EXPERIMENTATION AND VALIDATION OPENNESS FOR LONGTERM EVOLUTION OF VERTICAL INDUSTRIES IN 5G ERA AND BEYOND

[H2020 - Grant Agreement No.101016608]

Documentation on the EVOLVED-5G CAPIF core function (CCF) Release 1.0

Contributors

- **TID** (TELEFONICA INVESTIGACIÓN Y DESARROLLO)
- **FOGUS** (FOGUS INNOVATIONS & SERVICES P.C.)

Version 1.0

Date February 16, 2022

Distribution Public

LIST OF AUTHORS

Author's Name & Surname	*Organization's Full name*
Dimitris Tsolkas	FOGUS INNOVATIONS & SERVICES P.C
Jorge Moratinos Salcines	TELEFONICA INVESTIGACIÓN Y DESARROLLO
Stavros-Anastasios Charismiadis	FOGUS INNOVATIONS & SERVICES P.C.
Alejandro Molina Sanchez	TELEFONICA INVESTIGACIÓN Y DESARROLLO
David Artuñedo Guillen	TELEFONICA INVESTIGACIÓN Y DESARROLLO

TABLE OF CONTENTS

INTRODUCTION

To avoid duplication and inconsistency of approach between the various API specifications that 3GPP has released, the development of a **common API framework (CAPIF)** has been considered. CAPIF includes common aspects applicable to any northbound service APIs. As such, it is a complete 3GPP API framework that covers functionality related to: on-board and off-board API invokers, register and release APIs that need to be exposed, discovering APIs by third entities, as well as authorization and authentication.

CAPIF functionality is considered as a cornerstone in the realization of **5G openness**, since it allows secure exposure of 5G core APIs to third party domains, and also, enables third parties to define and expose their own APIs. Indeed, CAPIF has become already a fundamental feature for the 3GPP SA6, targeting the interaction of *Verticals* with the 5G system. More precisely, CAPIF compliance is required (Figure 1) in i) the development of Vertical Application Enablers (VAEs) for various industries (V2X, Factories of the Future, etc.), ii) the realization of the Service Enabled Architecture Layer (SEAL), as well as iii) the implementation of the service side of Edge Computing services.

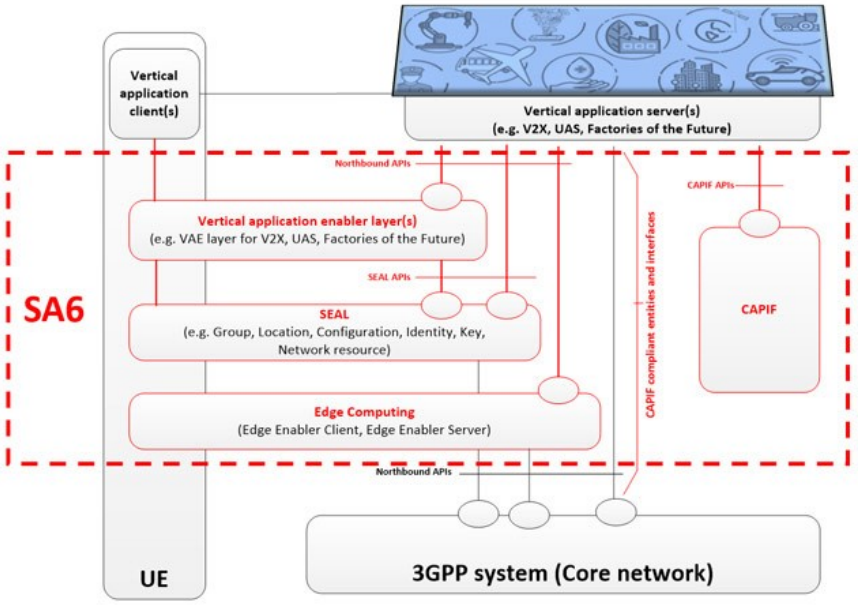

Figure 1 CAPIF in the context of 3GPP SA6 activities

This report serves as supplementary documentation that accompanies the Release 1 of the code that is being developed by TID and FOGUS in the context of EVOLVED-5G project in order to realize the Core Function of CAPIF, i.e., the main entity of the CAPIF Functional Architecture.

The scope of the development process reported in this document is twofold.

