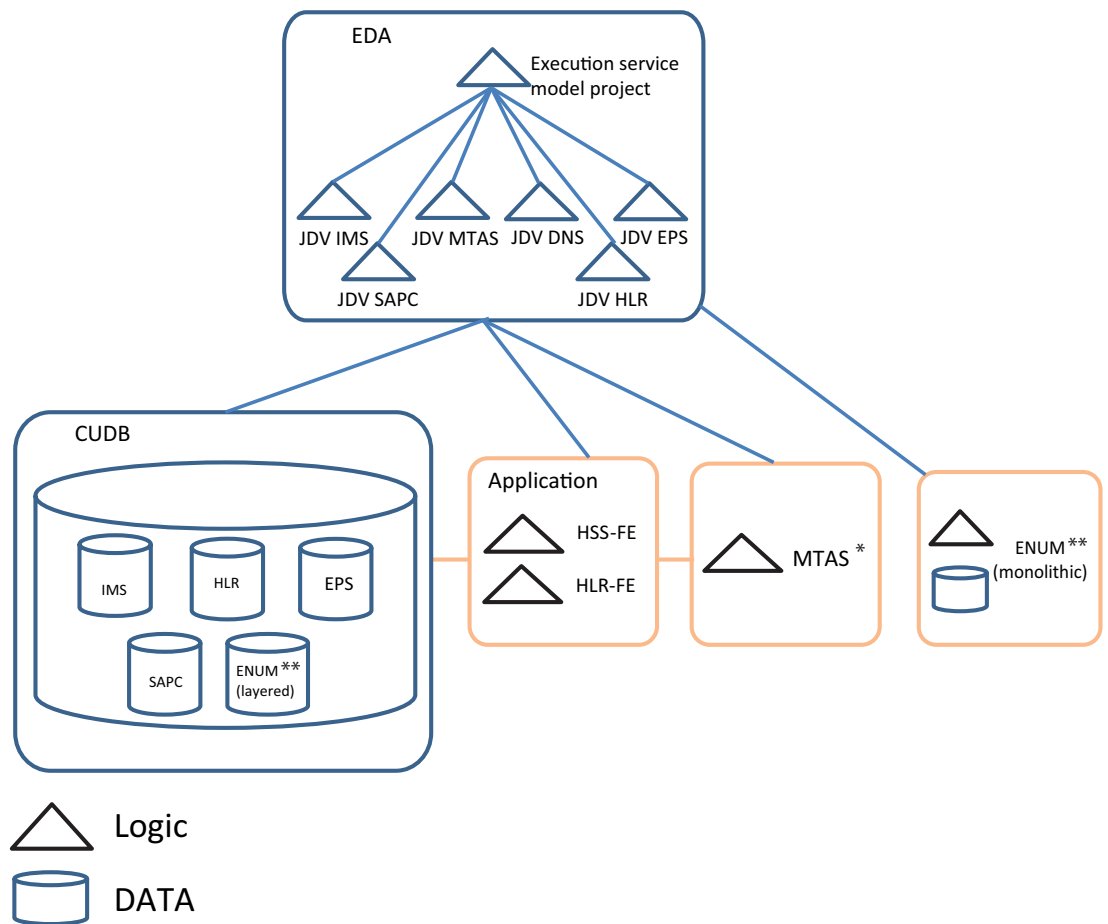
fundamental provisioning in DLA. It provides a single synchronous CAI3G interface towards Business Support System (BSS) or any entities. The VoLTE provisioning subscriptions are as follows:

- IP Multimedia Core Network Subsystem (IMS) subscription in Home Subscriber Server (HSS)
- IMS-based Multimedia Telephony (MMTel) subscription in Multimedia Telephony Application Server (MTAS)
- ENUM subscription in IPWorks
- Evolved Packet System/Long Term Evolution (EPS/LTE) subscription in HSS
- Optional, EPC subscription in Service Aware Policy Controller (SAPC)
- Optional, 2G/3G subscription in the Home Location Register (HLR)
- Optional, VoLTE service data synchronization from HLR to MTAS

The VoLTE execution service model initiates each subscription with basic VoLTE features as template. For information about the basic VoLTE features, see Section 2.5 on page 6.

The VoLTE execution service model also supports rollback to maintain data consistency in the network.

Figure 1 shows a typical DLA application deployment with VoLTE execution service model on top.



* MTAS data is stored in CUDB, through HSS-FE, as Sh transparent data.

