# 1. INTRODUCTION

Network-as-a-Service (NaaS) represents a paradigm shift whereby a network operator can make network capabilities available for external consumption, including monitoring and configuration related capabilities, through Application Programming Interfaces (APIs). The on-demand, secure and auditable exposure of these capabilities will pave the way for transforming telco networks into service enablement platforms, facilitating the application-to-network integration, which will be key to deliver enhanced and service-tailored customer experience in the 5G era. To ensure wide market adoption and an attractive economy of scale for all stakeholders in the value chain, it is essential to push the industrialization of open, global and user-friendly APIs. To that end, a number of industry partners have constituted CAMARA, a joint initiative between the Linux Foundation’s CNCF and GSMA to foster the definition, development and validation of these APIs.

Figure 1 depicts the reference architectural framework of CAMARA. As seen, it includes the following components:

* **Internal APIs**: these are the APIs which are implemented in telco assets, including network resources (core, access, transport functions), cloud resources (virtualized and cloud-native workload hosting infrastructures) and IT resources (OSS and NOC tools). These APIs are typically defined in SDOs or industry fora, and quiet tied to the underlying technology. Examples of these APIs include the ones defined by 3GPP, ETSI, TMF and CNCF, among others.
* **Service APIs**: these are the APIs that are made available for consumption to 3rd parties. These APIs shall be adhered to three main principles:
  + *Open*. Service APIs need to be designed following open process, be publicly available, and have various rights to use associated with it (Apache2.0 license).
  + *Global*. Service APIs shall allow every 3rd party to have an uniform and consistent service experience across different operators, with the effortless portability of applications across their platforms (design-once-run-everywhere approach) and easy service replicability.
  + *User-friendly.* Service APIs need to be abstracted out of internal APIs, to hide elco complexity and make them easy to use (consume) to 3rd parties with no telco expertise/background.
* **Transformation function**: it keeps the information on correspondences (mappings) between service APIs and internal APIs, and executes the workflows to enforce these mapping. The transformation function is typically deployed as a microservice provisioned with a workflow engine.
* **Exposure Gateway**: it provides all the capabilities that are needed to provide a controllable capability exposure to 3rd parties.

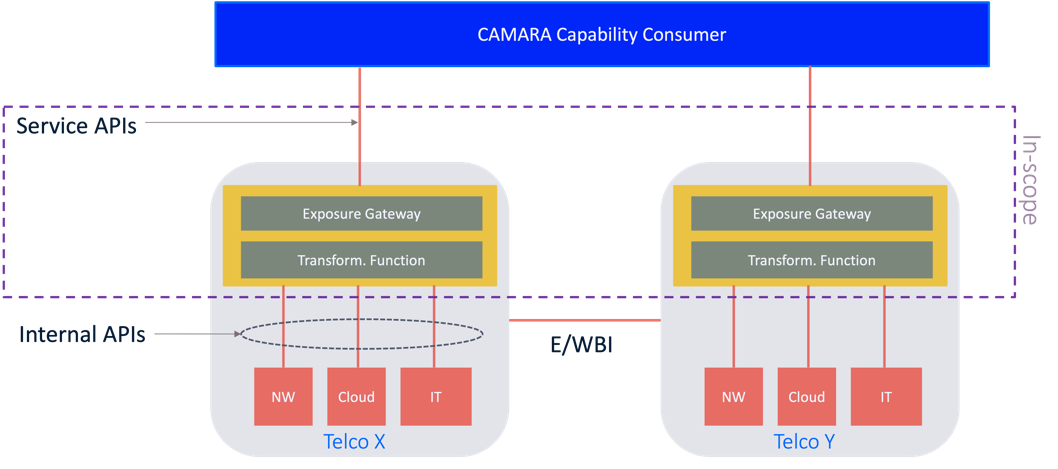


Figure : CAMARA reference architectural framework

The above architecture is only for reference, and it is not tied to any implementation. Actually, different architecture instantiations (flavors) can be found, associated to different business models and go-to-market strategies. Figure 2, Figure 3 and Figure 4 illustrates examples of these flavors.

