ETSI White Paper No. 55

MEC support towards Edge Native Design

1st edition – June 2023

Authors:

Dario Sabella, Alice Li, Hyeonsoo Lee, Luca Cominardi, Qian Huang, Emmanouil Pateromichelakis, Vivek Kashyap, Cristina Costa, Fabrizio Granelli, Walter Featherstone, Faraz Naim, Xu Yang, Hanyu Ding, Sundar Nadathur, Lijuan Chen, Dan Druta, Qi Tang, Bob Gazda, Gao Chen

ETSI
06921 Sophia Antipolis CEDEX, France
Tel +33 4 92 94 42 00
info@etsi.org
www.etsi.org

Authors

Alice Li (Huawei, ETSI ISG MEC vice-chair)	*Hyeonsoo Lee (SK telecom)*
Bob Gazda (InterDigital, ETSI ISG MEC WG DECODE delegate)	*Lijuan Chen (ZTE)*
Cristina E. Costa (National Inter-University Consortium for Telecommunications, CNIT, Italy)	*Luca Cominardi (ZettaScale Technology)*
Dan Druta (AT&T ETSI ISG MEC delegate, W3C co-chair of Web & Networks Interest Group)	*Qian Huang (China Unicom)*
Dario Sabella (Intel, ETSI ISG MEC chair)	*Qi Tang (Google, ETSI ISG MEC delegate)*
Emmanouil (Manos) Pateromichelakis (Motorola Mobility)	*Sundar Nadathur (Intel)*
Fabrizio Granelli (Consorzio Nazionale Interuniversitario per le Telecomunicazioni, Italy)	*Vivek Kashyap (Intel)*
Faraz Naim (Accenture)	*Walter Featherstone (Apple, ISG MEC Vice Chair; ISG MEC WG DECODE Chair)*
Gao Chen (China Unicom)	*Xu Yang (Huawei)*
Hanyu Ding (China Mobile)	

Contents

Executive Summary

Multi-access Edge Computing (MEC) offers application developers and content providers cloud-computing capabilities and an IT service environment at the edge of the network. Edge computing environments enable flexible deployments and the possibility to manage and run edge applications by adopting cloud-based technologies.

The Edge Native approach, as a natural evolution of Cloud Native, refers to *building and running edge applications which take edge environment characteristics in mind and can best utilize the capabilities of the edge environment*. In fact, if on one hand the edge environment is assumed to be a natural extension of the cloud to expand infrastructure to build a cloud-to-edge continuum, on the other hand the edge environment has a few key characteristics that distinguishes it from the traditional cloud environment, which leads to specific requirements for both the edge applications and edge system.

The Edge Native approach takes into account the principles and specific requirements of edge computing and combines them with the modern architectural approach introduced by Cloud Native. So, as a starting point Edge Native applications are architected in a similar way as Cloud Native to be elastic and distributed using microservices architecture and modern methodologies such as agile development, CI/CD and containers. However, the Edge Native approach further extends the application design to also exploit the specific benefits of the edge.

The goal of the present White Paper is to show these unique requirements for edge applications, describe what Edge Native is and how the ETSI MEC and other organizations in the industry can support the Edge Native design by helping the developers to meet the requirements. Also, the paper includes perspectives for further work in ETSI MEC. This White Paper is primarily targeted to edge application developers and the general technical community interested in ETSI MEC solutions or Edge Native application design concepts.

1 Introduction

Multi-access Edge Computing (MEC) is a promising technology that brings services closer to the end user, thus providing lower network latency, better data privacy and other benefits that are critical for business scenarios like V2X, AR/VR, Industry 4.0, etc.

MEC enables a flexible environment for deploying and managing edge applications by adopting cloud-based technologies, e.g., virtualization, service-based management, heterogeneous hardware management, etc. However, the edge environment has a few key characteristics that distinguishes it from the traditional cloud environment, which leads to specific requirements for both the edge applications and edge system. For example, the edge environment usually has limited resources and is geo-spatially distributed, and the users/clients are likely to move while consuming the edge services. These characteristics lead to the birth of the "Edge Native" concept. "Edge Native", which is inspired by "Cloud Native", refers to *building and running edge applications which take the above edge environment characteristics in mind and can best utilize the capabilities of the edge environment*.

Since its foundation in 2014, the ETSI Industry Specification Group (ISG) MEC has been focusing on the definition and standardization of the MEC system considering the unique challenges of the edge environment. Besides ETSI, other organizations like Eclipse [1], 5GDNA [2], etc. are also studying how to enable the "Edge Native" design and provide tools and guidelines for the application developers. This White Paper tries to summarize the existing works in the industry and their support for the "Edge Native" design.

The White Paper is organized as follows: while clause 2 explains in detail the concept of "Edge Native", clause 3 describes what "Edge Native" requires to the edge applications and MEC system; clauses 4 and 5 introduce how ETSI ISG MEC and other organizations support the "Edge Native" concept, and finally clause 6 concludes the White Paper with insights on future evolutions.

2 What is Edge Native

To understand the concept of Edge Native, we should first start from Edge Computing, that has extended the traditional client-server architecture by introducing an intermediary application component (at the edge). So, while edge computing is not a new concept anymore, from an application design perspective, Edge Native applications extend this concept because this intermediary application component is designed with its own deployment characteristics. To deploy an Edge Native Application is much like building a new application rather than just repackaging a cloud application. Porting a client server application to the edge involves identifying key requirements that can be translated into functional specifications tailored to the edge flexibility. It may also involve the extensive use of serverless microservices, event driven Pub/Sub design patterns and synchronized database technologies.

From a deployment perspective the edge component is strictly tied to the network and data center topology. It is also both an architectural and a deployment paradigm shift.

So, in a nutshell, Edge Native applications are *natively edge aware*: applications are designed explicitly with edge functionality that takes direct advantage of the proximity to the user but also includes additional security precautions for data storage.

When looking at Edge Native Computing it is very difficult to address all the characteristics into a single taxonomy. A holistic view needs to consider a three-dimensional perspective:

1. Functional Requirements
    - Latency sensitive applications (AR/VR, CDN)
    - Compute intense workload (AR/VR)
    - Integration with Network APIs (e.g., Location)
2. Control and management
    - Edge Applications as a Service
    - Edge Platform as a Service
3. Network Location and placement
    - On customer premise (Enterprise vCPE, setup box)
    - Access Network (RAN)
    - Core Network
    - Distributed Cloud

Applications and network functions can manifest permutations of all three characteristics and therefore the architecture supporting them needs to be flexible enough to accommodate a wide range of options.

2.1 Definition and characteristics of Edge Native applications

Having a look to the definitions in literature, the "Edge Native" concept is used to describe *applications that are built natively to leverage edge computing capabilities, leverage cloud-native principles while considering the unique characteristics of the edge areas such as resource constraints, security, latency, and autonomy* [3]. A similar concept is also described by recent publications (see Eclipse Foundation, Edge Native WG [1]).

