Release H - CAPIF

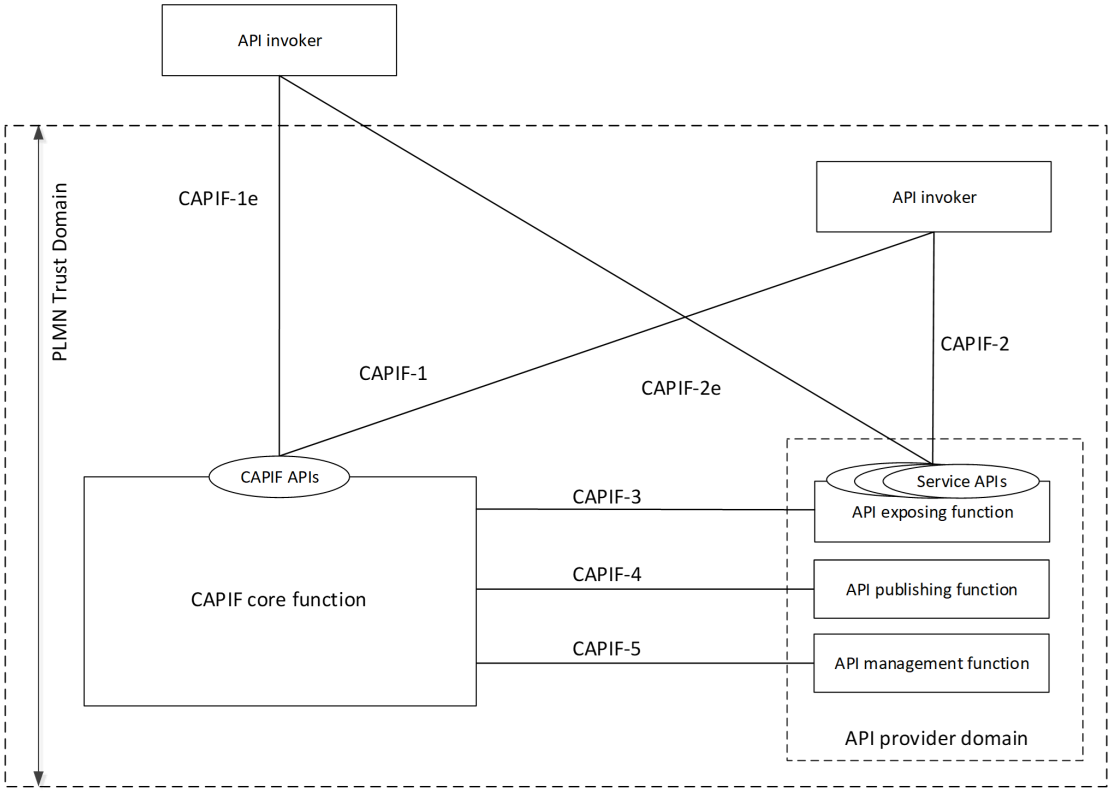
- Functional entities
  - CAPIF core function
  - API invoker
  - API provider domain
    - API exposing function
    - API publishing function
    - API management function
- CAPIF core function APIs
  - Examples to use CAPIF core function APIs
    - Register provider domain
    - Publish a new API
    - Onboard API invoker
- Security in CAPIF
  - Security procedures for API invoker onboarding
    - Security method negotiation
- Build CAPIF core
- Run CAPIF core

CAPIF stands for common API framework and it was developed by 3GPP to enable a unified Northbound API framework across 3GPP network functions, and to ensure that there is a single and harmonized approach for API development.

Among its key features are:

- Onboarding/offboarding API invoker
- Discovery APIs
- Register/deregister APIs
- CAPIF events subscription/notification
- Entity authentication/authorization
- Support for 3rd party domains i.e., allow 3rd party API providers to leverage the CAPIF framework
- Support interconnection between two CAPIF providers

Functional entities



CAPIF core function

The CAPIF core function 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;
- Monitoring the service API invocations;
- Onboarding a new API invoker and offboarding an API invoker;
- Supports publishing, discovery of service APIs information with another CAPIF core function in CAPIF interconnection.

API invoker

The API invoker is the entity which invokes the CAPIF or service APIs, typically provided by a 3rd party application provider who has service agreement with PLMN operator.