- First, to provide a description of the tool that realizes the CAPIF Core Function (CCF), i.e., a tool that implements all the CCF APIs as a bunch of services;
- Second, to provide a set of test plans that allows any third party that implements the other architectural components i.e., the API invoker and/or the API provider to check whether it can properly (as defined in the related 3GPP standards) interact with the CCF.

RELATED STANDARDS

The specification of the **CAPIF functional architecture** is covered in 3GPP TS 23.222 "Functional architecture and information flows to support Common API Framework for 3GPP Northbound APIs" (since Release 15) and it is also published in the related ETSI TS 123.222 "Common API Framework for 3GPP Northbound APIs". For reference purposes of the development process adopted, the most recent technical specification report is used, as presented in 3GPP TS 23.222 Rel.17 V17.5.0, 2021. Based on the architecture and the procedures defined in that report, additional information and requirements are considered, regarding **CAPIF security features and security mechanisms**, as presented in 3GPP TS 33.122 Rel.16 V16.3.0, 2020.
The specification of the **CAPIF APIs** that are needed for realizing the CAPIF functionality are part of 3GPP TS 29.222. The scope of this technical specification is to describe the protocol for the Common API Framework (CAPIF) for 3GPP Northbound APIs. Those APIs are also published in the related ETSI TS 129.222 "Common API Framework for 3GPP Northbound APIs". For reference purposes of the development process adopted, the most recent technical specification report is used, as presented in 3GPP TS 29.222 Rel.17 V17.3.0, 2021.

BACKGROUND ON CAPIF FUNCTIONALITY

To better specify the functionality that is being implemented, the CAPIF reference architecture and the list of CAPIF APIs are provided below (Figure 2).

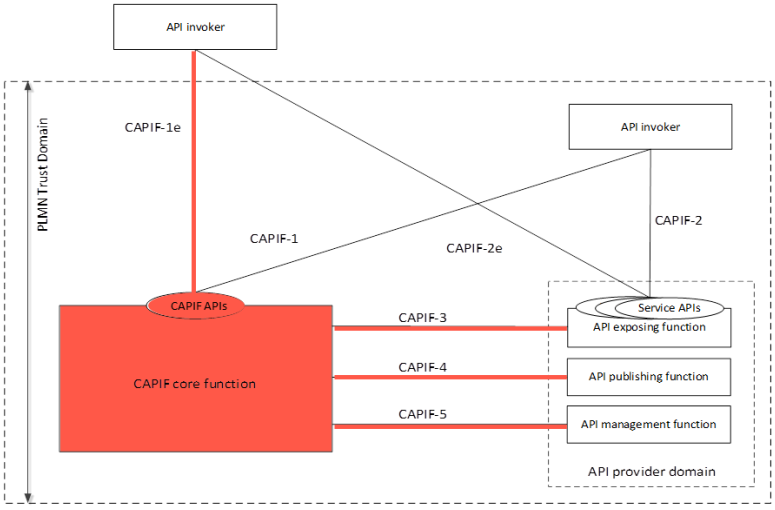

Figure 2 Main functional entities and Reference points of CAPIF architecture (3GPP TS 23.222). The Functional entity and the Reference points that are considered in the development process are highlighted.

The API invoker is typically provided by a 3rd party application that supports the following capabilities:

- Supporting the authentication by providing the API invoker identity and other information required for authentication of the API invoker;
- Supporting mutual authentication with CAPIF;
- Obtaining the authorization prior to accessing the service API;
- Discovering service APIs information; and
- Invoking the service APIs.

The CAPIF core function is the main entity of the CAPIF and it consists of the following capabilities:

- Authenticating the API invoker based on the identity and other information required for authentication of the API invoker;
- Supporting mutual authentication with the API invoker;
- Providing authorization for the API invoker prior to accessing the service API;
- Publishing, storing and supporting the discovery of service APIs information;
- Controlling the service API access based on PLMN operator configured policies;
- Storing the logs for the service API invocations and providing the service API invocation logs to authorized entities;
- Charging based on the logs of the service API invocations;
- Monitoring the service API invocations;
- Onboarding a new API invoker and offboarding an API invoker;
- Storing policy configurations related to CAPIF and service APIs;
- Support accessing the logs for auditing (e.g. detecting abuse); and
- Supports publishing, discovery of service APIs information with another CAPIF core function in CAPIF interconnection.