An Edge-Native application is built natively to leverage edge computing capabilities, which would be impractical or undesirable to operate in a centralized data center. Edge-native applications leverage cloud`-native principles while taking into account the unique characteristics of the edge in areas such as resource constraints, security, latency, and autonomy. Edge native applications are developed in ways that leverage the cloud and work in concert with upstream resources. Edge applications that don't comprehend centralized cloud compute resources, remote management and orchestration or leverage CI/CD aren't truly "edge native", rather they more closely resemble traditional on-premises applications. A traditional SCADA application within a nuclear power plant that has no connection to the cloud would not be considered an Edge-Native Application; even though the SCADA application may use edge capabilities to provide data ingest, data reduction, real-time decision support, or to solve data sovereignty issues).

To fully substantiate the advantage of moving towards edge, just using the technology isn't enough, one needs to develop and test Edge Native applications expeditiously. The key is to understand the unique characteristics of the edge environment and accordingly design the applications as well as the platform and management system to best suit them.

The following list summarizes some of these *characteristics* and highlights their impacts:

- **Low Latency**:

  With the advancement in the digital world, be it IoT, Machine Learning, Computing, Virtual Reality or Augmented Reality, the need for real-time interaction and data processing is extremely crucial. This need for a faster response time, or in other words low latency, is the primary driving force of edge computing. Cloud Native, which is ideal for centralized, large-scale data center deployment, encounters problems pertaining to resource allocation, low latency communication, as well as the customization, and O&M of edge nodes, which are core characteristics of Edge Native.

- **Enhanced Security:**

  Another driving force for deploying Edge Native applications is to keep data processing local to edge devices, and only send information to the cloud or a central database when required. This substantially decreases the amount of sensitive information that is shared across the network at any one time, compared to a cloud-native application that is constantly transmitting data back and forth. The high concern of the local data security also requires the support of better isolation, data encryption and identity authentication methods.

- **Resource Constraints:**

  Unlike cloud-native applications which typically run in data centers where resources can be considered as unlimited, Edge Native applications could be deployed in a single rack or server which located in an edge office or in a workshop of a factory. This limit on physical space results in limited hardware resources for the edge infrastructure. This makes it very difficult to conduct compute massive tasks, and could lead to low accuracy of the data, weaker defense of abnormal and malicious operations, and constraints on the potential workloads.

- **Mobility/Portability:**

  Based on application provider's preference, edge applications may be deployed or migrated to different MEC platforms in a pre-configured manner or dynamically on demand. For example, in the V2X scenario, edge applications need to be relocated to minimize the end-to-end latency for V2X service operation. This requires both application and MEC APIs to support dynamic provisioning, be simple for flexible consumption; as a consequence, this scenario requires the application developer to be aware of the network conditions and capabilities to ensure the KPIs for the application service are met.

- **Resiliency**:

  Traditional edge devices usually lack resiliency support and could spell catastrophe for the entire system without the right tools to quickly remedy the issue. Edge-native applications should overcome this shortage and support mechanisms by other methods, such as, automatically reroute processing to other edge applications or sites. This may add another dimension of distributed resources resiliency when compared to a typical cloud-based approach.

- **Heterogeneous Infrastructure**:

  With the many foreseeable scenarios and challenges, it's clear there is no silver bullet for edge deployment and operation (see ETSI White Papers n.30 [4] n.24 [5]). Different options for data collection, compute, storage, communication, and orchestration are available along a continuum from the edge to the cloud. This makes the edge a heterogeneous mix of resources that can be extremely diverse in terms of capabilities and operating conditions, which distinguishes Edge Native from Cloud Native.

- **Network Awareness:**

  Edge Native platform may support both abstracted network services and application services. The former, especially when we consider services exposing complex mobile network parameters, e.g., Radio Network Information Service (RNIS), V2X Information Services (VIS) allow the Edge Native applications to have the knowledge of the network logic and thus enable flexible portability and configuration of the applications.

The main Edge Native characteristics, from a developer and application design point of view, are summarized and explained in the following table:

Table 1: Paradigm shift from Cloud native to Edge Native app design

Cloud Native Applications	Edge Native Applications	Impacts on App design
More extensive usage of (centralized) remote cloud servers	Keep more data processing local to edge devices	Edge Native Apps only send information to the cloud or a central database when required, while Cloud Native Apps may constantly transmit data back and forth
May adapt passively to network degradation	Located closer to user data, access network, and can gather/monitor network and context information	Edge Native Apps may actively utilize network conditions, location, distance/latency to optimally adapt application behavior to meet the KPIs and influence endpoint behavior
Assume optimal availability of network, storage, and compute resources	May operate with intermittent connectivity to the remote cloud, and with limited resources when deployed on the field	Edge Native Apps may use relatively limited compute, storage, network resources. May have to backup to cloud or resort to distributed applications or mobility
Applications deployed in certain area/region or data centers	Edge-native development platform allows for the dynamic deployment and movement of resources across the system	Edge Native environments are more likely to be scattered than cloud native environments, and applications may consider tracking the mobility of users and provide services accordingly
High availability and replication are assumed to be provided by the supporting platform infrastructure	Automatically reroute processing to other edge devices on the network in case of an outage via local rules designed to increase reliability and avoid data loss	Edge Native environments may be more vulnerable to disasters, and may need to rely on cross-site resiliency mechanism to ensure the service continuity
User and sensitive data is located in the cloud, and it needs eventually to be transferred from client to remote server.	Local access to user and sensitive data. Eventual data transfer is done from client to edge server.	Edge Native Apps keep data processing local to edge, with benefits in terms of security, e.g., by solving data sovereignty issues. Information is sent to the cloud or a central database only when required.
Assume homogenous and abstracted infrastructure resources	Deployed on heterogeneous and (in some cases) specific hardware configurations	Edge applications may not always have device and hardware abstractions. Therefore, where possible, Edge Native Apps should be flexible and portable to run on heterogeneous hardware

3 Use Cases and Requirements for Edge Native

Here we list some use cases that demonstrate Edge Native characteristics and capabilities described in the previous section. The reader may notice that these are not "edge native use cases" but they can benefit from edge native applications. Furthermore, it is worth noting that the use cases are not the only ones that may benefit from edge native applications, but they are relevant examples in this perspective.

3.1 Edge Enterprise use cases

Description: A private 5G wireless LAN connecting enterprise or site-specific edge devices can support high network load and real-time communication. The diagrams below illustrate a network deployment where an Edge Native platform hosts functionalities for edge orchestration and cross-edge collaboration, as well as enterprise service applications. The Edge Native applications may control workloads on the factory floor (such as a robotic arm), run analytics on camera feeds, apply security policies for access to public cloud or to enterprise private cloud, onboard and validate BYOD devices of end-users, etc. All these deployments will be subject to design requirements as described in the Table 1.