** EDA can be configured to provision either layered or monolithic IPWorks/EMUM, not both of them.

Figure 1 VoLTE Fundamental Provisioning with VoLTE Execution Service Model Template

2.4

VoLTE Execution Service Model Design Base Project

The CA design base project, `VOLTEAutoProvisioning_EP-<version>.tar.gz`, supports the following features as a template for the VoLTE execution service model. Customers can follow this guide to extend the service model, for full support of the VoLTE subscription features.

- Schema validation of the service model request
- Service mapping from service model request to the sub applications request
- Service mapping from HLR to MTAS
- Error handling, including rollback



- Execution service model test code

2.5 VoLTE Basic Services Data Model

The VoLTE execution service model CA design base projects support a composed data model for: IMS, HLR, EPS, SAPC, DNS, and MTAS.

The data model is implemented in CUDB, and the communication with CUDB is handled by:

- JDV IMS
- JDV HLR
- JDV EPS
- JDV SAPC
- JDV DNS
- JDV MTAS

Note: JDV MTAS is responsible to send MMTel request to MTAS, which stores the service data in CUDB, through HSS-FE, as Sh transparent Data.

The VoLTE execution service model consolidates the data model for exposure towards BSS and the VoLTE decision service model, for the VoLTE fundamental provisioning.

The following subsections show the basic VoLTE provisioning services data model in each function. For more information on the data model, refer to *Solution Description VoLTE*, Reference [2].

2.5.1 IMS

Table 1 IMS Parameters

Parameter	Description
associationId	Contains the identifier of the IMS Association.
chargingProfId	Identifies the common Charging Profile configured in the HSS-FE. Default value is DefaultChargingProfile Common Charging Profile must be preconfigured in the HSS-FE.



Parameter	Description
chargingId	The MSISDN to use for charging purposes.
isPsi	Indicates if the IMS users belonging to this subscriber are Public Service Identities, either Distinct or Wildcarded. Default value is <code>false</code>
privateUserId	Identifies the IMS PrivateUser with IMPI in NAI format. For USIM user, it is derived from the IMSI as specified in TS 23.003 (that is, <code><IMSI>@ims.mnc<MNC>.mcc<MCC>.3gpp network.org</code>). For ISIM user, use the same format as for the USIM user. This, to ensure that the UE always use the same BSF address and XCAP root URI.
userPassword	Used for authentication service purposes.
secondPrivateUserId	Identifies an extra IMPI associated to the multiServiceConsumer, a second private user Id used for ICS. This IMPI must be derived from IMSI. If privateUserId is derived or userIMSI is provided, or both, the identified IMSI must be the same in all attributes. If the IMSI is defined for another subscriber, it is not allowed.
allowedAuthMechanism	Contains a list of authentication mechanisms supported by the multiServiceConsumer (Private User). Default value is <code>DIGEST</code>
userBarringId	Indicates if the multiServiceConsumer (Private User) is barred or not. Default value is <code>false</code>
userImsi	Contains the IMSI associated to the PrivateUser. If IMPI is not derived, it must be added with the <code>Create</code> operation.



Parameter	Description
msisdn	The Mobile Subscriber ISDN Number assigned to a <code>PrivateUser</code> for charging purpose.
publicIdValue	<p>Identifies the IMS Public Identity with IMPU in SIP URL or TEL URL format.</p> <p>The following IMPUs are required for both USIM and ISIM users:</p> <ul style="list-style-type: none">• sip:<MSISDN>@ims.mncXX.mccXX.3gppnetwork.org" - used for call session establishment and traffic.• tel:<MSISDN> - used for call session establishment and traffic.• sip:<IMSI>@ims.mnc<MNC>.mcc<MCC>.3gppnetwork.org - used for registrations and barred for traffic.
xcapAllowed	<p>States if the Public Identity can be used (also) in XCAP authentication.</p> <p>Default value is <code>false</code></p> <p>If the IMPU is a PSI or if it is a wildcarded public identity, the XCAP authentication must be set to <code>false</code>.</p>
xcapPassword	Used for XCAP authentication service purposes.
implicitRegSet	<p>Identifies the Implicit Registration Set (IRS), which the Public Identity belongs to.</p> <p>The first defined Public Identity of an IRS, automatically becomes the default Public Identity.</p>
isDefault	<p>Indicates if the Public Identity is the default of its IRS.</p> <p>When an IMPU of a specific type is set to default, the former one of the same types is unmarked automatically as such. Only one IMPU of each type (SIP URI or TEL URL) can be defined as default in a set. The first IMPU of an IRS must be marked as default.</p>
serviceProfileId	Serves as a reference to identify the Subscriber Service Profile assigned for the specific Public Identity.