The API provider is an entity that provides API exposing, publishing and management functions. For each one of the functions specific reference points have been defined (CAPIF-3,4, and 5, as presented in Figure 2).

- **The API exposing function (AEF)** is the provider of the service APIs and is also the service communication entry point of the service API to the API invokers. The API exposing function consists of the following capabilities:
  - Authenticating the API invoker based on the identity and other information required for authentication of the API invoker provided by the CAPIF core function;
  - Validating the authorization provided by the CAPIF core function; and
  - Logging the service API invocations at the CAPIF core function.
- **The API publishing function (APF)** enables the API provider to publish the service APIs information in order to enable the discovery of service APIs by the API invoker. The API publishing function consists of the following capability:
  - Publishing the service API information of the API provider to the CAPIF core function.
- **The API management function (AMF)** enables the API provider to perform administration of the service APIs. The API management function consists of the following capabilities:
  - Auditing the service API invocation logs received from the CAPIF core function;
  - Monitoring the events reported by the CAPIF core function;
  - Configuring the API provider policies to the CAPIF core function;
  - Monitoring the status of the service APIs; and
  - Registering and maintaining registration information of the API provider domain functions on the CAPIF core function.

From the deployment perspective, the distributed model (defined in 3GPP TS23.222) is under the scope of the development process.

In distributed deployment, the CAPIF core function and the API provider domain functions reside in different domains. The CAPIF core function interacts with the API exposing function, the API publishing function and the API management function via CAPIF-3, CAPIF-4 and CAPIF-5 reference points respectively. The API invoker can interact independently with the CAPIF core

function and the API exposing function including the service APIs. In this deployment, the API exposing function appears as an agent for all service API invocations from the API invoker. The API invoker obtains the service API information and its entry point details from the CAPIF core function via CAPIF-1 interface. The first point of entry for the service API is the API exposing function during API invocation. The API exposing function acts as agent for service API applying any access control or policy control to the interactions between the API invoker and the service API in coordination with the CAPIF core function via CAPIF-3 interface.

From the business perspective, in the development process the API invoker is considered as a 3rd party application which is subject of a service agreement with the CAPIF provider, and as such, it resides outside the trusted domain of the CAPIF provider.

For an API invoker that resides in an untrusted domain, the CAPIF-1e reference point, which exists between the API invoker and the CAPIF core function (Figure 2) is used to discover service APIs, to authenticate and to get authorization.

IMPLEMENTATION ASPECTS

The approach that is followed for the implementation of the CAPIF Core Function is presented in Figure 3.

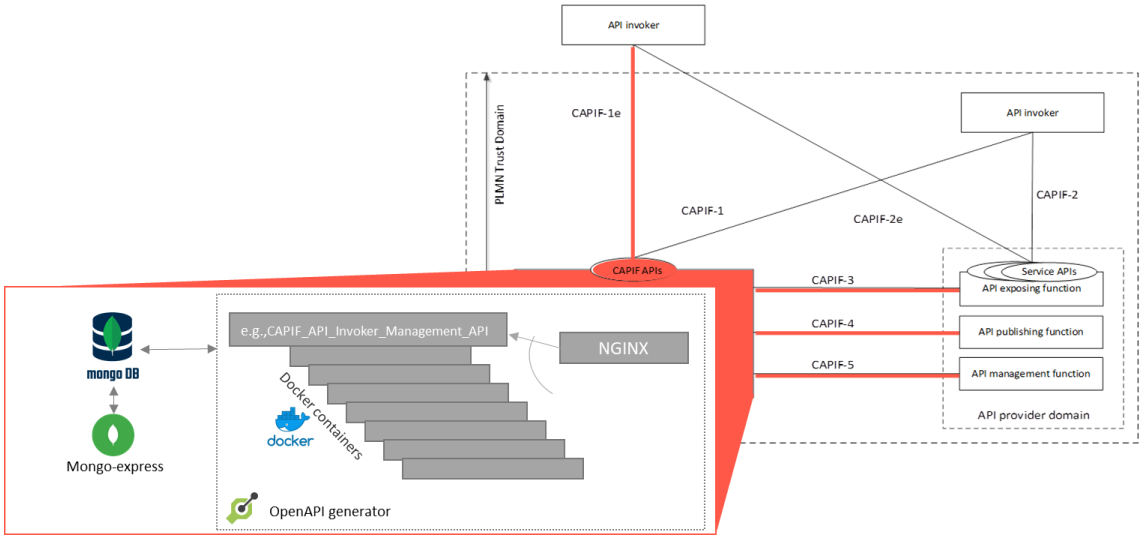

Figure 3 Main tools used for the Implementation of the CAPIF core function

The main principles followed for the code production are:

- The CAPIF API descriptions are adopted from the following repository: https://github.com/jdegre/5GC_APIs
- The **OpenAPI framework**: https://github.com/OpenAPITools/openapi-generator is used to automatically generate
  - **CAPIF core function APIs services** as specified in 3GPP TS29.222. Each API service is hosted by a separated Docker container.
  - **NGINX server** (hosted in a separate docker container) serves as a proxy for the CAPIF API requests.
  - **JWTauth** (hosted in a separate docker container) to manage the JSON web token creation to be used in each CAPIF API call.

- Auxiliary features are deployed for manual interaction with CCF APIs, using either **Curl** or **PostMan**.
- A full set of test plans to accompany the CAPIF API services. The test plans exploit the capabilities of **Robot framework** to check the functionality of the CAPIF API services.
- A **MongoDB** is set in a separated container to support as backend/registry the CAPIF core function (CCF). **Mongo Express GUI** is also deployed (in a container) to check/monitor inputs and records.

REMARKS ON RELEASE 1.0

The entire code of the CAPIF core function Release 1.0 can be found in the following public repository of the EVOLVED-5G project:

Code: EVOLVED-5G_CAPIF_API_Services_RELEASE1.0
Development branch: EVOLVED-5G_CAPIF_API_services_development_branch

The above-mentioned repository includes a set of the Python-flask Mockup servers that provide the CAPIF core function APIs as well as auxiliary code and instructions for installing and testing the API services.

More precisely, the related repository includes the following folders:

- **services**: Services developed following CAPIF API specifications. Also, other complementary services (e.g., NGINX and JWTauth services for the authentication of API consuming entities).
- **tools**: Auxiliary tools. Robot Framework related code and OpenAPI scripts
- **test**: Tests developed using Robot Framework
- **docs**: Documents related to the code in the repository
    - images: images used in the repository
    - test_plan: test plan descriptions for each API service referring to the test that are executed with the robot framework.
    - testing_with_postman: auxiliary JSON file needed for the Postman-based examples.
- **iac**: Infrastructure as Code, contains all the files needed for the deployment of the structure that supports the services. (It is used only for the case of non-local deployment of the CCF services e.g., in the Openshift of EVOLVED-5G project))
    - Terraform: Deploy file
- **pac**: Jenkins files to manage different automated tasks. (It is used only for the case of non-local deployment of the CCF services e.g., in the Openshift of EVOLVED-5G project)
    - Jenkins Pipelines

In Release 1.0, three main CCF APIs services (plus an auxiliary one) are provided:

- *JWT Authentication APIs (Auxiliary API service)*
- **CAPIF Invoker Management API**
- **CAPIF Publish API**
- **CAPIF Discover API**

EXPLORING CCF RELEASE 1.0

Complete "HOW TO" instructions are available in the README file in the repository.

INSTALL AND RUN CAPIF SERVICES IN A LOCAL REPOSITORY

Three different options are offered as described in the related Github repository:

- Run All CAPIF Services locally with Docker images
- Run each service using Docker
- Run each service using Python

NOTE 1: The use of OpenAPI generator during the initial set up process will provide the images of all the CCF API services. However, it should be noted that in Release 1.0, the functionalities of CAPIF Invoker Management API, CAPIF Publish API, and the CAPIF Discover API are available and ready to use/test.

NOTE 2: After the successful installation the Mongo DB Dashboard will be available on http://0.0.0.0:8082/ (if accessed from localhost) or http://<Mongo Express Host IP>:8082/ (if accessed from another host)

FAMILIARIZE WITH CAPIF CORE FUNCTION APIS

Two different options are offered, as listed below:

- Execute CAPIF API calls using PostMan (step-by-step instructions)
- Execute CAPIF API calls using Curl (step-by-step instructions)

EXECUTE TEST PLANS FOR THE CAPIF CORE FUNCTION APIS

For testing the functionality of the APIs provided, a complete set of tests has been prepared (CCF_Release1.0_test_plans). For the execution of the tests the Robot framework is used. Instructions for setting up the Robot Framework and running the tests are also provided (step-by-step instructions)