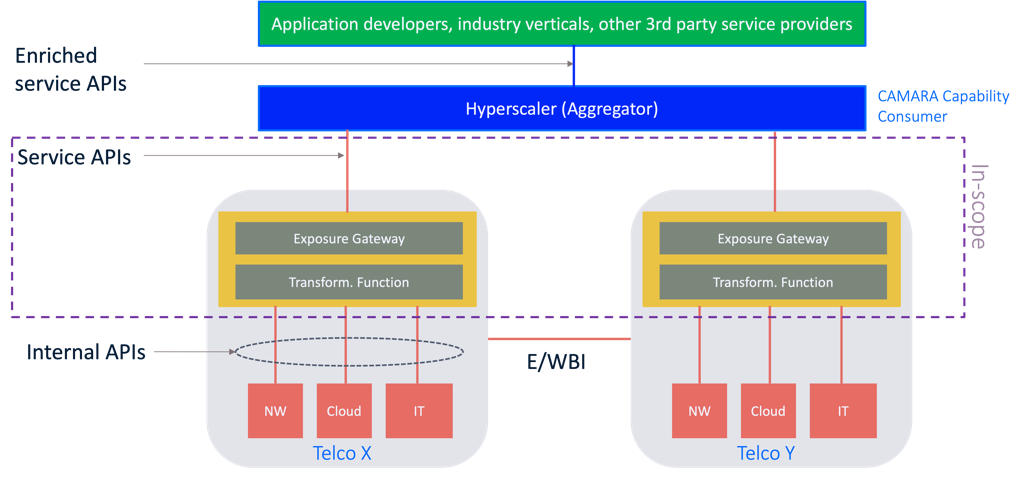


Figure 2: Example of realization of CAMARA architecture (flavor #1)

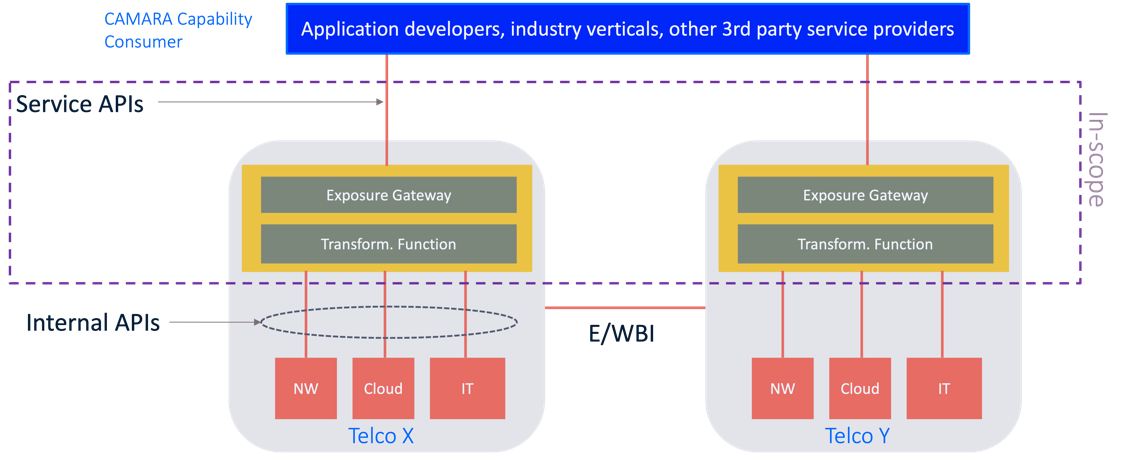


Figure : Example of realization of CAMARA architecture (flavor #2)

Graphical user interface, diagram

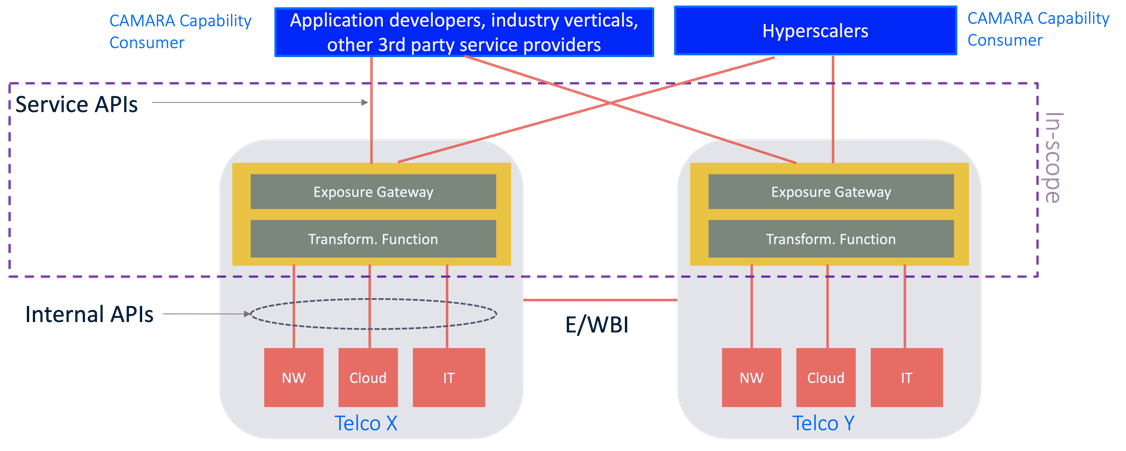
Description automatically generated

Figure 4: Example of realization of CAMARA architecture (flavor #3)