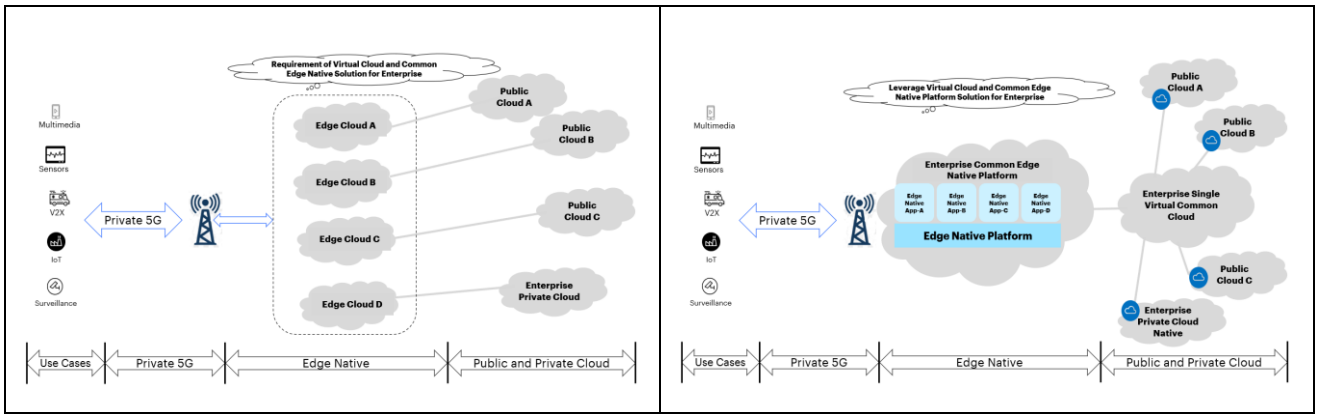

Figure 1: Edge Enterprise scenarios: (left) traditional multi-Cloud implementation; (right) leveraging Common Edge Native Platform and virtual common Cloud.

Requirements: Centralized management with cross edge collaboration optimizes edge resources and creates a common pool of resources for the enterprise. Well defined interfaces are essential to overcome the challenges of integrating with multi-provider access to public cloud backends. Strong data control and security framework are crucial to enable the Enterprise to protect against data breaches on premise and in transit to the public cloud.

3.2 Vehicle-to-Everything (V2X) Cooperative Perception Scenario

Description: In a cooperative perception scenario, several vehicles exchange data related to their trajectories. Data is gathered at the network edge and is used to build a view of the surrounding environment. Artificial Intelligence powered analytics on this data may be used to predict potential collisions and dangers. This use case relies on 5G and AI, which allow to make the road safe and the vehicular traffic fluid (e.g., in the roundabout situation, where fluidity and safety are of paramount importance}.

The European Commission has set the goal for a safer, more fluid, and thus less polluting traffic to be reached by 2030[1].

Vehicles' cooperative perception is a challenging task because it deals with numerous vehicles that have to detect in real-time the surrounding traffic scenario, exchange their sensed data, and share their intended maneuvers with other vehicles with minimum latency. AI agents must cope with mutating radio network performance, should then dynamically learn changing network states, and take reconfiguration action.

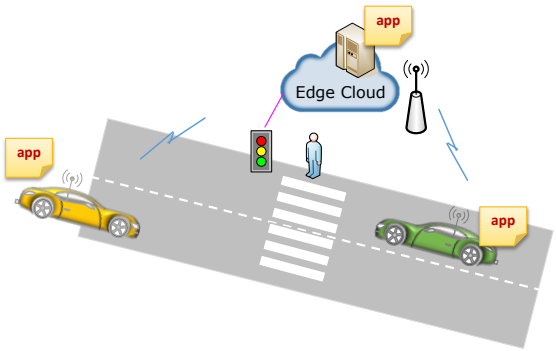

Figure 2: Edge Native V2X applications in Cooperative Perception scenarios

Requirements: Figure 2 shows a typical V2X scenario for Cooperative Perception. There are many challenges to be solved by edge native apps in this scenario (see T-220019 [6] and SAE_J3224 [7]) including real time data collection from multiple sources. The V2X Edge Native applications will therefore handle high volume of data from the vehicular UEs, and support AI agents for analysis and prediction. Where needed the Edge Native applications offload non-time critical operations to other edge nodes or to the cloud. It is important to take appropriate decisions on offloading computation to take advantage of the geo-spatial distribution of the edges. Additionally, it is supposed to leverage on a security framework offered by infrastructure (in fact, in these heterogeneous V2X environments it is assumed the presence of multiple operators and service providers, where security for the operations is an essential aspect).

3.3 Media services over edge environment

Description: This use case describes a media service scenario using edge environments. In the media market, there is a growing variety of devices such as TVs, smartphones, and connected cars that enable media consumption. There is also an increasing demand for high-quality, large-scale data processing and ongoing demand for two-way media services. To support these trends, next-generation broadcasting systems can build edge-based broadcast infrastructure systems while providing new combined service scenarios with existing communication networks based on the IP-based transmission standard called ATSC (Advanced Television Systems Committee), currently at Release 3.0[2].

For example, in an environment that supports dual connectivity, media broadcasting can be carried out using edge-based broadcast networks, while user-specific advertising and two-way interactive services can be facilitated through 5G networks. Furthermore, by providing offloading scenarios between 5G and ATSC 3.0 networks based on user location, it is possible to reduce 5G-network congestion while ensuring a high-quality media consumption experience for users.

[1] https://transport.ec.europa.eu/transport-themes/mobility-strategy_en

[2] https://www.atsc.org/atsc-documents/type/3-0-standards/

Requirements: To provide media services, the Edge Native approach needs to consider the following aspects. Firstly, it should be capable of providing an optimal edge environment, including CPU, GPU, storage, etc., tailored to the needs of operators and content providers. Secondly, in order to offer converged services across broadcasting and broadband cores, a secure mechanism for exchanging user information and network state between heterogeneous systems should be defined. Lastly it can be necessary to establish an integrated system that accurately metes the transmission volume of each content and generates billing information based on it.

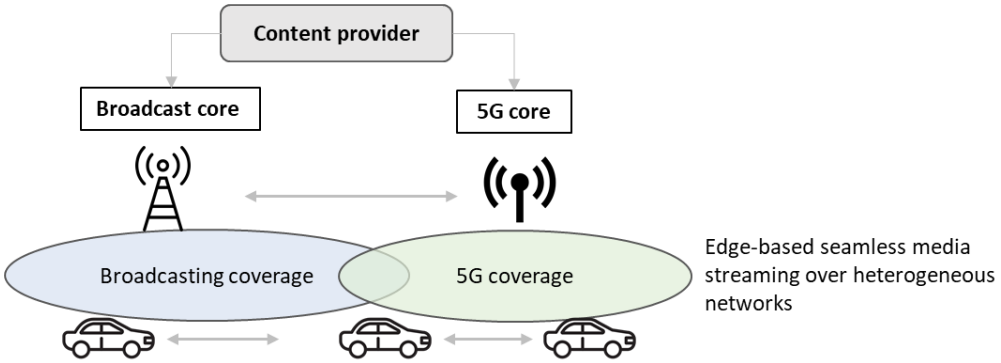

Figure 3: Edge-based media service

3.4 Drones and high-altitude platform integration with 5G for service delivery

Description: This use case describes an deployment of a flexible and automated MEC solution for deploying connectivity and services using a swarm of drones. Mobile Base Stations work in cooperation with a high-altitude platform (HAP) (e.g., a balloon or ultra-light vehicles). The drones (often referred also as UAVs – Uncrewed Aerial Vehicles) build the radio access network (RAN) and the edge, providing robust and flexible communications, thus representing the programmable network infrastructure used to connect users and deploy services in an on-demand framework.