Parameter	Description
sessionBarringInd	Indicates if the Public Identity is barred for session establishment. Default value is <code>false</code>
priorityLevel	Indicates the service priority level for the IMS Public Identity (IMPU).
subscriberServiceProfile	A list of subscriber service profiles associated to a Public Identity.
configuredServiceProfile	Contains a list of IMS common Service Profiles in the HSS-FE associated to a specific Subscriber Service Profile. Common Service Profiles must be preconfigured in the HSS-FE, including VoLTE relevant iFCs for MTAS/SCC-AS, ICS and SRVCC.
subscribedMediaProfile	Used to identify the set of session description parameters that the <code>multiServiceConsumer</code> (Private User) is authorized to request.
individualServiceProfile	Contains a list of IMS individual Service Profiles associated to a specific subscriber service profile.

2.5.2

MMTel

Table 2 MMTel Parameters

Parameter	Description
publicIdValue	Identifies MMTel user with IMPU in SIP URL format. This identity must already be provisioned in the CUDB and the default Public Identity in the ISR must be used.



Parameter	Description
service-profile-identity	<p>Serves as a reference to identify the MMTel common Service Profile in MTAS that a user is linked to.</p> <p>The value must be of the format SIP URI (RFC 3261).</p> <p>Common Service Profile must be preprovisioned in MTAS.</p> <p>If the <code>service-profile-identity</code> element contains an empty string, the semantic is to remove the service profile link.</p>
mmtel-charging-profile	<p>Serves as a reference to identify the MMTel common Charging Profile in MTAS to be used.</p> <p>Common Charging Profile must be preconfigured in MTAS if the <code>mmtel-charging-profile</code> element is set. If the element is not set, the default Charging Profile is used.</p>

2.5.3

ENUM

Table 3 ENUM Parameters

Parameter	Description
msisdn	<p>The IMS Public Identity with IMPU in TEL URL format.</p> <p>The value is the ENUM⁽¹⁾ number mapping from <code>msisdn</code>.</p>
publicId	<p>Identifies the IMS Public Identity with IMPU in SIP URL format.</p> <p>The value is the ENUM number mapping to <code>publicId</code>.</p>
flags	<p>One or two characters string, to represent the flags.</p> <p>Valid values are <code>n</code>, <code>nu</code>, <code>r</code>, <code>d</code>, or <code>c</code>.</p>
order	<p>An unsigned integer to represent the order type.</p>



Parameter	Description
preference	An unsigned integer to represent the preference type.
service	A string to identify the service.

(1) Provides E164 number translation to SIP URI

2.5.4

EPS

Table 4 EPS Parameters

Parameter	Description
imsi	Identifies the EPS user.
msisdn	The Mobile Subscriber ISDN Number assigned to the user for charging purpose.
epsProfileId	Serves as a reference to identify the EPS common Service Profile in HSS-FE assigned to the user. Common Service Profile must be preconfigured in the HSS-FE.
epsSessionTransferNumber	Indicates the Session Transfer Number (STN) in international number format for Single Radio Voice Call Continuity (STN-SR), which is used for SRVCC call transfer to be downloaded to the MME.
epsMultimediaPriorityService	Indicates whether the user is subscribed to the priority service in the EPC network.

2.5.5

SAPC

Table 5 SAPC Parameters

Parameter	Description
pcSubscriberId	Identifies the SAPC subscriber.
pcGroup	Contains a list of SAPC common Groups in the SAPC-FE, assigned to the subscriber. Common Groups must be preconfigured in the SAPC-FE.



Parameter	Description
pcSubscriberQualificationData	Contains a list of SAPC common Service Profiles, including QoS Profile, Charging Profile, Bearer QoS Profile, and General Profile. Common Service Profiles must be preconfigured in the SAPC-FE.
pcMpsProfileName	Serves as a reference to identify the SAPC common Service Profile for Multimedia Priority Services to use. Common Service Profile must be preconfigured in the SAPC-FE.