The focus of the present document will be on the Exposure Gateway. For this architecture component, it is proposed to use 3GPP Common API Framework (CAPIF) [1] as reference implementation solution. The objective of this document is to provide an overview of CAPIF capabilities (Section 2 and 3), and to illustrate their usability in CAMARA (Section 4). For readers interested in looking for further details, references and annexes have been also included in this document.

# 2. MOTIVATION

The need to expose telco capabilities to 3rd party applications were first conceived in 3GPP, with the definition of Service Capability Exposure Function (SCEF, Rel-13) [2] and Network Exposure Function (NEF, Rel-15) [3]. SCEF/NEF are 3GPP network functions that allow for secure exposure of EPC/5GC network capabilities, including subscription management (HSS/UDM), mobility management (MME/AMF), policy and charging management (PCRF/PCF) and communication (BM-SC). They act as an intermediation layer protecting the network infrastructure from direct access by the 3rd party consumers.

Having SCEF/NEF deploy in telco networks in is a necessary but not sufficient condition to offer the complete suite of capabilities that 3rd parties may want to consume. Indeed, SCEF/NEF scope is limited to expose capabilities from the core network. However, the operator managed assets also include resources/functions from:

* Other network domains: fixed access, access network, transport network (IP/MPLS, optical/DWDM and microwave backhauling) and data network (hosting value-added functions and services). For the exposure of capabilities from these domains, APIs such as those being defined in O-RAN, BBF, 3GPP and IETF might be used.
* Cloud domain, with IaaS/CaaS to host virtualized and cloud-native workloads, respectively. This domain is typically distributed across operator’s infrastructure footprint, following the edge-to-cloud continuum. For the exposure of capabilities from this domain, APIs such as those being defined in ETSI NFV, Linux Foundation’s CNCF or other de-facto solutions (e.g. Openstack, k8s) might be sued.
* IT domain, covering tools used in OSS/BSS and NOC systems, taking care of network and service management aspects (FCAPS management, orchestration, topology & inventory management, etc.). For the exposure of capabilities from this domain, APIs such as those being defined in 3GPP, ETSI and TMF might be used.

All the APIs referenced above correspond to internal APIs in CAMARA initiative. These APIs need to be abstracted into service APIs that are suitable for 3rd party consumption. However, for this exposure, there is a need to implement a set of support/common capabilities such as *onboarding, authentication and authorization, discovery, auditing, accounting*, to name a few. These capabilities, to be provided by the Exposure Gateway, allow policing the interaction between the API provider and consumer, when the two entities belong to non-trusted domains. This is what happens precisely in CAMARA initiative, where the operator (acting as API provider) and the 3rd party (acting as API consumer) are defined in different administrative domains.

For the sake of consistent testing and validation activities in CAMARA, it is important to look for a reference solution for the Exposure Gateway. In this regard, it is considered the use of 3GPP defined Common API Framework (CAPIF) solution, which provides the abovementioned support/common capabilities while easing the integration of 3rd party applications to operator assets. Figure 5 illustrates the role of CAPIF in helping in this integration.

Diagram

Description automatically generated

Figure 5: The role of CAPIF.

One of the main advantages of CAPIF solution is that though specified by 3GPP, it is not tied to 3GPP APIs; indeed, CAPIF can be used as Exposure Gateway solution for any API, no matter API semantics. This property, together with the fact that CAPIF is a normative solution (defined by an SDO) with wide acceptance at industry, makes CAPIF an ideal reference solution for Exposure Gateway in CAMARA.

# 3. CAPIF architecture

The CAPIF is organized into functional entities to describe a functional architecture which enables an API invoker to access and invoke service APIs. Figure 1 shows the reference point based functional model for the CAPIF. In this model:

* The CAPIF is hosted within the PLMN Operator (MNO) network. The PLMN Operator acts an CAPIF provider.
* The API invoker is typically provided by a 3rd party application provider who has service agreement with the PLMN Operator.
* The API invoker may or may not reside within the same trust domain as the PLMN Operator network. The API invoker within the PLMN trust domain interacts with the CAPIF via CAPIF-1 and CAPIF-2. The API invoker from outside the PLMN trust domain interacts with the CAPIF via CAPIF-1e and CAPIF-2e.
* The API exposing function, the API publishing function and the API management function of the API provider domain (together known as API provider domain functions) within the PLMN trust domain interacts with the CAPIF core functions via CAPIF-3, CAPIF-4 and CAPIF-5, respectively.