The API invoker supports the following capabilities:

- Triggering API invoker onboarding/offboarding;
- 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.

API provider domain

API exposing function

The entity which provides the service communication entry point for the service APIs. Provides access control, logging, charging, provides authentication and authorization support.

API publishing function

The entity that enables the API provider to publish the Service APIs information in order to enable the discovery of APIs by the API invoker.

API management function

The entity which registers and maintains registration information of the API provider domain functions on the CAPIF core function

CAPIF core function APIs

The table below lists the CAPIF Core Function APIs that are currently implemented:

Service Name	Service Operations	Operation Semantics	Consumer(s)
CAPIF_Discover_Service_API	Discover_Service_API	GET /allServiceAPIs	API Invoker, CAPIF core function
CAPIF_Publish_Service_API	Publish_Service_API	POST /{apfId}//service-apis	API Publishing Function, CAPIF core function
	Unpublish_Service_API	DELETE /{apfId/service-apis/{serviceApiId}	API Publishing Function, CAPIF core function
	Update_Service_API	PUT /{apfId/service-apis/{serviceApiId}	API Publishing Function, CAPIF core function
	Get_Service_API	GET /{apfId}/service-apis	API Publishing Function, CAPIF core function
CAPIF_API_Invoker_Management_API	Onboard_API_Invoker	POST /onboardedInvokers	API Invoker
	Offboard_API_Invoker	DELETE /onboardedInvokers/ {onboardingId}	API Invoker
	Notify_Onboarding_Completion	Subscribe/Notify	API Invoker
	Update_API_Invoker_Details	PUT /onboardedInvokers/{onboardingId}	API Invoker

	Notify_Update_Completion	Subscribe/Notify	API Invoker
CAPIF_Security_API	Obtain_Security_Method	PUT /trustedInvokers/{apiInvokerId}	API Invoker
	Obtain_Authorization	POST /securities/{securityId}/token	API Invoker
	Obtain_API_Invoker_Info	GET /trustedInvokers/{apiInvokerId}	API exposing function
	Revoke_Authorization	DELETE /trustedInvokers/{apiInvokerId}	API exposing function
CAPIF_API_Provider_Management_API	Register_API_Provider	POST /registrations	API Management Function
	Update_API_Provider	PUT /registrations/{registrationId}	API Management Function
	Deregister_API_Provider	DELETE /registrations/{registrationId}	API Management Function

Examples to use CAPIF core function APIs

Register provider domain

POST http://<CAPIF core URL>/api-provider-management/v1/registrations

Request body: APIProviderEnrolmentDetails

APIProviderEnrolmentDetails example

```
{
    "apiProvDomInfo": "Provider domain",
    "apiProvFuncs": [
        {
            "apiProvFuncInfo": "rApp as APF",
            "apiProvFuncRole": "APF",
            "regInfo": {
                "apiProvPubKey": "APF-PublicKey"
            }
        },
        {
            "apiProvFuncInfo": "rApp as AEF",
            "apiProvFuncRole": "AEF",
            "regInfo": {
                "apiProvPubKey": "AEF-PublicKey"
            }
        },
        {
            "apiProvFuncInfo": "rApp as AMF",
            "apiProvFuncRole": "AMF",
            "regInfo": {
                "apiProvPubKey": "AMF-PublicKey"
            }
        },
        {
            "apiProvFuncInfo": "Gateway as entrypoint AEF",
            "apiProvFuncRole": "AEF",
            "regInfo": {
                "apiProvPubKey": "AEF-Gateway-PublicKey"
            }
        }
    ],
    "regSec": "PSK"
}
```

Publish a new API

POST http://<CAPIF core URL>/published-apis/v1/APF_id_rApp_as_APF/service-apis

Request body: ServiceAPIDescription

ServiceAPIDescription - example


```
{
    "apiName": "example A",
    "description": "Example A API of rApp",
    "aefProfiles": [
        {
            "aefId": "AEF_id_rApp_as_AEF",
            "description": "Example A rApp as AEF",
            "versions": [
                {
                    "apiVersion": "v1",
                    "resources": [
                        {
                            "resourceName": "exampleA",
                            "commType": "REQUEST_RESPONSE",
                            "uri": "/exampleA/subscription/subscription_id_1",
                            "operations": [
                                "GET"
                            ]
                        }
                    ]
                }
            ],
            "protocol": "HTTP_1_1",
                        "securityMethods": ["PSK"],
                        "interfaceDescriptions": [
                                {
                                  "ipv4Addr": "string",
                                  "port": 65535,
                                  "securityMethods": ["PKI"]
                                },
                                {
                                  "ipv4Addr": "string",
                                  "port": 65535,
                                  "securityMethods": ["PKI"]
                                }
                          ]
        }
    ]
}
```