2.5.6

HLR

Table 6 HLR Parameters

Parameter	Description
imsi	Identifies the HLR subscriber.
msisdn	The Mobile Subscriber ISDN Number assigned to the user, for charging purpose.
ProfileData	Contains a list of HLR common Service Profiles including Subscriber Profile, GPRS Profile, CAMEL Profile, and gsmSCF Profile. Common Service Profiles must be preconfigured or preprovisioned in the HLR-FE and in the CUDB.
SubscriberData	Indicates, by the ICS flag, whether it is an ICS user. Provides and activates Baring Outgoing International Call (BOIC), with Basic Service Group TS10 (Voice), and TS20 (SMS).



3 Preparing for VoLTE Service Model Development

3.1 Preparing for CA Development Environment

CA service models are developed in Designer Studio. For more information about the configuration of Designer Studio environment, refer to *User Guide for Designer Studio*, Reference [5].

3.2 VoLTE Decision Service Model Development Workflow

The VoLTE decision service model design base project contains the auto provisioning criteria, and the project is located in the `CXP9029435-<version>.tar.gz > ca-deployables-<version>.tar.gz` file, in the Dynamic Activation distribution.

To develop the CA for VoLTE decision service model, use the `VOLTEAutoProvisioning_DP-<version>.tar.gz` project as a design base, and follow these steps:

1. Analyze the service model application. See Section 4.1 on page 15 for details.
2. Define the service model data model. See Section 4.2 on page 15 for details.
3. Define the service model access control model. See Section 4.3 on page 16.
4. Initialize the service model projects. See Section 4.4 on page 16 for details.
5. Validate the request schema. See Section 4.5.1 on page 17 for details.
6. Validate the VoLTE auto provisioning criteria. See Section 4.5.2 on page 17 for details.
7. Handle the concurrent requests. See Section 4.5.3 on page 17 for details.
8. Map the default service profiles to the execution service model. See Section 4.5.4 on page 18 for details.
9. Map error codes. See Section 4.5.5 on page 19 for details.
10. (Optional) Integrate with UPG. See Section 4.5.6 on page 19.
11. Notify the results. See Section 4.5.7 on page 19 for details.
12. Build and deploy the projects. See Section 5 on page 25 for details.



13. Test and verify the projects. See Section 7 on page 29 for details.

3.3

VoLTE Execution Service Model Development Workflow

The VoLTE execution service model design base project contains the provisioning workflow, and the project is located in the `CXP9029435-<version>.tar.gz` > `ca-deployables-<version>.tar.gz` file, in the Dynamic Activation distribution.

To develop the CA for VoLTE execution service model, use the `VOLTEAutoProvisioning_EP-<version>.tar.gz` project as a design base, and follow these steps:

1. Analyze the service model application. See Section 4.1 on page 15 for details.
2. Define the service model data model. See Section 4.2 on page 15 for details.
3. Define the service model access control model. See Section 4.3 on page 16.
4. Initialize the service model projects. See Section 4.4 on page 16 for details.
5. Validate the request schema. See Section 4.6.1 on page 21 for details.
6. Map the service model request to the southbound application requests. See Section 4.6.2 on page 21 for details.
7. Handle exceptions and ignored errors. See Section 4.6.3 on page 22 for details.
8. Handle data inconsistency. See Section 4.6.4 on page 22 for details.
9. Build and deploy the projects. See Section 5 on page 25 for details.
10. Test and verify the projects. See Section 7 on page 29 for details.



4 VoLTE Service Model Development

This section holds information about the VoLTE service model development.

4.1 Analyzing the VoLTE Service Model Application

The first step for VoLTE developers is to confirm that the VoLTE requirements are supported, decide what customer-specific VoLTE features that are to be provisioned, and the corresponding data model details.

4.2 Defining the VoLTE Service Model

The procedure of defining a VoLTE service model data model is as follows:

- Customize the northbound interface data model:

Note: The northbound interface namespace must contain `http://schemas.ericsson.com/ma/CA/` as prefix for license control, otherwise the CA application is not deployed. For example, the namespace for VoLTE decision service model is `http://schemas.ericsson.com/ma/CA/Service/`, and the namespace for VoLTE execution service model is `http://schemas.ericsson.com/ma/CA/VoLTESubscriber/`.

VoLTE decision service model interface includes the notification identities IMSI, MSISDN, and so on. Customers can use it directly. Update is required when the sub MO interface needs to be upgraded by new services, demanded by VoLTE requirements.

VoLTE execution service model interface integrates the complete sub MOs interface, which provides full application services. Customers can use it directly. Update is required when the sub MO interface needs to be upgraded by new services, demanded by VoLTE requirements.