Figure 4: CAPIF architecture

In the following, we provide details on the CAPIF functional entities (Section 3.1) and interfaces connecting them (Section 3.2).

**3.1 CAPIF functional entities**

The CAPIF is structured into the following entities:

* **CAPIF Core Function (CCF).** It acts as an orchestrator that manages the interaction between service consumers (vertical apps) and service providers (e.g., NEF, SEAL). The main responsibilities of CCF are authentication of the API invoker, authorization of the API invoker to access the available service APIs, monitoring the service API invocations.
* **API Invoker**. It represents the vertical app which consumes the service APIs utilizing CAPIF. API Invoker provides to the CCF the required information for authentication, discovers and then invokes the available service APIs.
* **API Exposing Function (AEF)**. It is responsible for the exposure of the service APIs. Assuming that API Invokers are authorized by the CCF, AEF validates the authorization and subsequently provides the direct communication entry points to the service APIs. AEF may also authorize API invokers and record the invocations in log files.
* **API Publishing Function (APF).** It is responsible for the publication of the service APIs to CCF in order to enable the discovery capability to the API Invokers.
* **API Management Function (AMF)**. It supplies the API provider domain with administrative capabilities. Some of these capabilities include, auditing the service API invocation logs received from the CCF, on-boarding/off-boarding new API invokers and monitoring the status of the service APIs.

3GPP considers two main architectural deployment models, centralized, when the CCF and API Provider domain functions are co-located, and distributed (Figure 1), when CCF and API Provider domain functions are not co-located and they are interacting through CAPIF-3/4/5 interfaces.

**3.2 CAPIF interfaces**

Interfaces defined in CAPIF architecture are as follows:

* **CAPIF-1/1e**. API Invoker and CCF interact over CAPIF-1/1e interfaces supporting authentication and authorization of API Invokers, discovery of service APIs, and onboarding / off boarding of the API invokers. CAPIF-1 and CAPIF-1e interfaces are used when API invoker is within and outside of PLMN trust domain, respectively.
* **CAPIF-2/2e**. API Invoker and the AEF interact over CAPIF-2/2e interfaces supporting authentication and authorization of API Invoker and service API invocations by the API Invoker. CAPIF-2 and CAPIF-2e interfaces are used when API invoker is within and outside of PLMN trust domain respectively.
* **CAPIF-3**. AEF interacts over the CAPIF-3 interface for enforcing access and policy related controls for service API invocations initiated by the API Invoker.
* **CAPIF-4**. APF interacts over CAPIF-4 interface for publishing and un-publishing of service API information on CCF.
* **CAPIF-5**. AMF interacts over CAPIF-5 interface or management of service APIs, API invoker and API provider domain function information, onboarding / offboarding of API provider domain functions.

**3.3 CAPIF security**

CAPIF security requirements, architecture and procedures are specified by 3GPP in TS 33.122 [2]. In this section, we summarize the main findings that are relevant for CAMARA.

* + 1. **Interface protection**

The key security requirements for all the CAPIF interfaces are mutual authentication, integrity, replay and confidentiality protection. CAPIF-1/2/3/4/5 interface’s protection is based on Transport Layer Security (TLS) as specified in IETF RFC 5246. TLS is used based on the domain administrator’s policy to protect these interfaces within the trusted domain.

For integrity, replay and confidentiality protection of the CAPIF-1/-1e interface, CCF and API Invoker establish TLS session based on certificate based mutual authentication. Further details are captured in Section 3.3.2.

The API Invoker and the AEF apply the security method selected by the CCF, for authentication, authorization and the protection of CAPIF-2e interface. Further details are captured in Section 3.3.3.

* + 1. **CAPIF-1/-1e authentication and authorization**

The following considerations apply over the CAPIF-1/-1e interface.