Onboard API invoker

POST http://<CAPIF core URL>/api-invoker-management/v1/onboardedInvokers

Request body: APIInvokerEnrolmentDetails

APIInvokerEnrolmentDetails - example


```
{
    "apiInvokerInformation": "rApp as invoker 1",
         "apiList": [
                {}
        ],
    "NotificationDestination": "http://invoker-app:8086/callback",
    "onboardingInformation": {
                "apiInvokerPublicKey": "{PUBLIC_KEY_INVOKER_1}",
                "apiInvokerCertificate": "apiInvokerCertificate"
  },
  "requestTestNotification": true
}
```

More examples can be found in the postman collection below:

Postman collection

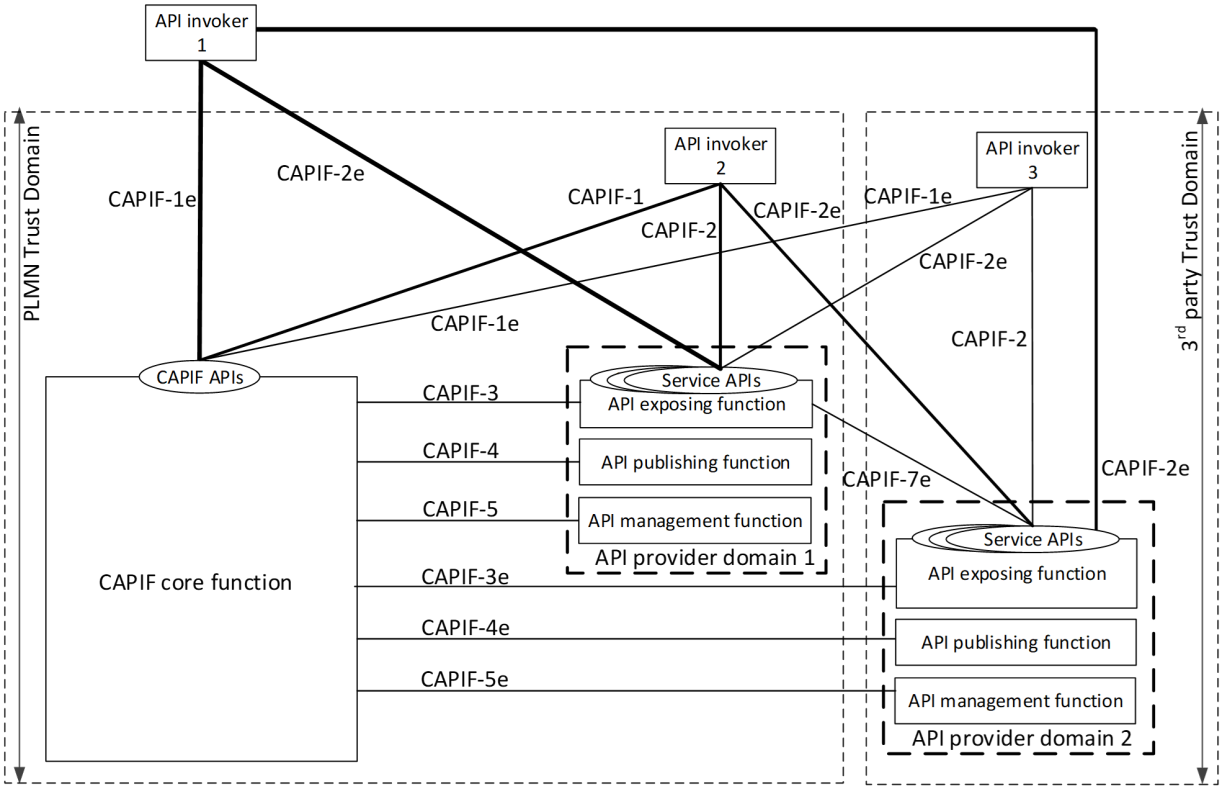
Security in CAPIF

Security related information about CAPIF can be found in 3GPP TS 33.122 Security aspects of Common API Framework (CAPIF).

