MEC Application Development Document

Version 001

Revision History

Version Number	Date Released	Summary of Change
001	17th July 2020	First release

Contents

1. **Scope** 5
2. **Introduction** 5
3. **Different Stages in MEC System** 6
   3.1 Stage 1: Communication Between Client Application and Device Application 6
   3.2 Stage 2: MEC app on-boarding and MEC app Instantiation 7
      3.2.1. Fetching the App 7
      3.2.1. App Instantiation 7
      3.2.1. Back-propagating the application info to the Client App 7
4. **API Details** 8
   a) Interface between Client App and Device App 8
   4.1 GET – /mi/v1/app_list [app list parameters] 8
   4.2 POST – /mi/v1/app_context [app context parameters] 8
   4.3 PUT- /mi/v1 /app_context/{context_Id} [app context parameters] 8
   4.4 DELETE – /mi/v1/ app_Contexts/{Context_Id} 8
   b) Interface between MEC app and MEC platform 8
   Table 1: API Summary 9
   Table 2: JSON Message 9
5. *References* 10

List of Figures

Figure 1 : Application development overview introduced by MEC [2].....5
Figure 2 : Communication between Client app and Device app 6
Figure 3 : MEC app on-boarding and Instantiation.....7
Figure 4 : Message flow between MEC app. and Cloud app.**Error! Bookmark not defined.**

List of Tables

Table 1: API Summary.....9
Table 2: JSON Message.....9

1. Scope

The MEC system developed as part of the Indigenous 5G test-bed follows the ETSI architecture [1]. The broad guidelines for developing applications for ETSI-based MEC systems are provided in [2]. The current document provides the MEC app developers with the various APIs supported by the 5G test-bed and the related configuration aspects.

2. Introduction

The ETSI reference architecture for the MEC platform provided in shows the functional elements that comprise of the multi-access edge (MEC) system and the reference points (interfaces) between them. A MEC host usually deployed in the network edge, it contains MEC platforms and performs computation and storage of applications in VMs or containers. MEC platform provides a secure environment where MEC applications can via Restful APIs discover, advertise, consume and also offer services **(*see* Figure 1]).**

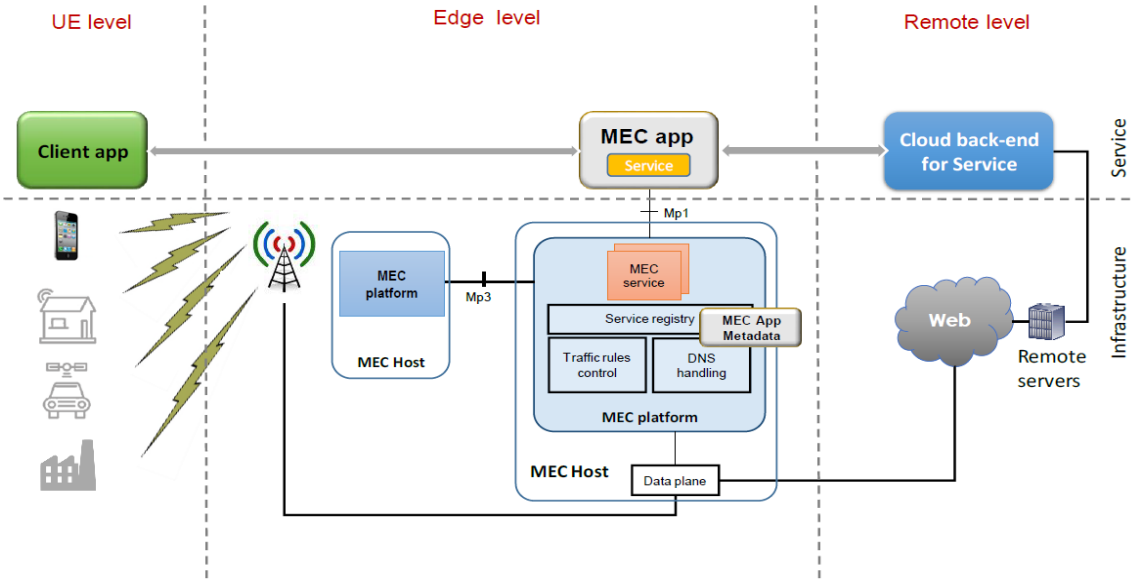


Figure 1: Application development overview introduced by MEC [2]