* **Security method negotiation**: the API invoker and the CAPIF core function shall negotiate a security method that shall be used by the API invoker and the API exposing function for CAPIF-2e interface authentication and protection. After successful mutual authentication on CAPIF-1e interface, based on the API invoker's subscribed service APIs, access scenarios (whether the API invoker access the AEF prior to service API invocation or upon the service API invocation) and AEF capabilities, the CAPIF core function shall choose the security method and sends the chosen security methods along with the information required for authentication of the API invoker at the AEF to the API invoker. The information may include the validity time of the CAPIF-2e credentials.
* **API discovery**: after successful authentication between API invoker and CCF, the CCF shall decide whether the API invoker is authorized to perform discovery based on API invoker ID and discovery policy.
* **Topology hiding**: when topology hiding is enabled, the CCF shall respond to service APIs discovery requests with AEF information, which exposes the service API and acts as topology hiding entity.
  + 1. **CAPIF-2/-2e authentication and authorization.**

Based on the selected security method by the CCF (see Section 3.3.2), one of the following methods shall be used by the API invoker and the AEF for CAPIF-2/-2e interface authentication and authorization.

* **TLS-PSK**: API Invoker and AEF establish dedicated secure session using TLS based on Pre-Shared Key (PSK) authentication. CAPIF-1/-1e authentication is used to bootstrap a PSK for authenticating a TLS connection for CAPIF-2/-2e. AEF obtains the PSK from CCF over CAPIF-3 interface.
* **TLS-PKI**: API Invoker and AEF establish dedicated secure session, using TLS based on certificate based mutual authentication.
* **TLS with OAuth**: API invoker authenticates with the AEF by establishing a TLS session with the AEF based Server (AEF) side certification authentication or certificate-based mutual authentication. AEF uses the access token based on OAuth 2.0 authorization framework as specified in IETF RFC 6749, to authorize API Invoker’s Northbound API invocations. Client credentials grant is the supported mechanism by API Invoker and CCF for OAuth 2.0 access token request and issuance.

# 4. CAPIF services

The concept of *CAPIF services* represents the set of capabilities which are made available for consumption to entities utilizing the CAPIF architecture. These capabilities are offered through northbound APIs with the following properties:

* Extensibility, such that it is possible to accommodate future requirements
* Access control support
* Charging support
* Backwards/Forwarding compatibility, therefore guaranteeing compatibility with different versions of the same API
* RESTful compliant, using CRUD operations. Service operations can use custom API operations (RPC-style interaction), when it is seen a better fit for the style of interaction to model, e.g. non-CRUD service operation.

The available CAPIF services and their respective APIs are listed in Table 1. According to their semantics, CAPIF services can be arranged into four CAPIF categories: common services, security services, management services and internal connectivity services. Annex A provides a deep dive on these categories.

Table 1: CAPIF services

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **CAPIF services (3GPP TS 23.222 clause)** | **CAPIF interface** | **CAPIF APIs1 (3GPP TS 23.222 clause)** |
| Common services | Publishing service API (clause 8.3) | CAPIF-4 | CAPIF\_Publish\_Service\_API (clause 10.3) |
| Unpublishing service API (clause 8.4) |
| Retrieving service API (clause 8.5) |
| Updating service API (clause 8.6 |
| Discovering service API (clause 8.7) | CAPIF-1/-1e | CAPIF\_Discover\_Service\_API (clause 10.2) |
| Management services | Logging service API invocation (clause 8.19) | CAPIF-3 | CAPIF\_Logging\_API\_Invocation (clause 10.8) |
| Charging service API invocation (clause 8.20) | N/A2 |
| Monitoring service API invocation (clause 8.21) | CAPIF-4 | CAPIF\_Monitoring (clause 10.7) |
| Auditing service API invocation (clause 8.22) | CAPIF-5 | CAPIF\_Auditing (clause 10.9) |
| Security services | Onboarding API invoker (clause 8.1) | CAPIF-1/-1e | CAPIF\_API\_invoker\_management (clause 10.5) |
| Off boarding API invoker (clause 8.2) |
| Authenticating with CCF (clause 8.10) | CAPIF-1/-1e | CAPIF\_Security (clause 10.6) |
| Authorizing with CCF (clause 8.11) |
| Authenticating with AEF (clauses 8.14 and 8.15) | CAPIF-2/CAPIF-2e | AEF\_Security API (clause 11.2). |
| Authorizing with AEF (clause 8.16) |
| Access control policy (clause 8.17) | CAPIF-2/-2e/3 | CAPIF\_Access\_Control\_Policy (clause 10.10) |
| Registering API provider (clause 8.28) | CAPIF-5 | CAPIF\_API\_provider\_management (clause 10.12) |
| Deregistering API provider (clause 8.30) |
| Updating API provider (clause 8.29) |
| Internal connectivity services4 | Dynamic routing (clause 8.27)5 | CAPIF-1/-1e/-2/-2e/3 | CAPIF\_Routing\_Info (clause 10.11) |
| NOTE 1: CAPIF service APIs include CCF APIs (3GPP TS 28.322, clause 10.x) and AEF APIs (3GPP TS 28.322, clause 11.x).  NOTE 2: This API is within the scope of SA5 WG, particularly within the SA5\_CH (charging subgroup). Details on the role of CAPIF in charging service API invocations are specified in Annex B.  NOTE 3: For simplicity, event subscription/un-subscription and notification services (3GPP TS 28.322, clause 8.8) have not included. These CAPIF services apply to the following CAPIF interfaces:   * CAPIF-1 or CAPIF-1e. The API invoker can subscribe to and unsubscribe from CAPIF events and receive notifications from the CCF. * CAPIF-3. The AEF can subscribe to and unsubscribe from CAPIF events and receive notifications from the CCF. * CAPIF-4. The APF can subscribe to and unsubscribe from CAPIF events and receive notifications from the CCF. * CAPIF-5. The AMF can subscribe to and unsubscribe from CAPIF events and receive notifications from the CCF.   NOTE 4: For simplicity, the CCF interconnection (3GPP TS 28.322, clause 8.25) have been omitted. CCF interconnection represents the CAPIF capability of setting up communication between CCF instances. This communication is conveyed and managed through the CAPIF-6 interface (if CCF instances are all within the same PLMN trust domain) and CAPIF-6e interface (if CCF instances are from different PLMN trust domains).  NOTE 5: This service requires CAPIF having topology hiding (3GPP TS 28.322, clause 8.13). This CAPIF capability was presented in Section 3.3.2. The CAPIF supports API topology hiding by dynamically configuring the address of the AEF providing the service API to the AEF entry point providing the topology hiding. | | | |