CAPIF establish security requirements for all the interfaces defined in the specification. There are also security requirements that are applicable to all CAPIF entities, such as:

- CAPIF shall provide mechanisms to hide the topology of the PLMN trust domain from the API invokers accessing the service APIs from outside the PLMN trust domain.
- CAPIF shall provide mechanisms to hide the topology of the 3rd party API provider trust domain from the API invokers accessing the service APIs from outside the 3rd party API provider trust domain.
- CAPIF shall provide authorization mechanism for service APIs from the 3rd party API providers.
- CAPIF shall support a common security mechanism for all API implementations to provide confidentiality and integrity protection.

The image below shows the functional security model for CAPIF architecture. CAPIF-1, CAPIF-2, CAPIF-3, CAPIF-4, CAPIF-5 and CAPIF-7 are interfaces that lie within the PLMN trust domain while the CAPIF-1e , CAPIF-2e, CAPIF-3e, CAPIF-4e, CAPIF-5e and CAPIF-7e interfaces are CAPIF core and AEF access points for API Invokers outside of the PLMN trust domain.



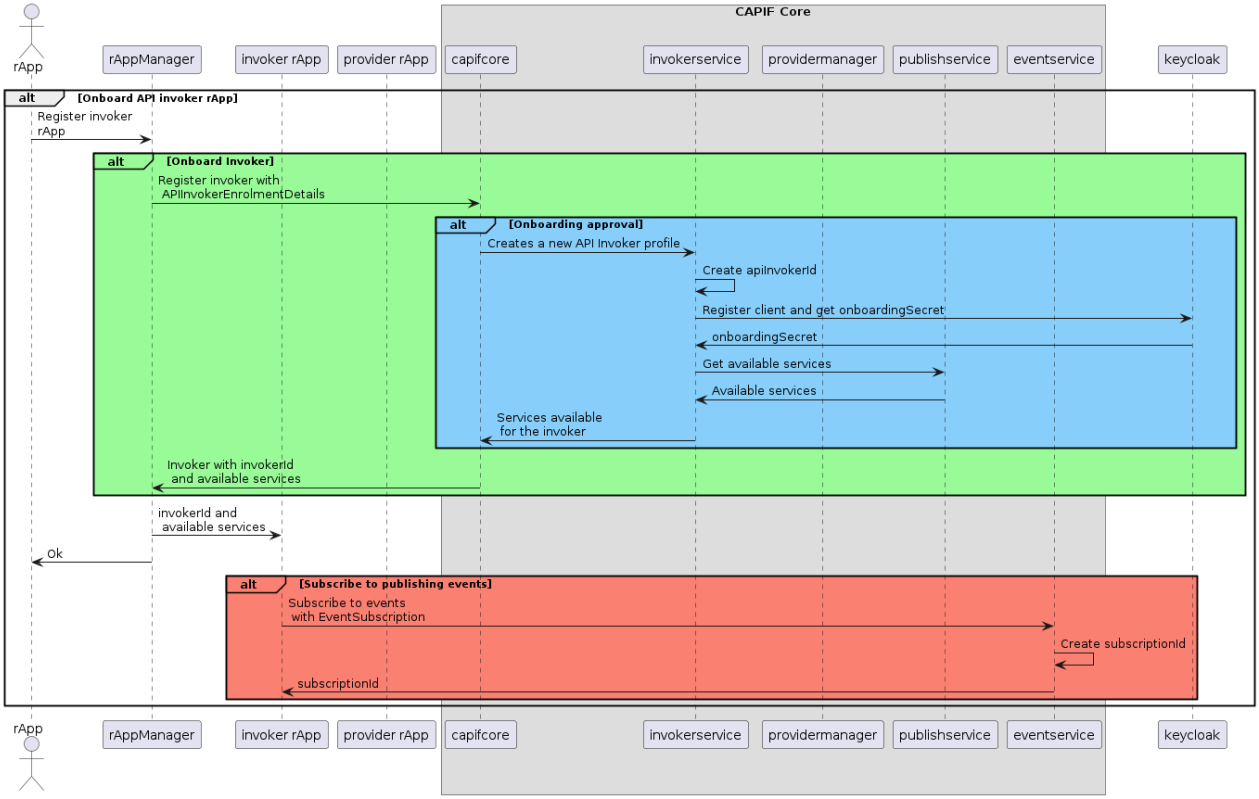
Authentication and authorization are required for both API invokers that lie within the PLMN trust domain and API invokers that lie outside of the PLMN trust domain. For an API invoker that is outside of the PLMN trust domain, the CAPIF core function in coordination with the API exposing function utilizes the CAPIF-1e, CAPIF-2e and the CAPIF-3 interfaces to onboard, authenticate and authorize the API invoker prior to granting access to CAPIF services.