- a Find the northbound interface in the `CXP9029435-<version>.tar.gz` > `ca-sourcecode-<version>.tar.gz` > `JDV-VOLTE-AUTO-P` file.
 - b Define the namespace, operations, and other related parameters of the requests in the schema and WSDL files according to *Generic CA/3G Interface 1.2*, Reference [4].
- Customize the southbound interface data model:
 - a Find VoLTE decision service model southbound interface, which is the northbound interface of the VoLTE execution service model.



Find DPHelp interface, which is located in the `CXP9029435-<version>.tar.gz > ca-sourcecode-<version>.tar.gz > JDV-DPHELP-P` file.

- b Define the namespace, operations, and other related parameters of the requests in the schema and WSDL files according to *Generic CAI3G Interface 1.2*, Reference [4].

4.3 Defining the VoLTE Service Model Access Control Model

For information on how to define the VoLTE Service Model Access Control Model, refer to *User Guide for Resource Activation*, Reference [6].

4.4 Initializing the VoLTE Service Model Project

4.4.1 Importing the VoLTE Service Model Design Base Project

CA developers can initialize their own VoLTE project by importing the existing VoLTE service model design base project.

Follow the steps:

1. Start Designer Studio to import the CA southbound interface for decision service model or standard southbound applications interface for execution service model, according to the section *Updating Southbound Interface* in *User Guide for Designer Studio*, Reference [5].
2. Click **Service Models > Import Service Models**.
3. In the **Import Service Models** dialog, click **Choose File** and select **VOLTEAutoProvisioning_DP-<version>.tar.gz** or **VOLTEAutoProvisioning_EP-<version>.tar.gz**.
4. Click **Next** and **Finish**.
5. If customers define the northbound interface for the decision service model or execution service model, import the customized northbound interface by clicking **Actions > Update northbound schema**.

4.4.2 Renaming the VoLTE Service Model Design Base Project (Optional)

Follow these steps to rename the VoLTE service model design base project:

1. In the **Import Service Models** dialog, select **VoLTE Service Model**.
2. Click **Actions** and select **Rename**.



3. In the **Rename Service Model** dialog, fill in name and version, and click **Confirm**.

4.5 Handling the VoLTE Decision Service Model Business Operations

After VoLTE decision service model receives an XML request, the model executes the inner defined tasks.

The handling steps of an incoming request are as follows:

1. Validate the inbound request. See Section 4.5.1 on page 17.
2. Handle VoLTE Auto Provisioning criteria. See Section 4.5.2 on page 17.
3. Handle concurrent requests. See Section 4.5.3 on page 17.
4. Map default service profiles. See Section 4.5.4 on page 18.
5. Map error codes. See Section 4.5.5 on page 19.
6. (Optional) Integrate with UPG. See Section 4.5.6 on page 19.
7. Notify results. See Section 4.5.7 on page 19.

4.5.1 Handling CAI3G Data Validation

Designer Studio already supports the CAI3G interface schema validation for the service model.

4.5.2 Handling VoLTE Auto Provisioning Criteria

The current VoLTE auto provisioning criteria is based on the following cases:

- The end user is an EPS user.
- The end user is not doing IMSI changeover.
- The end user is not a VoLTE user.

When VoLTE auto provisioning criteria is changed, the tasks must be redefined in decision service model.

4.5.3 Handling Concurrent Requests

Handling concurrent requests for the decision service model includes: `CreateLockService` and `DeleteLockService`:

- The `CreateLockService` task is created at the beginning of executing the VoLTE decision service model, and the task must be enabled for rollback.

- The `DeleteLockService` task is created at the end of executing the VoLTE decision service model.

The concurrent request `MoType` is `LockService@http://schemas.ericsson.com/ma/CA/DPHelper`.

For details about how to assign parameters and set condition for the `CreateLockService` task and `DeleteLockService` tasks, refer to the VoLTE decision service model template in Designer Studio as well as *User Guide for Designer Studio*, Reference [5].

For details on how to handle concurrent requests during the Create and Delete operations, see the following files in `CXP9029435-<version>.tar.gz` > `ca-sourcecode-<version>.tar.gz` > JDV-DPHELP-P:

- `com.ericsson.jdv.dphelper.handler.CreateLockServiceLogic`
- `com.ericsson.jdv.dphelper.handler.DeleteLockServiceLogic`

4.5.4 Mapping Default Service Profiles

In the VoLTE decision service model, define the default VoLTE service profiles before executing the `CreateVolteSubscription` task to create the VoLTE user.