Figure 5 shows some of these services, and how they impact the different CAPIF actors, including the API invoker, the CCF and the API provider domain functions (AEF, APF and AMF). This figure is only provided for illustration purposes, and does not represent the order of the operations.



Figure 5: Overview of CAPIF service

# 4. USE OF CAPIF in CAMARA

Figure 6 shows how CAPIF components are mapped into CAMARA reference architecture (see Figure 1).

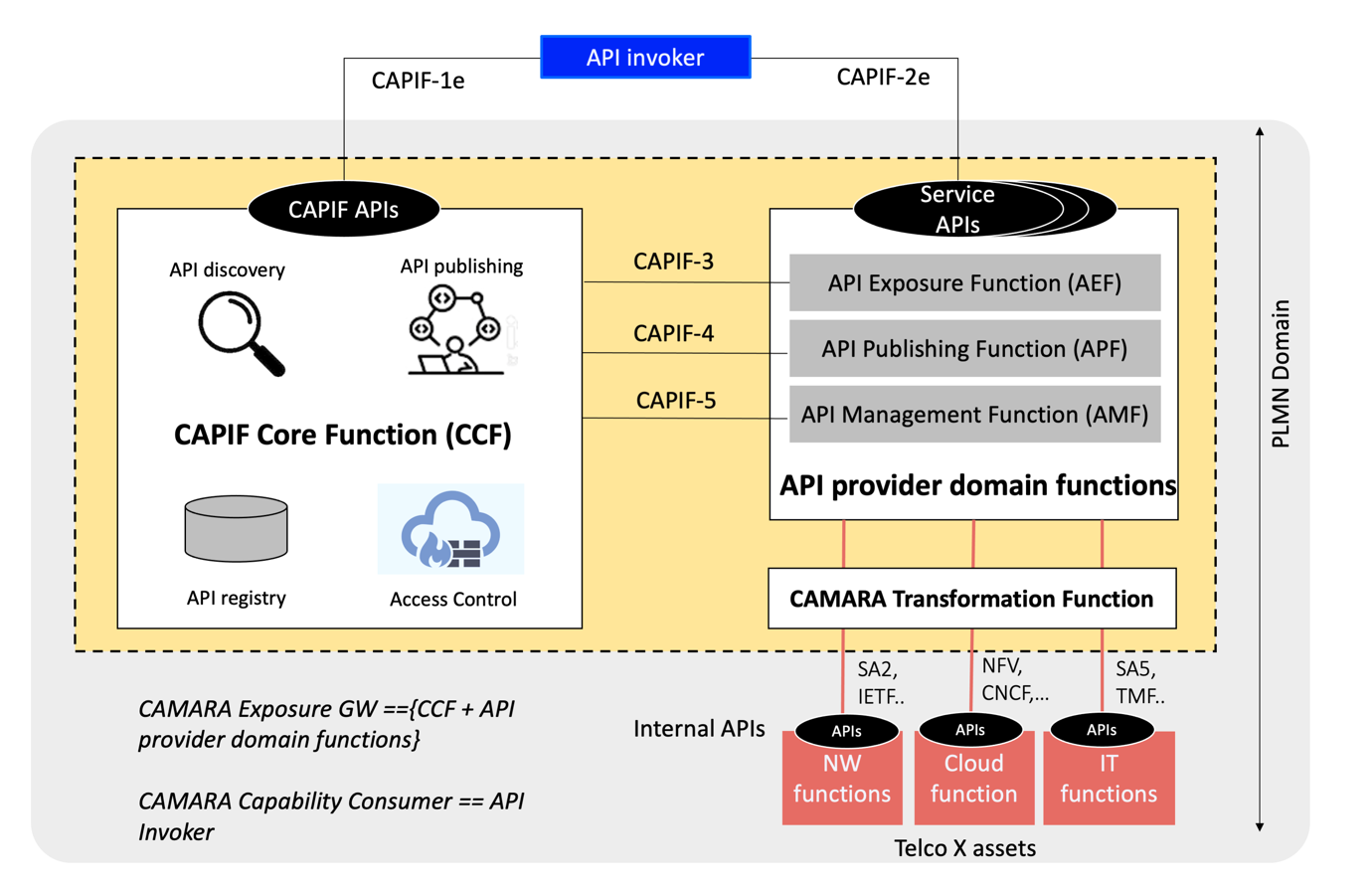


Figure 6: Mapping CAPIF solution in CAMARA

# 5. REFERENCES

1. 3GPP TS 23.222, “Common API Framework for 3GPP Northbound APIs”
2. 3GPP TS 29.122, “T8 reference point for northbound APIs”
3. 3GPP TS 29.522, “Network Exposure Function Northbound APIs; Stage 3”

# ANNEX a: details on CAPIF services

**A.1 Common services**

This category includes the following CAPIF services.

**A.1.1 Publish service API**

The CAPIF supports publishing service APIs by the API provider. The APF consumes this service to publish a service API to the CCF. The publication includes details about the specific service API. When a publication occurs, CCF registers all the related information in an internal repository called API registry.



**A.1.2 Unpublish service API**

The CAPIF supports un-publishing service APIs by the API provider. The APF consumes this service to unpublish a service API from the CCF.



**A.1.3 Update service API**

The CAPIF supports updating service APIs by the API provider. The APF consumes this service to update the information related to the published service APIs, e.g. a change in the characteristics of the API. This procedure is initiated by the APIF to the CC.



**A.1.4 Retrieve service API**

APF requests from CCF information related to previous published services.



**A.1.5 Discovery service API**

This service enables API Invokers to retrieve the available services that have been registered in CCF, i.e. the services available in the CCF hosted API registry.