Please refer to [2] , for the broad guidelines for developing applications for ETSI-based MEC systems. Here we expect the developer to have knowledge of UE level, Edge Level, Cloud level of application processing. Applications are to be developed as *Client app*, *MEC app* and the *Cloud app* in correspondence to the provided details of the following stages:

- Stage 1: **Communication between Client App and Device App**
- Stage 2: **MEC app on-boarding and Instantiation**
    - ***Fetching the App***
    - ***App Instantiation***
    - ***Back-propagating the application info to the Client App***

All the phases of application in MEC include:
a. Solution on the basis of Domain name.
b. Services with cloud backend
c. Manage user context
d. Application Packaging and on-boarding
e. Provide data to fulfill application requirements

3. Different Phases in MEC System

3.1 STAGE 1: *Communication between Client Application and Device Application.*

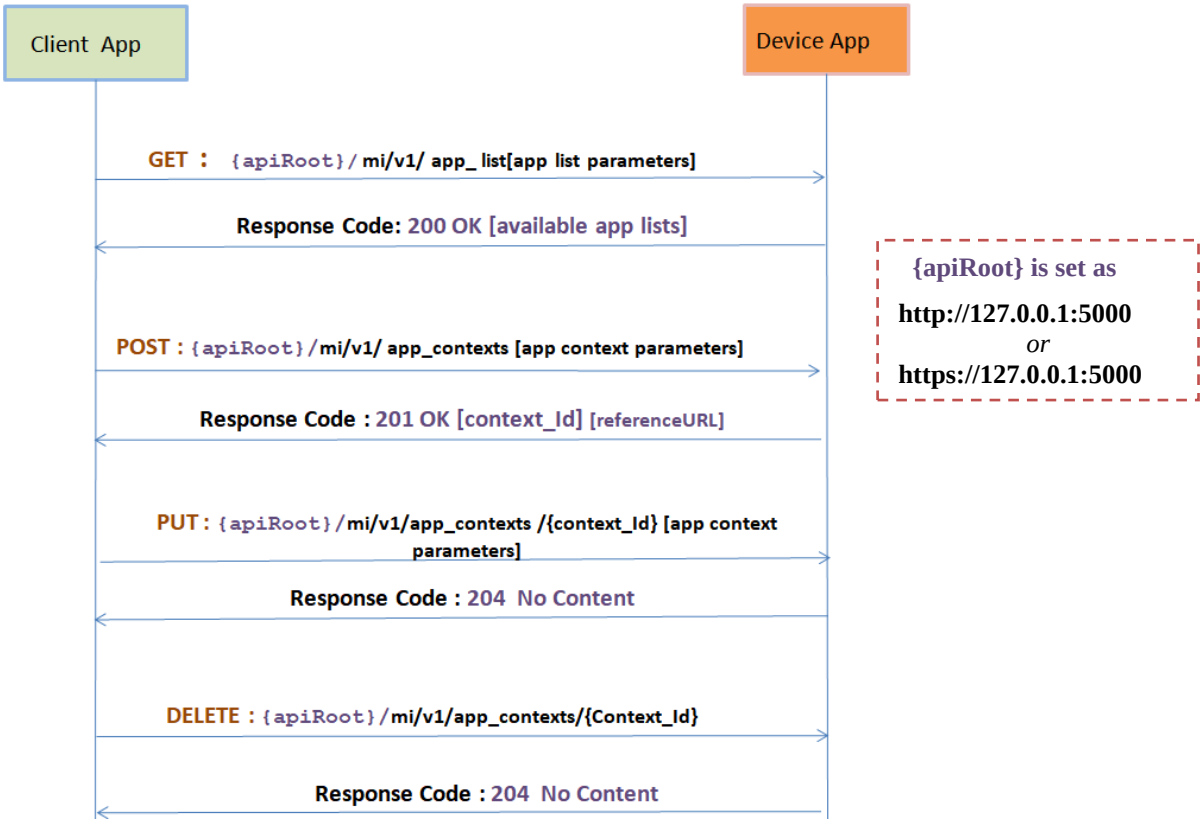


Figure 2: Communication between Client app and Device app

To establish a successful communication setup between the Client App and the MEC app instance, the initial communication from Client app goes through an interface, called *Device application*, which is needed for invoking the operations on subsequent interfaces on the MEC system to forward client requests to the MEC system. **Figure 2]** depicts the communication between client(s) to the Device application. It consists of the following points (***also see*** **Table 1]**).

- **a.** Send **GET** request to Device app to retrieve available app-list with given parameters **(*see* Table 2])** and receive response
- **b.** Send **POST** request to Device app for the creation of app context and receive response.
- **c.** Send **PUT** request to Device app if any change in callback reference, this may be optional.
- **d.** Send **DELETE** request Device app for termination of running application.

More Details about the above points are available in **Section [0]** of this document. For insightful details about parameters ***see* Table 2].** Client application(s) and the device application, as well as the relevant MEC system components and the above ways how the client-side application components can communicate with their MEC side peers (***see* [2]**). A device application is logically separate from the actual client application which is the one requesting services from the MEC application. Any end user device may have the client application installed? A client application most often is unaware of the edge deployment of the server application (***see*** [4]).