The IMS service data of the `CreateVolteSubscription` task is mapped by the `GetDefaultVolteProfile` task. Other services data is mapped by setting data parameters directly.

`GetDefaultVolteProfile` request `MoType`: `DefaultVolteProfile@http://schemas.ericsson.com/ma/CA/DPHelper/`

`CreateVolteSubscription` request `MoType`: `VoLTESubscription@http://schemas.ericsson.com/ma/CA/VoLTESubscriber/`

For details about how to assign parameters and set conditions for the `GetDefaultVolteProfile` task and `CreateVolteSubscription` tasks, see the VoLTE decision service model template in Designer Studio as well as *User Guide for Designer Studio*, Reference [5].

For details on how to generate the default service profiles during the Get operation, see the following file in `CXP9029435-<version>.tar.gz` > `ca-sourcecode-<version>.tar.gz` > JDV-DPHELP-P:

`com.ericsson.jdv.dphelper.handler.GetDefaultVolteProfileLogic`

The default IMS service profiles named `GetResponseDefaultVolteProfile.xml` is provided as the template, which is located in `CXP9029435-<version>.tar.gz` > `ca-sourcecode-<version>.tar.gz`



```
> JDV-DPHELP-P/JDV-DPHelper-Provisioning/src/main/resources/META-INF/template/.
```

4.5.5 Mapping Error Codes

User self-defined error function is provided by VoLTE auto provisioning. When the current scenarios do not meet VoLTE auto provisioning criteria, a new CA validation error code is returned.

In VoLTE decision service model template, create an error code task to map the original error code to a new CA validation error code.

The new error code request MoType is `ErrorCode@http://schemas.ericsson.com/ma/CA/DPHelper/`

For details about how to assign parameters and set conditions for the new error code task, see the VoLTE decision service model template in Designer Studio and *User Guide for Designer Studio*, Reference [5].

For details on how to create error code requests, see the following file in `CXP9029435-<version>.tar.gz` > `ca-sourcecode-<version>.tar.gz`
> JDV-DPHELP-P:

- `com.ericsson.jdv.dphelper.handler.CreateErrorCodeLogic`

4.5.6 Integrating with UPG (Optional)

Customer can first develop MV-NE of UPG to support synchronizing user's supplementary service. And define a task in the VoLTE decision service model to send the synchronization subscription request to UPG through the MV-NE after the VoLTE execution service model is successfully completed.

For more information on MV-NE development, refer to *Customer Adaptation Development Guide for Resource Activation*, Reference [3].

4.5.7 Notifying Results

Customers need to configure the target BSS endpoint in WSDL file to send asynchronous CAI3G notification results to BSS:

- **Username:** Set the username value in `<wsse:UsernameToken>`, and the default value is `admin`.
- **Password:** Set the password value in `<wsse:UsernameToken>`, and the default value is `admin`.
- **ReplyTo:** Set the ReplyTo value in `<wsa:Address>`, and the default value is `http://business456.example/client1`.



- **FaultTo:** Set the FaultTo value in <wsa:Address>, and the default value is http://business456.example/client1.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
  xmlns:cai3="http://schemas.ericsson.com/cai3gl.2/"
  xmlns:async="http://schemas.ericsson.com/async/">
  <Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>sogadm</wsse:Username>
        <wsse:Password>sogadm</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
    <wsa:MessageID>6B29FC40-CA47-1067-B31D-00DD010662DA</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://business456.example/client1</wsa:Address>
    </wsa:ReplyTo>
    <wsa:FaultTo>
      <wsa:Address>http://business456.example/client1</wsa:Address>
    </wsa:FaultTo>
    <async:FireTime>2012-05-30T09:00:00</async:FireTime>
    <async:GroupId>IMSI=123456789000</async:GroupId>
  </Header>
  <Body>
    <cai3:Create xmlns:cai3="http://schemas.ericsson.com/cai3gl.2/"
      xmlns:ns="http://schemas.ericsson.com/ma/Service/">
      <cai3:MOType>DefaultProfile@http://schemas.ericsson.com/ma/Service/
    </cai3:MOType>
    <cai3:MOId>
      <ns:serviceType>VOLTE</ns:serviceType>
      <ns:imsi>123456789000</ns:imsi>
    </cai3:MOId>
    <cai3:MOAttributes>
      <ns:CreateDefaultServiceProfile serviceType="VOLTE">
        <ns:serviceType>VOLTE</ns:serviceType>
        <ns:imsi>123456789000</ns:imsi>
      </ns:CreateDefaultServiceProfile>
    </cai3:MOAttributes>
  </cai3:Create>
</Body>
</Envelope>
```

Example 1 Asynchronous Request

If customers need to change the configurations in future, follow these steps to configure the preceding values for the notification NBIA submodule:

1. Set the configuration values for auto-provisioning as the root user:

```
$ bootloader.py config set --parameter @AUTO_PROVISIONING_USER@ --value <Username value>
```

```
$ bootloader.py config set --parameter @AUTO_PROVISIONING_PWD@ --value <Password value>
```

```
$ bootloader.py config set --parameter @AUTO_PROVISIONING_REPLY_TO@ --value <ReplyTo value>
```

```
$ bootloader.py config set --parameter @AUTO_PROVISIONING_FAULT_TO@ --value <FaultTo value>
```

2. Activate the configuration for auto provisioning as the root user:



```
$ bootloader.py node activate --host all
```

A successful response contains the IMSI, while a failed response contains the IMSI and the error information.

4.6 Handling the VoLTE Execution Service Model Business Operations

The VoLTE execution service model defines the fundamental service execution tasks for IMS, MTAS, DNS, EPS, SAPC, and HLR. When the VoLTE execution service model receives an XML request, the service model executes the inner defined tasks.

The handling steps of an incoming request are as follows:

1. Validate the inbound request. See Section 4.6.1 on page 21.
2. Map the request data. See Section 4.6.2 on page 21.
3. Handle exceptions. See Section 4.6.3 on page 22.
4. Handle data inconsistency. See Section 4.6.4 on page 22.

4.6.1 Handling CAI3G Data Validation

Designer Studio already supports the CAI3G interface schema validation for the service model.

4.6.2 Mapping Request Data

The VoLTE execution service model defines the fundamental service execution tasks for IMS, MTAS, DNS, EPS, SAPC, and HLR.

- In the **Create** tab, map northbound request data to all tasks.

The `getHLR` task is to do the migration of data from HLR to MMTel (MTAS).

- In the **Get** tab, map northbound request data to the `GetIMSService` and `GetSAPCService` tasks. Other tasks mapping request data such as application identities are from the `GetIMSService` task response data.

The `getHLR` task is to judge whether the user has Volte flag(`ics=1`). If not, do not return HLR data.

- In the **Delete** tab, map northbound request data to the `GetIMSService` and `DeleteSAPCService` tasks. Other tasks mapping data such as application identities are from the `GetIMSService` task response data.



The `getHLRService` task is to judge whether the user has `Volte` `flag(ics=1)`. If not, loose it.

- In the **Set** tab, map request data to all tasks.

For details, see the service model template in `VOLTEAutoProvisioning_EP-<version>.tar.gz`.

For more information about how to map the VoLTE northbound request to the southbound applications, refer to the section *Assign Parameters for a Task* in *User Guide for Designer Studio*, Reference [5].

Note: Designer Studio executes tasks in parallel, while VoLTE execution service model solution executes tasks in sequential order and in parallel. For the tasks that are executed in sequential order, configure them first according to the section *Define Condition for a Task* in *User Guide for Designer Studio*, Reference [5].

4.6.3 Handling Exceptions

The execution service mode handles the following specific errors:

For the `Create` operation:

- Configure the error handling for the `GetEPSService` task. If EPS subscription does not exist, `Create` an EPS service. If EPS subscription exists, `Set` the EPS service.
- Configure the error handling for the `GetHLRService` task. If HLR subscription does not exist, `Create` an HLR service, if HLR subscription is exists, `Set` the HLR service.

For the `Delete` operation, if any service does not exist, configure loose error handling.

4.6.4 Handling Data Inconsistency

During the `Create` and `Set` operations on the service model, the requests towards tasks can fail. If one or more task requests for this service model operation fail, previously successful requests are rolled back.

The rollback is done by creating a task. To create a task, select **Enable rollback** in the **Error Handling** tab. If rollback is successful, the service model `Create` and `Set` operations return the information that the operations failed, but rollback was successful.

During the `Delete` operations towards the service model, configure loose error handling if data does not exist.



For details on how to handle data inconsistency during the Create/Set/Delete operations, see the Create/Set/Delete container of the service model template in `VOLTEAutoProvisioning_EP-<version>.tar.gz`.





5 Building and Deployment

5.1 Deploying DPHelper JDV

Before deploying CA service model, deploy DPHelper JDV first to support CA service model.

5.1.1 Preparing for CA Package

The CA development package is part of the Dynamic Activation software deliverable. Regardless of native deployment or visualized deployment, download the CA Package SW Container CXP9029435.zip from SW Gateway, and extract the 19089-CXP9029435_<version>.tar.gz file to the /var/log/installfiles/ directory.

This package contains the following packages:

- ca-deployables-<version>.tar.gz - Deployable CA Design Based projects
- ca-plugins-<version>.tar.gz - Development tools
- ca-repository-<version>.tar.gz - Maven repository and settings.xml file
- ca-sourcecode-<version>.tar.gz - CA Design Based Projects source code
- ca-dup-convert-tool-<version>.tar.gz - DUP migration tools

5.1.2 Installing DPHelper JDV

To install DPHelper JDV:

1. Extract the ca-deployables-<version>.tar.gz file to a local directory.
2. Upload JDV-DPHELPER-Provisioning-<version>.tar.gz file to the /var/log/installfiles/ directory.
3. Change the owner and the group of the JDV-DPHELPER-Provisioning-<version>.tar.gz file.