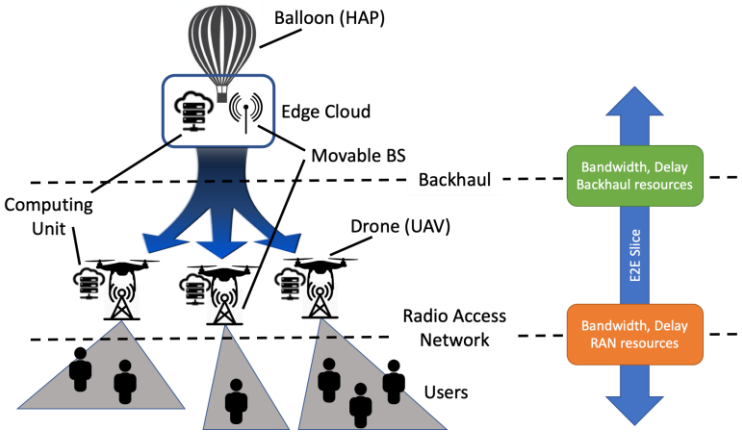

Figure 4: Pictorial view of the UAVs/HAPs use case.

Requirements: A "MEC application enablement framework" has been defined by ETSI to support the network exposing information towards authorized third-party applications. ETSI has delivered seven Group Specifications that define different REST MEC service APIs addressing various aspects that range from mobile edge service APIs to application lifecycle management. The information is made accessible to Edge Applications developers through the Mp1 interface, that allows accessing to data either provided by the MEC platform itself, such as Radio Network Information (RNI API) or Location Information (Location API), and those provided as a service by other MEC applications. Figure 5 depicts the exemplary architectural integration between HAP/UAV and MEC system. In this scenario, MEC services are associated to a network slice of the 3GPP 5G architecture with specific Quality of Service (QoS) requirements (additional details about this scenario are available in [30]).

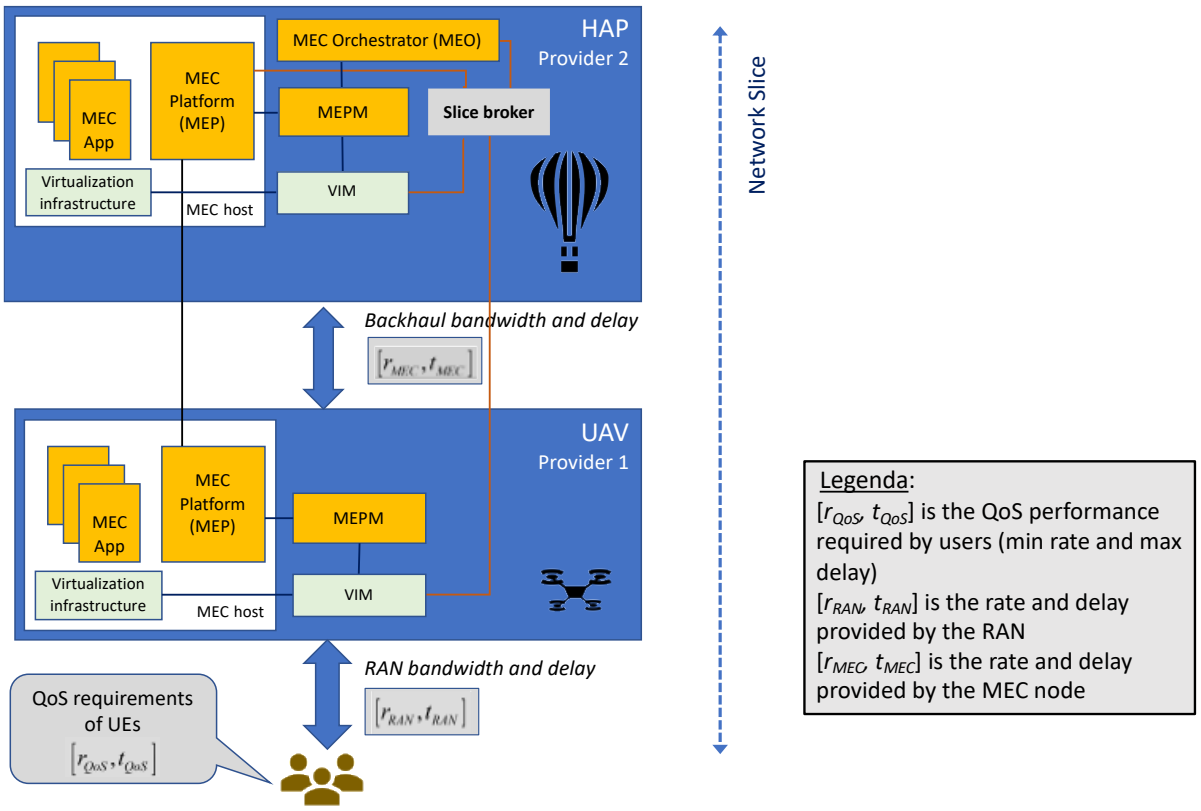

Figure 5: Exemplary architectural integration with MEC
(Note: Slice Broker is a hypothetical additional element in this scenario, i.e., not belonging to the ETSI MEC architecture)

This multi-tier highly dynamic architecture makes a very interesting case for Edge Native applications, including highly distributed ones. For example, control and mission planning applications can themselves be considered as Edge Applications, taking advantage of the virtualization environment for reconfiguring the UAV mission and services provided. Designing these applications as Edge Native allows to address their requirements as follows.

Firstly, the mobility of both UEs and Mobile Base Stations (i.e., the UAVs hosting the MEC host) may change the strategy adopted by the applications running on the UAVs as well as mission planning. We can assume also that the HAP can move, although on a different scale (may need e.g., to adjust altitude to weather conditions, such as winds). Secondly, geo-spatial data are an important input for control apps and mission planning, as well as monitoring and mapping applications (e.g., IoT). Thirdly, both USVs and HAPs have limited resources, not only in terms of computational and storage capabilities but also in terms of battery life (that can be partially mitigated with the use of renewables) and other UAVs specific resources. Orchestration, when crossing resources, performance, and mission requirements, can get very sophisticated. Fourthly, autonomy and resilience must always be taken into consideration since UAVs may lose connection with HAP, and HAP itself may lose connection with central office, or the connection is not optimal (must decide what to do, e.g., move the UAV or move the app). Fifth, especially in this multitier architecture, portability of application may have a certain relevance not only because UAVs may have different hardware capabilities (including services and sensors, and a different mix of resources) but also because they may be required to run on the HAP as well, thus they are required to cope with different coverage, different latency, different network conditions. Last, but not least, in some cases low latency should be considered, depending on the specific application scenario and mission. In this perspective, leveraging MEC service APIs available at the edge can be a critical asset for designing these applications as Edge Native.