**A.2 Management services**

This category includes the following CAPIF services.

**A.2.1 Logging service API invocation**

Upon invocations (i.e., from API Invokers), CCF may store valuable information such as API Invoker’s ID, IP address, service API name. AEF utilizes the logging service API invocation to access the potential logs files that have been stored in CCF.



**A.2.2 Auditing service API invocation**

This service can be used to control CAPIF interactions with API Invokers (e.g., invocation events, onboarding events, authentication), which are stored in CCF. AMF initiates a request to fetch the respective logs.



**A.2.3 Charging service API invocation**

AEF can use this service to retrieve charging related information flows from the CCF.



**A.2.4 Monitoring service API invocation**

Monitoring event service is used by AMF in order to get notified whether an event occurs in the CCF. Some of the events are the availability of service APIs (e.g. active, inactive), changes in service APIs (e.g. after and update), service API invocations, API invoker status (e.g. onboarded, offboarded) and performance related events (e.g., load conditions).



**A.3 Management services**

This category includes the following CAPIF services.

**A.3.1 Authenticating with CCF**

This CAPIF service allows an API invoker to be authenticated from the CCF. In this option, invoker initiates a direct authentication request to the CCF using CAPIF-1/-1e interface.

In the step 2 of the workflow below, the security negotiation procedure specified in Section 2.2.2 occurs.

****

**A.3.2 Authorizing with CCF**

The API invoker requires to execute this procedure when it needs to obtain or re-obtain (e.g. upon expiry of the authorization information) the authorization to access the service API. Once the API invoker receives the authorization to access the service API, the API invoker can perform one or multiple service API invocations as per the permission limit.