Security procedures for API invoker onboarding

The API invoker and the CAPIF core function shall establish a secure session.

With a secure session established, the API Invoker sends an Onboard API Invoker Request message to the CAPIF core function. The Onboard API Invoker Request message carries an onboard credential obtained during pre-provisioning of the onboard enrolment information.

The CAPIF core function shall respond with an Onboard API invoker response message. The response shall include the CAPIF core function assigned API invoker ID, API invoker's certificate and the API invoker Onboard_Secret (generated by the CAPIF core function provided by keycloak).



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.

As a pre-condition the API invoker must be onboarded with the CAPIF core function.

After successful mutual authentication on CAPIF-1e interface, the API invoker may send CAPIF-2/2e security capability information to the CAPIF core function in the Security Method Request message, indicating the list of security methods that it supports for each AEF.

The CAPIF core function shall select a security method to be used over CAPIF-2/2e reference point for each requested AEF, taking into account the information sent by the API invoker and send a Security Method Response message to the API invoker indicating the selected security method for each AEF.

Build CAPIF core

Download the following repos:

```
git clone "https://gerrit.o-ran-sc.org/r/nonrtric/plt/sme"
```

To build the application, go into the repo and run the following command:

```
cd sme/capifcore
go build
```

To run the unit tests for the application, run the following command:

```
go test ./...
```

The application can also be built as a Docker image, by using the following command. Use the version found in the file container-tag.yaml.

```
docker build -t o-ran-sc/nonrtric-plt-capifcore:<image-version> .
```

Run CAPIF core

The CAPIF core needs one configuration file for keycloak including information about keycloak host (url, port and authentication for admin user) and realms. Before using CAPIF API invoker management, an invoker realm must be created in keycloak. Make sure it is created before running CAPIF core. After creating the realm in keycloak, set the name in the keycloak.yaml configuration file.

keycloak.yaml

```
# Keycloak configurations
authorizationServer:
  host: "keycloak"
  port: "8080"
  admin:
    user: "admin"
    password: "secret"
  realms:
   master: "master"
   invokerrealm: "invokerrealm"
```

There are a number of environment variables that could be set when starting the application, however they have values by default so there is no need to set them unless specific values are needed.

- -port <port (default 8090)>
- -secPort <Secure port (default 4433)>
- -chartMuseumUrl <URL to ChartMuseum>
- -repoName <Helm repo name (default capifcore)>
- -loglevel <log level (default Info)>
- -certPath <Path to certificate>
- -keyPath <Path to private key>

To run the Core Function from the command line, run the following commands from this folder. For the parameter `chartMuseumUrl`, if it is not provided CAPIF Core will not do any Helm integration, i.e. try to start any Halm chart when publishing a service.

```
 ./capifcore [-port <port (default 8090)>] [-secPort <Secure port (default 4433)>] [-chartMuseumUrl <URL to
ChartMuseum>] [-repoName <Helm repo name (default capifcore)>] [-loglevel <log level (default Info)>] [-
certPath <Path to certificate>] [-keyPath <Path to private key>]
```

There is a docker compose file available that can be use to start CAPIF core together with Keycloak:

```
docker-compose up
```

NOTE! In the configuration file in configs/keycloak.yaml when running locally the host value must be set to localhost (Eg. host: "localhost") and when using docker-compose set value to keycloak (Eg. host:"keycloak")