3.5 Real-Time task offload use cases

Description: The concept of real-time task offload to the nearby MEC infrastructure brings many advantages. It reduces latency, ensures faster response times, and provides a seamless user experience. This use case (in Figure 6) considers an Edge Native App service where a user requires real-time language translation service using speech AI models. Leveraging MEC and Edge Native Apps, the user's device (such as phone, AR/VR, wearable, and IoT devices) can offload the real-time computationally intensive Speech-to-Text, Language Translation and Text-to-Speech tasks to the nearby edge nodes.

Recent research has demonstrated the remarkable capabilities of large AI models that surpass the limitations of small models (see OpenAI, “GPT-4 Technical Report” [33], “Google USM: Scaling Automatic Speech Recognition Beyond 100 Languages.” [34] and “Massively Multilingual Neural Machine Translation.” [35]) and achieve enhanced performance (see “AudioLM: a Language Modeling Approach to Audio Generation” [36]). However, it requires significant memory and computational resources. For general consumer-app developers, it becomes if not infeasible but challenging and costly to deploy the models on devices (especially wearables and IoT devices). Meanwhile, the substantial data (such as raw audio or video data) transfer involved in network communication presents considerable challenges to both network capacity and the computing power of cloud computing infrastructures. In this context, Edge Native App emerges as a solution with the potential to effectively address these challenges.

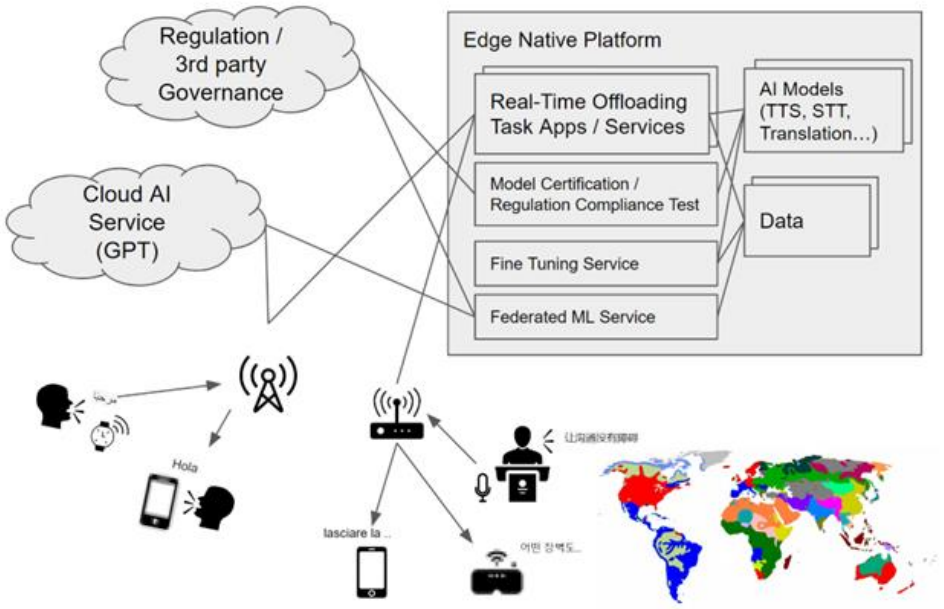

Figure 6: Edge Native App example for a real-time AI language translation task offload scenario

To provide users with safe and consistently high-quality services across different platforms [16], it is foreseeable AI models especially generative models will require increased regulations and governance in the future (see "Generative Language Models and Automated Influence Operations: Emerging Threats and Potential Mitigations." [33] and “ETSI Activities in the field of Artificial Intelligence Preparing the implementation of the European AI Act” [34]. Protection of biometric information, such as voice characters, and privacy-related data becomes crucial. Development of regulations specifically tailored to these AI models is necessary to mitigate risks and ensure ethical and responsible use. It is worth to note that Edge Native Apps are particularly suited for language-related tasks due to the natural geographic diversity of human languages and dialects. Customization and optimization of language AI models using local data can enhance the user experience. By leveraging Edge Native Apps, consumers benefit from consistent quality of service (QoS) across different applications.

The above description is tailored to the case of real-time translation AI model scenarios, but it can be extended to many other real-time task offloading use cases which are key for MEC (for further background, see the ETSI GS MEC 002 [10] specification listing all use cases and technical requirements [10].

Requirements: The ETSI MEC systems offer several key features. Firstly, ETSI MEC establishes standardized system requirements for the deployment of language AI models, which support low latency, robustness, high availability, flexibility, and efficiency. Secondly, ETSI MEC ensures that clear and well-defined API interfaces are available for the usage, maintenance, and governance of AI models, aiming for cost-effectiveness, flexibility, and scalability. Furthermore, given the increasing usage of data and AI in the future, more work will be required from regulation side, and ETSI MEC standards will need to further leverage and comply with those regulations. ETSI MEC systems support necessary regulation compliance (see White Paper 46 [16], emphasizing an open, transparent, neutral, and collaborative platform to AI that prioritizes safety and broad benefits. These systems maintain a neutral standpoint and impose restrictions on data collection to safeguard user privacy (see ETSI White Papers n. 49 [15] and n. 46 [16]).

4 ETSI MEC Support on Edge Native

In this clause we provide a description of some key enablers currently supported by the ETSI MEC standard. The work toward support of edge native application is not finished, of course, but at least current specifications offer a good starting point for the application development in that perspective.

4.1 Application Mobility Service

Application mobility is a unique feature of the MEC system, which supports relocation of user context and/or application instance from one MEC host to another, or between a MEC host and a Cloud, especially when the MEC host is attached to the mobile operator's networks. Relocation decisions may be based on device mobility, customer profiles, application preferences and/or MEC infrastructure capability.

ETSI GS MEC 021 [9] defines the AMS (Application Mobility Service) API with the data model and data format. Such service is expected to communicate via web interfaces (such as MEC's RESTful API approach) and perform a specific business function. The application mobility service will provide the services to the registered consumers:

- Endpoint information of adjacent application instances with communication links.
- Identification of application instance running on the target MEC host.
- Communication link information between the source and target instances of the same application.
- Notification of application mobility status.
- Assistance to clean up the user information at the source application instance and MEC platform when the user context has been transferred to the target application instance.

In combination with the set of services made available by the AMS API, DevOps practices are considered as being highly complementary. In fact, if from one hand, those systems can be easily expanded by adding new micro services without unnecessarily affecting other parts of the application, thereby offering flexibility and achieving scalability, on the other hand the MEC Orchestrator (MEO) can select proper application instance to provide service for the device that moved to the new location Application mobility may involve multiple functional entities in MEC system to relocate application instances and transfer user and application specific information, depending on different implementation approaches. The determination of the need for synchronization as well as the synchronization of the user context are application implementation dependent, this facilitates service continuity and offers programmers the opportunity to design their applications with the capability to leverage and optimize the user context relocation procedure mechanisms to determine the right relocation timing and data synchronization. The actual user context transfer procedures are dependent on the specific MEC app deployment and the underlying support of the MEC host (e.g. operating system, virtualization layer, but also interworking with the 5G core network), and this aspect is currently outside the scope of MEC specifications. Therefore, it is recommended to collaborate with 3GPP, for identifying which 3GPP procedures may require extension when MEC is deployed in 5G networks.