In the step 2, the CCF validates the authentication of the API invoker (using authentication information agreed in Section A.3.1) and checks whether the API invoker is permitted to access the requested service API. If authorized, the CCF issues the OAuth 2.0 access token to the API invoker.



**A.3.3 Authenticating with AEF**

This CAPIF service allows an API invoker to be authenticated from the AEF. To reduce latency during API invocation, the API invoker associated authentication information can be made available at the AEF after authentication between the API invoker and the CCF. Unlike the authentication with CCF service (see section A.3.1), in this new CAPIF service the AEF authenticates an invoker with assistance from CCF.

This authentication can be made prior to the service API invocation or upon the service invocation.





**A.3.4 Authorizing with AEF**

After authentication occurs, API invokers initiate request to retrieve service APIs. AEF checks whether the invoker is authorized to do so. If the AEF does not have the required information, AEF inquiries CCF. Thus, AEF and CCF can invalidate invoker’s configured authenticated at any moment.



**A.3.5 Access control policy**

The CAPIF service enables AEF to obtain the configured policies to perform access control on the service API invocations.



**A.3.6 Registering API provider**

This CAPIF service enables AMF to register the API provider domain functions to CCF in order to be authorized and use CAPIF’s functionalities.



**A.3.7 Deregistering API provider**

This CAPIF service enables AMF to de-register one (or more) existing API provider function(s) from CCF.



**A.3.8 Updating API provider**

This CAPIF service enables AMF to update the registration of API provider domain function(s) to CCF.



**A.3.9 Onboarding API invoker**

This CAPIF service enables a one-time onboarding process that enrolls an API invoker as a recognized user of the CAPIF. Invokers initiate the on-boarding process by sending a request to the CCF. If the enrolment information provided is valid, CCF onboards the invoker and creates a new profile, which is sent back upon the response.



**A.3.10 Offboarding API invoker**

This CAPIF service enables offboarding the API invoker from the CAPIF. The offboarding process makes the API Invoker no longer a recognized user of the CAPIF.

**A.4 Internal connectivity services**

**A.4.1 Dynamic routing**

This feature requires the topology hiding capability to be activated. This capability enables hiding the topology of the PLMN trust domain from the API invokers accessing the service APIs from outside the PLMN trust domain. In this case, API invokers access an AEF which acts as an entry point. Thus, the information for the entry AEF is shared with API invoker in the discovery service. Then, subsequently, AEF resolves the actual destination address of the requested service API and forwards the initial request.



# ANNEX B: CAPIF relationship with other api frameworks

This annex provides the relationship of CAPIF with the OMA Network APIs and the ETSI MEC API framework. The relationship of CAPIF with these external API frameworks is illustrated in the table D-1. "Yes" means that the external API framework supports the CAPIF functionality, "No" means that the API framework does not support the CAPIF functionality, and "Partial" means that it provides a mechanism that partially supports the CAPIF functionality.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CAPIF functionalities** | **OMA Network APIs** | | **ETSI MEC API framework** | |
| **Supported** | **Reference** | **Supported** | **Reference** |
| Publish and discover service API information | Partial | OMA-TS-NGSI\_Registration\_and\_Discovery | Yes | ETSI GS MEC 011 [7] |
| Topology hiding of the service | Yes | Individual API exposing function | Yes | Individual API exposing function |
| API invoker authentication to access service APIs | Partial | OMA-ER\_Autho4API | Partial | ETSI GS MEC 009 [8] |
| API invoker authorization to access service APIs | Partial | OMA-ER\_Autho4API | Partial | ETSI GS MEC 009 |
| Charging on invocation of service APIs | No |  | No |  |
| Lifecycle management of service APIs | No |  | No |  |
| Monitoring service API invocations | No |  | No |  |
| Logging API invoker onboarding and service API invocations | No |  | No |  |
| Auditing service API invocations | No |  | No |  |
| Onboarding API invoker to CAPIF | No |  | No |  |
| CAPIF authentication of API invokers | No |  | No |  |
| Service API access control | Partial | OMA-ER\_Autho4API | Partial | ETSI GS MEC 009 |
| Secure API communication | Yes | OMA-ER\_Autho4API | Yes | ETSI GS MEC 009 |
| Policy configuration | No |  | No |  |
| API protocol stack model | Partial | for REST: OMA-TS\_REST\_NetAPI\_Common | Partial | for REST:  ETSI GS MEC 009 |
| API security protocol | Partial | OMA-ER\_Autho4API | Partial | ETSI GS MEC 009 |
| CAPIF support for service APIs from multiple providers | No |  | No |  |