```
# chown actadm:activation /home/bootloader/repository/JDV-DPHELPER-Provisioning-<version>.tar.gz
```

4. Copy JDV-DPHELPER-Provisioning-<version>.tar.gz file to the /home/bootloader/repository/ directory.



5. Install the DPHelper JDV as the `root` user to the `dve-application` module.

```
$ bootloader.py submodule add -n JDV-DPHELPER-Provisioning-<version>.tar.gz -t jdv -p dve-application --host <hostname>
```

6. Activate the installed JDV as the `root` user.

```
$ bootloader.py node activate --host all
```

5.1.3 Uninstalling DPHelper JDV

To uninstall DPHelper JDV:

1. Uninstall the DPHelper JDV as the `root` user to the `dve-application` module.

```
$ bootloader.py submodule delete -n JDV-DPHELPER-Provisioning-<version>.tar.gz -p dve-application --host <hostname>
```

2. Activate the installed JDV as the `root` user.

```
$ bootloader.py node activate --host all
```

5.2 Deploying and Undeploying a CA Service Model

For information on how to deploy and undeploy a CA service model, follow the instructions in *User Guide for Designer Studio*, Reference [5].

For information on how to find VoLTE decision service model, see Section 3.2 on page 13.

For information on how to find VoLTE execution service model, See Section 3.3 on page 14.



6 Configuring Dynamic Activation for VoLTE Provisioning

6.1 Configuring NE in Dynamic Activation

Customers need to configure the VoLTE relevant NEs, IMS-HSS, MMTel-MTAS, ENUM-IPWorks, EPS-HSS, SAPC and HLR, in Dynamic Activation. For general information about how to configure NE in Dynamic Activation, refer to *User Guide for Resource Activation*, Reference [6].

6.2 Configuring Notification Endpoint in CUDB

CUDB sends the VoLTE auto provisioning notification to Dynamic Activation, and CUDB needs to configure the notification endpoint towards Dynamic Activation in the following formats:

- `http://<VIP-TRAFFIC-IP>:8080/soap/services/notification`
- `https://<VIP-TRAFFIC-IP>:8181/soap/services/notification`





7 Testing and Provisioning over CAI3G

This section provides some common tools that can be used for testing a customer adaptation.

7.1 Testing Tools for Sending Requests – SoapUI

SoapUI is a cross-platform functional testing solution. With an easy-to-use graphical interface and enterprise-class features, SoapUI allows to easily and rapidly create and execute automated functional, regression, compliance, and load tests. CA developers can use this tool to send CAI3G requests to Dynamic Activation and check the response from Dynamic Activation.

Download and install SoapUI from <http://www.soapui.org>.





Reference List

Ericsson Documents

- [1] *Library Overview*, 18/1553-CSH 109 628 Uen
- [2] *Solution Description VoLTE*, 2/221 02-CSH 109 628 Uen
- [3] *Customer Adaptation Development Guide for Resource Activation*, 5/1553-CSH 109 628 Uen
- [4] *Generic CAI3G Interface 1.2*, 2/155 19-FAY 302 0003 Uen
- [5] *User Guide for Designer Studio*, 10/1553-CSH 109 628 Uen
- [6] *User Guide for Resource Activation*, 1/1553-CSH 109 628 Uen