As for the uncertainty of motion, application mobility may involve multiple MEC systems. The scenario of application mobility between two MEC systems is currently not specified in ETSI GS MEC 040. Instead, the ongoing GS MEC 021 v3.1.1 version will specify and update to support new requirement for the inter MEC systems mobility. Procedure between E/WBI defined in MEC 040 e.g., MEC application instance discovery, could be reused.

4.2 Edge Service Discovery and Advertisement

MEC enables the implementation of MEC applications as software-only entities that run on top of a virtualized infrastructure, which is located in (or close to) the network edge. The MEC platform, with the collection of essential functionalities offers an environment to run MEC applications, including a set of MEC service APIs (see Annex B), and also enable MEC applications to *provide* and *consume* MEC services (see Figure 7 and also ETSI GS MEC 011 for further context [12]).

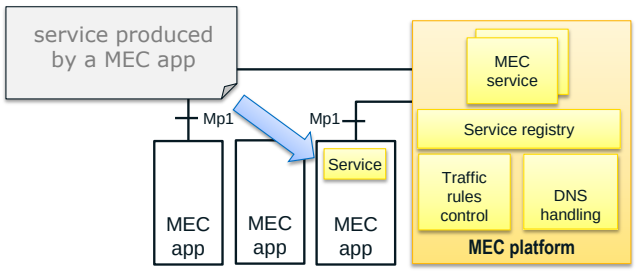

Figure 7: A MEC application instance may consume and/ or produce MEC services

Edge service discovery and advertisement is part of basic functions enabled via Mp1 reference point between the MEC platform and the MEC applications, as defined in ETSI GS MEC 011 [12]. When a MEC service is registered by the service producing MEC application, the authorized relevant applications will be notified about the newly available service. Moreover, the authorized relevant applications will also be notified about the service availability changes of that service. As part of service registration, the relevant information about the service is provided to the platform, and the service is bound to a transport that is either provided by the MEC platform, or by the application itself. It is also possible for a MEC application instance to query the availability of a MEC service or a list of MEC services.

Please note that there are other basic functions defined in ETSI GS MEC 011 [12], including:

- MEC service management:
  - authentication and authorization of producing and consuming MEC services,
  - a means for service producing MEC applications to register/deregister towards the MEC platform the MEC services they provide, and to update the MEC platform about changes of the MEC service availability,
  - a means to notify the changes of the MEC service availability to the relevant MEC application,
  - discovery of available MEC services.

- MEC application support:
  - MEC application start-up procedure,
  - MEC application graceful termination/stop,
  - MEC application registration.

In particular, MEC App registration is a ETSI GS MEC 011 [12] procedure available from version v3.1.1 onward. The MEC App registration (which is optional, in the sense that it is up to the app developer to decide if registration is necessary) is intended to support different types of MEC App: 1) a MEC App instance instantiated by MEC management, 2) a MEC App instance not instantiated by MEC management (in both sub-cases where there is or there isn't an EAS profile available[3]). The general purpose of MEC App registration is to ensure that the MEC application instance is discoverable for an instance that is not instantiated by the MEC Management (e.g., for federation purposes). Therefore, the support of MEC App registration is very important especially in MEC Federation environments (e.g., typically encountered in the V2X scenarios described above).

Furthermore, MEC App registration is key for application developer perspective, also because it essentially allows any new MEC services (e.g., produced by a service producing MEC App 1, in Figure 8) to be discoverable and consumable at application level (e.g., a further MEC App 2 consuming at higher level those services). Figure 8 below depicts this important enabler from edge native applications. More background is explained in Annex B, which also provides a further explanation of technical context and standardization landscape.

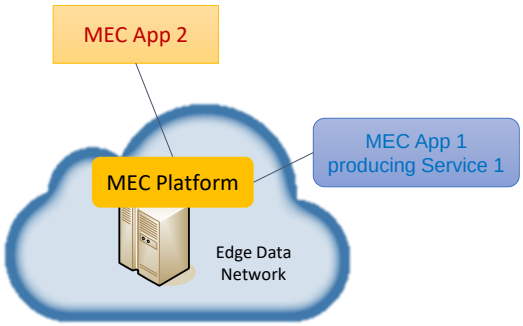

Figure 8: A MEC application 2 consuming a MEC service produced by MEC App 1

4.3 MEC Application lifecycle management

ETSI GS MEC 010-2 [13] defines lifecycle management (LCM) of MEC application, and also describes application rules and requirements. MEC application package defined by ETSI MEC has bundle of constituent files provided by the application provider including package metadata (application descriptor, manifest) and artifacts (software image(s), e.g., VM image or container images, and optionally other files), or URIs to artifacts. MEC application descriptor (template) in the application package includes application

[3] For more details, please read ETSI GS MEC 011. Further background on the alignment with 3GPP can be found here.

rules (such as traffic rules, DNS rules) and requirements (such as resource requirements and services dependency).

Application provider can set values for the parameters in the application template to create a specific MEC application package, which can be on-boarded to MEC system. Then, MEC application can be instantiated on the MEC system- based on package, e.g., MEC applications can have a certain number of rules and requirements associated to them, such as required resources, maximum latency, required or useful services, etc. These edge native requirements are validated by the MEC system management. Other MEC app LCM operations defined in ETSI GS MEC 010-2 [13] are query application instance information, change application instance state, terminate application instance, query application lifecycle operation status, subscribe to notifications relating to app LCM, etc.

4.4 MEC Security

The ISG MEC published a comprehensive set of specifications for security. ETSI ISG MEC standardizes a variety of MEC services by specifying implementation agnostic, RESTful APIs using HTTP.

In ETSI GS MEC 009 [14] the general principles, patterns, and common aspects of MEC Service APIs are specified. More in detail, this specification defines design principles for RESTful MEC service APIs, provides guidelines and templates for the documentation of these, and defines patterns of how MEC service APIs use RESTful principles. The general principles defined in GS MEC 009 apply for all the APIs using Mp1 reference point between MEC Applications and MEC platform (thus applicable also to service producing MEC Applications exposing their services via the MEC platform. The MEC platform also follows the above design principles defined by GS MEC 009, to allow applications to securely interact with the MEC system.

The ETSI GS MEC 011 specification (ref. [12]) focuses on the functionalities securely enabled via the Mp1 reference point between MEC applications. MEC service APIs can expose useful information securely for Application Developers (for more details see the Application Enablement API in ETSI GS MEC 011 [12]).