3.2 **STAGE 2:** MEC App on-boarding and Instantiation

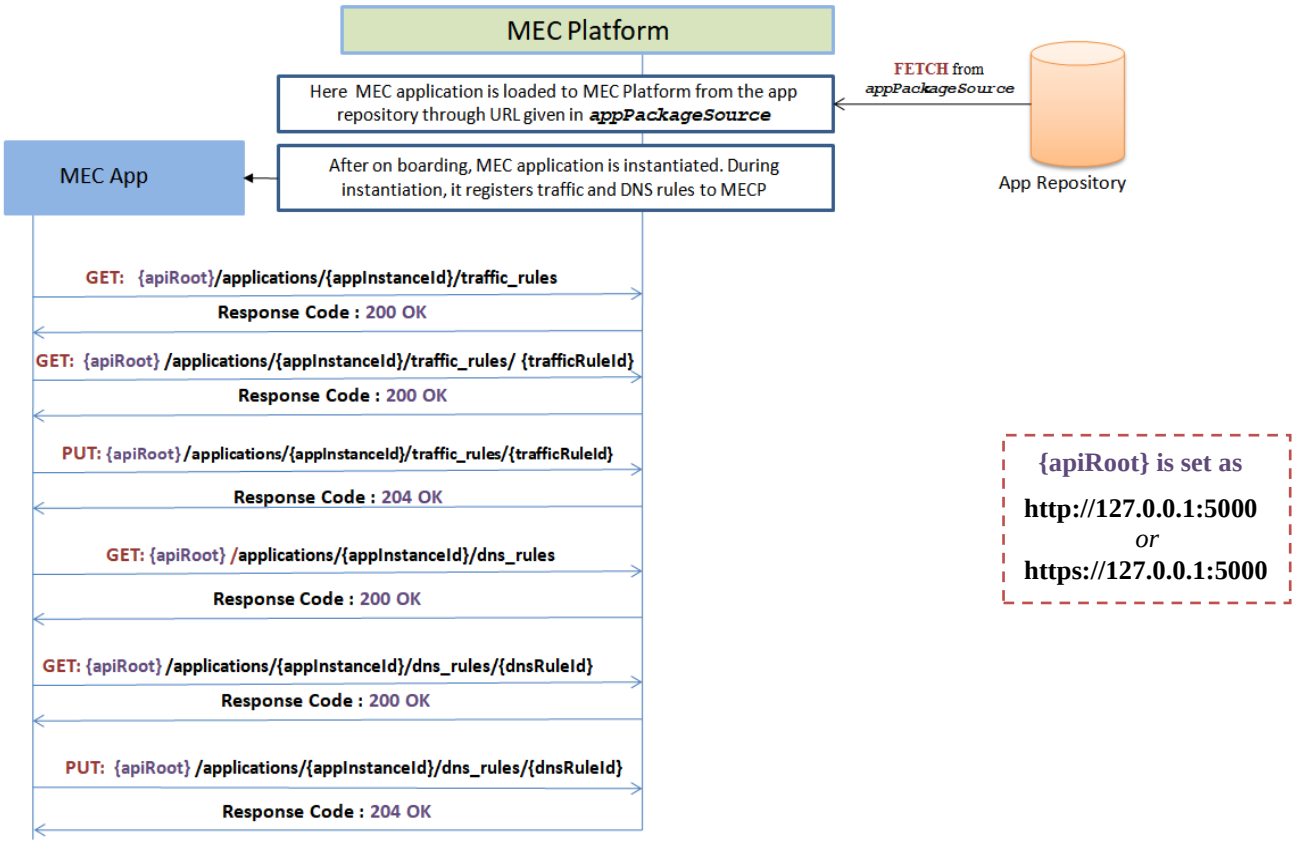


Figure 3: MEC App on-boarding and Instantiation

3.2.1 Fetching the application

Prior to installing the application package on the MEC system, the App package needs to be available in any App repository on the internet bearing a URL. MEC HOST ensures to load the application package into MEC platform from **`appPackageSource`** address **(*see* Figure])** which is available in the app context request parameter (as a URL) **(*see* Table 2])**.

3.2.2 Application Instantiation

After application on boarding, the MEC app is ready to run on the MEC Platform. Once the MEC application is installed it registers Traffic and DNS rules **(*see 10*)** to MEC Platform through the APIs **(*see* Figure])** Package on boarding includes checking the integrity and authenticity of the signed packages, validating application traffic rules, DNS rules and basic requirements. MEC applications are packaged by application developers (or in some cases also by MEC operators), with all the necessary dependencies according to the specific MEC platform requirements **(*see* [2]).**

3.2.3 Back-propagating the application info to the Client App

After the registration process, the MEC application is run using the **`referenceURL`** parameter (*that is present in application context information made available to the Client App*). Successful completion of these stages allows an MEC application to send app traffic to the Cloud application (on the basis of event triggering and the rules that have been registered). These traffic and DNS rules can be called along with App