There are many challenges related to security that need to be considered in future standardization work: Infrastructure security and protection from physical to virtual and application levels, Data protection and User security which includes data encryption - at rest, in transit and in motion (see ETSI White Papers n. 49 [15] and n. 46 [16]).

In particular, ETSI specifications have studied many security issues in virtualization environment, as stated in ETSI GS NFV-SEC 001 v1.1.1 [17] and ETSI GS NFV-SEC 009 v1.1.1 [18], which provide guidelines for multi-layer administrations, secure crash along with performance isolation, etc. These guidelines could also be good references for the edge environment, because ETSI MEC architecture shares some commonality with the ETSI NFV one, and also introduced an architectural variant called MEC in NFV as defined in ETSI GS MEC 003 [8] and ETSI White Paper n. 46 [16].

5 MEC in the Edge Native landscape

5.1 Eclipse Foundation

The Eclipse Foundation founded in 2020 the Edge Native Working Group [1] to deliver production-ready open-source platforms for the development, operation, and management of Edge Native applications. For the Edge Native WG, the edge is anywhere and everywhere outside of traditional IT environments. The Edge Native approach is a natural corollary to Cloud Native because the edge native assumes that the edge is a natural extension of the cloud to expand infrastructure to build a cloud-to-edge continuum. There are some fundamental differences between the cloud and the edge, specifically the architecture, principles, and primitives. An edge compute platform takes advantage of core edge attributes such as location, network topology and latency, and disparate hardware. Edge native systems are aware of these attributes and incorporate them into their core operations. An Edge Native approach considers the primitives and principles of edge computing outlined above and combines it with the modern architectural approach that has come to define Cloud Native [32]. Namely, that Edge Native applications are architected in an equivalent way as Cloud Native to be elastic and distributed using microservices-based application architecture and modern methodologies such as agile development, CI/CD and containers. Like the Cloud Native approach, Edge Native applications are architected such that they are loosely coupled and not hard coded to any one type of infrastructure, that way applications can be deployed and optimized for edge attributes as opposed to any immutable edge.

5.2 Edge Multi-Cluster Orchestrator (EMCO)

The Edge Multi-Cluster Orchestrator (EMCO) is an open-source project under Linux Foundation (LF)[4]. It is aimed at addressing the various challenges posed by edge computing. It originated with the ONAP4K8s module in ONAP but has been a separate project since 2019; in October 2021, it came under the LF Networking group. EMCO is designed for lifecycle management of cloud-native geo-distributed applications and network functions (NFs) at scale across Kubernetes clusters in telco and colocation edges (near/far edges, regional/local locations), public and private clouds, as well as on-premises data centers. Edge/MEC clusters are characterized by some distinctive features: much larger scale than cloud-based clusters, geographical dispersal, much variation in configuration and resource availability, and evolving needs and requirements that arise as edge computing takes off. EMCO addresses by delivering several key attributes for the orchestration of edge-native applications: geo-distribution, scaling, granular placement among clusters based on user-defined criteria, and powerful ways of customizing the deployment to different clusters[5]. Also, EMCO's automation is based on declarative intents: it offers intents for structuring and onboarding applications as Helm charts; intents for defining provider and tenant networks; placement intents based on cluster properties, hardware requirements for specific microservices within the application; and intents for customizing each deployment, including a way to apply patches on the Kubernetes objects defined in Helm charts. EMCO itself is composed of a set of microservices; it can be extended to new use cases by adding new controllers or workflows. For example, one could add a latency-aware placement intent by writing a new placement controller.

[4] EMCO project: https://project-emco.io/

[5] EMCO Edge Relocation WG: https://wiki.lfnetworking.org/display/EMCO/Edge+Relocation

EMCO is a component for MANO in the 5G Super Blueprint initiative[6] in the Linux Foundation, which is currently aiming at defining and demonstrating a stack including ETSI MEC-compliant implementations, including usage of 5G core and for various use cases.

5.3 CNCF: IoT Edge Working Group

The Cloud Native Computing Foundation (CNCF) is another LF project with the mission to make cloud native computing ubiquitous. Since the edge native applications leverage cloud native principles the broad mission is applicable to edge applications the IoT Edge Working Group of CNCF in its white paper[7] analyses the characteristics of cloud native and edge-native applications and offers a set of 'edge native principles.

Edge native applications, in commonality with cloud native apps, are decoupled from the infrastructure enabling portability. Similarly, the platforms have well defined methods for issue detection and metrics, as well as interfaces to manage apps and resources at scale. In both scenarios the apps are implemented and hosted using a variety of popular languages and frameworks.

However, an edge native application will also account for the unique characteristics due to hardware variety, resource limitations, security, latency, and control. Furthermore, edge native applications are developed in ways that leverage the cloud and work in concert with upstream resources. This leads to important differences.

Edge native applications may often utilize distributed and real time data models, work with constrained resources, rely on hardened infrastructure and may have lower resilience than in the cloud. Edge applications may be deployed on a very wide scale (10,000s) and support a large number of external devices (100,000s). The applications are decentralized with a mix of remote and centralized management and zero touch provisioning of hardware and software. Apps need to account for heterogeneity of hardware and connected devices, real time requirements, often assume zero trust environments and account for varied network speeds and connectivity.

The CNCF IoT WP identifies the following core principles for edge applications written to account for the above similarities and differences. The broad five principles are:

1. Resource and device awareness: Edge applications are aware of the hardware variety, external devices and of variable and unreliable network connectivity.
2. Resource usage optimization: Similarly, an edge application has to optimize on resource usage on a continuous basis including application scale, migration, and/or placement based on various conditions.
3. Portability and reuse within limits: Edge applications thought written in platform agnostic and vendor neutral way will require adaptation to local resources and restraints.
4. Span: Edge applications can straddle geographic, latency and failure domains and public or public clouds
5. At scale management: Edge applications need to be centrally observable integrating distributed data collection and closed loop automation. The edge application deployment also requires infrastructure and platform management at scale including unique device, hardware, and virtualization platform orchestration. The application orchestration and corresponding infrastructure linkages need to be managed.

[6] 5G Super Blueprint: https://wiki.lfnetworking.org/display/LN/5G+Super+Blueprint

[7] For further details, the paper itself may be accessed here: https://www.cncf.io/wp-content/uploads/2023/03/CNCF_WhitepaperReport_23.pdf

5.4 CAMARA and Telco Edge Cloud industry initiatives

While ETSI ISG MEC has laid out the specifications for various aspects of MEC computing, such as ETSI GS MEC 011 [12] for Edge Platform Application Enablement, the real-life adoption of these specifications requires both a set of standardized vendor-agnostic APIs as well as an open implementation of those APIs. Several industry groups are involved in defining and publishing these APIs and implementations. This clause documents the main work of some principal industry groups.

The GSMA Operator Platform Group (OPG)[8] is an alliance of over 50 operator groups and 35 non-operators, with a mission to "develop requirements to deliver a common solution to the ecosystem." In fact, they have a subgroup named Operator Platform API subgroup (OPAG) which also proposes high-level APIs based on those requirements to ETSI MEC and other organizations.