Instance ID. App Instance ID and port number is received by the MEC App as a command line argument while running the instance of application. Since, MEC application and Cloud application are expected to be on separate networks so they can communicate directly. Refer to **[6]** for more details.

API Details

a) Interface between Client App and Device App

Global definitions of API in this section are defined as - **{apiRoot}/{apiName}/{apiVersion}/**
{apiRoot} is defined as **–http://127.0.0.1:5000** *or* **https://127.0.0.1:5000.**
{apiName} is defined as – **mi**
{apiVersion} is defined as – **v1**

3.3 GET –/mi/v1/app_list[app list parameters]

GET method is used to resolve app list query information between Client app and the available Device app

- **a.** Resource URI- **{apiRoot}/mi/v1/app_list [app list parameters]**
- **b.** Request parameters - Variables of input app list parameters (*see* Table 2])
- **c.** Response - Available application lists
- **d.** Response Code - 200 OK otherwise 404 if list not found or client provided a URI that cannot be mapped to a valid resource URI. (see
- **e.** Table 1])

3.4 POST –/mi/v1/app_context [app context parameters]

The POST method is used to create a new application context in the area of MEC system**.**

- **a.** Resource URI – **{apiRoot}/mi/v1/app_contexts [app context parameters]**
- **b.** Request parameter - Variables for resource "all UE App Contexts" (*see* Table 2])
- **c.** Response - **contextId**, **referenceURL** and others
- **d.** Response Code - 201 OK, otherwise 404 if not found or if given clientURI not mapped to with valid resource URI (*see*
- **e.** Table 1]).

3.5 PUT- /mi/v1/app_context/{context_Id} [app context parameters]

The PUT method is used to update the callback reference of the existing application context.

- **a.** Resource URI - **{apiRoot}/mi/v1/app_contexts/{contextId}[app context parameters]** (*see* Table 2])
- **b.** Request Parameters - All UE App Context, contextId. Only the attribute callback Reference is allowed to be updated.
- **c.** Response Code – 204. No content otherwise, 404 if not found or if client URI not mapped to a valid resource URI (*see*
- **d.** Table 1]).

3.6 DELETE–/mi/v1/app_Contexts/{Context_Id}

The Delete method is used to delete the app context of the existing application.

- **a.** Resource URI – **{apiRoot}/mi/v1/app_contexts/{contextId}**
- **b.** Request Parameters- contextId (*see* Table 2]).
- **c.** Response Code - 204 No Content, otherwise 404 in the case of error or if client URI not mapped to a valid resource URI (*see*
- **d.** Table 1]).

b) Interface between MEC app and MEC platform

The communication between MEC app and MEC platform is based on "applications" predefined interface. There are many API (end points) related to Traffic rules and DNS rules for more details (*see* [3], *see* [6]).

Table 1: API Summary

Sr.	Name Of End Point	METHOD	RESPONSE CODE	INPUT	OUTPUT
1	**mi/v1/app_list**	GET	200 if SUCCESS otherwise 404	App list query	Available Lists
2	**mi/v1/app_contexts**	POST	201 if SUCCESS Otherwise 404	UE app Context request	contextID and others
3	**mi/v1/app_contexts/{contextId}**	PUT	204 If SUCCESS Otherwise 404	contextId	Response Code 204 No Content
4	**mi/v1/app_contexts/{contextId}**	DELETE	204 If SUCCESS Otherwise 404	contextId	Response Code 204 NO Content

Table 2: JSON Message [5]

Type of Message	Input Parameters	Output Massage JSON
app_list	**appName, appProvider, appSoftVersion, appDescription, vendorSpecificExt, vendorId**	{ "appList": [{ "appInfo": { "appDId": "string", "appName": "string", "appProvider": "string", "appSoftVersion": "string", "appDVersion": "string", "appDescription": "string", "appCharcs": { "memory": 0, "storage": 0, "latency": 0, "bandwidth": 0, } } }], "vendorSpecificExt": { "vendorId": "string" } }

app_contexts	**contextId: "NONE",**
	associateUeAppId,
	callbackReference,
	appInfo,
	appDId,
	appName,
	appProvider,
	appSoftVersion,
	appDVersion,
	appDescription,
	appPackageSource
		{
		"contextId": "string",
		"appInfo": {
		"appDId": "string",
		"appName": "string",
		"appProvider": "string",
		"appSoftVersion": "string",
		"appDVersion": "string",
		"appDescription": "string",
		"referenceURL": "string",
		"appPackageSource": "string"
		}
		}

5 References

[1] MEC in 5G networks, ETSI White Paper 28.

[2] Developing Software for Multi-Access Edge Computing, ETSI White Paper 20.

[3] Mobile Edge Computing (MEC); Mobile Edge Platform Application Enablement, *ETSI GS MEC 011. V1.1.1*, 2017.

[4] Multi-access Edge Computing; Framework and Reference Architecture, *ETSI GS MEC 003, V2.1.1*, 2019.

[5] Multi-access Edge Computing; UE Application Interface, *ETSI GS MEC 016, V2.1.1*, 2019.

[6] https://forge.etsi.org/swagger/editor/?url=https://forge.etsi.org/gitlab/mec/gs011-app-enablement-api/raw/master/MecAppSupportApi.yaml [API Details]