One of the key requirements stated by OPG is service continuity in scenarios involving many telco operators. For example, in Vehicle-to-Vehicle and Vehicle-to-Cloud scenarios, a roaming vehicle may need to communicate via different operators with seamless handoff for reliable service. OPG proposes a model of federation of providers, with a consistent Northbound API from each provider to onboard applications, a Southbound API for each provider to expose and configure their backend networks, and an East-West-Bound Interface for partner operators to communicate.

GSMA has also created the Open Gateway project[9] [42], defined as a "framework of common network (APIs) designed to provide universal access to operator networks for developers." With the lofty aim of integrating developers and operators in a unified API economy forming the "world's largest connectivity platform", this project has begun by standardizing several APIs[10] [43], such as consistent number/location verification and billing for subscribers as well as Quality on Demand and Edge Site Selection/Routing for applications.

While standard APIs are indispensable for an interoperable platform, an open-source implementation of those APIs with appropriate backends from different providers is essential for making that vision real. That is the goal of CAMARA[11], an open-source project born in Linux Foundation (LF) as initiative in alignment as well with GSMA OPG, which aims to create "*a federated platform solution for exposing operator network capabilities to external application*". CAMARA operates a set of GitHub repositories [12] where the standardized APIs are developed in open source, including provider backends from leading CoSPs.

This sustained collaboration effort is bearing fruit rapidly. Bridge Alliance[13], a consortium of more than 30 operators spanning Asia, Africa and beyond, has created the Federated Edge Hub[14] , that aims to be "*a truly global collaboration platform for cross market 5G, MEC and web 3.0 applications like the metaverse*". Its test bed, which has been used in real world scenarios, leverages simple discovery APIs from CAMARA, thus demonstrating the power of global standards, collaborative APIs and open source.

[8] GSMA Operator Platform Group, https://www.gsma.com/futurenetworks/operator-platform-hp/

[9] GSMA Open Gateway: https://www.gsma.com/futurenetworks/gsma-open-gateway/

[10] GSMA Open Gateway APIs: https://www.gsma.com/futurenetworks/gsma-open-gateway-api-descriptions/

[11] Linux Foundation project CAMARA: https://camaraproject.org/

[12] https://github.com/camaraproject

[13] https://www.bridgealliance.com/

[14] https://www.gsma.com/foundry/bridge-alliance-federated-edge-hub/

5.5 Industry best practice for securing edge native apps

The establishment of a uniform level of security policies, procedures, and Minimum Baseline Security Standard (MBSS) for all network elements is extremely important for edge native application design.

In particular, the structural heterogeneity and distribution of the edge network, the diverse ecosystem in computing nodes and devices, results in a coarse degree of data access management, and malicious actors may penetrate the core of the edge device. Untrusted computing nodes joining the network may hack user data at the edge network and interrupt the operation. Additionally, because of the performance limitations of edge nodes, these devices can hardly resist to network attacks, such as man-in the-middle attacks and denial of service attacks, which leads to the breakdown of the edge network and instability. For edge security, building a secure supply-chain is vital, vendor compliance is a necessity, security assurance, vulnerability, integrity of any third-party elements together with trust and privacy of far edge devices like IoT ecosystem is also extremely important. Attacks and issues that compromise privacy and security often occur in three conditions of edge networks: the infrastructure layer security, network layer, and application layer. Figure 9 below provides a possible Edge Native security Framework (thus not necessarily related to an ETSI MEC standard), which can be considered as reference for edge native application design.

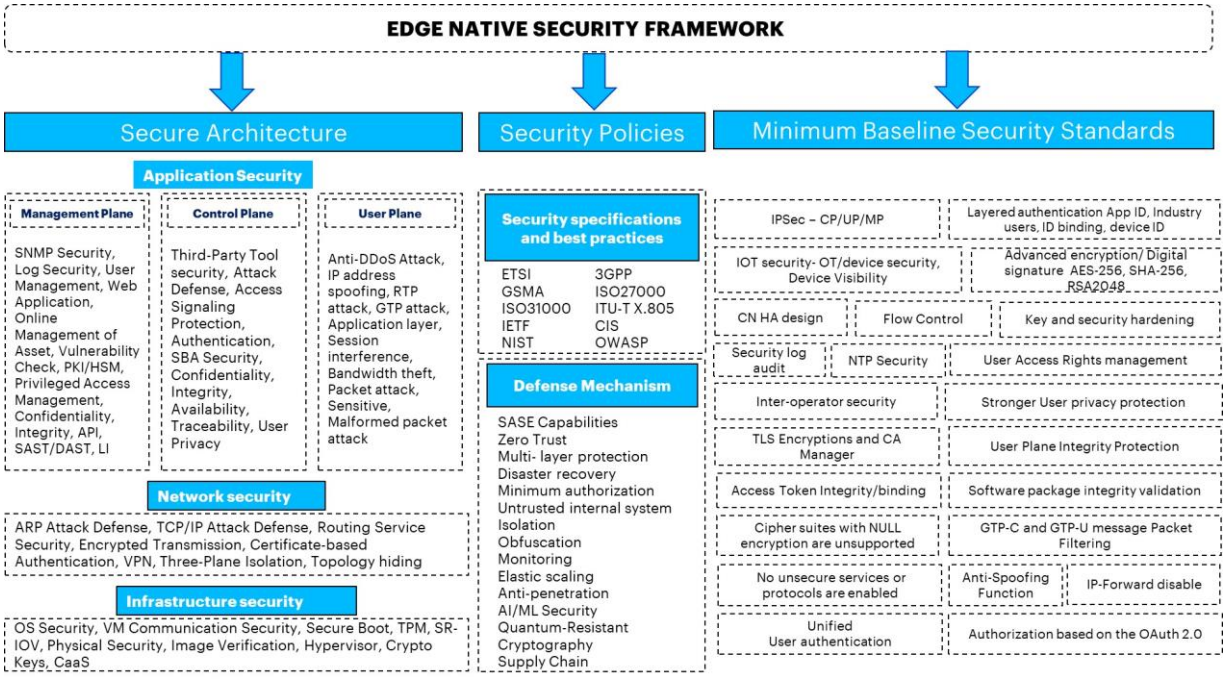

Figure 9: Edge Native security Framework

6 Final remarks and Future Evolutions

In this White Paper we have provided an overview and vision about the Edge Native approach, as a natural evolution of Cloud Native, referring to *building and running edge applications which take edge environment characteristics in mind and can best utilize the capabilities of the edge environment*. The primary target for this work is identified by edge application developers and the general technical community interested in MEC solutions or Edge Native application design concepts.

In this paper we first showed the unique requirements for edge applications, described what Edge Native is and how the ETSI MEC and other organizations in the industry can support the Edge Native design by helping the developers to meet the requirements. Also, the paper included some relevant examples of use cases that can benefit from the introduction of edge native application design, with some possible perspectives for further work in ETSI MEC. Notably, with the recent emergence of MEC federation (triggered by GSMA Operator Platform Group, and the more recent creation of CAMARA and Open Gateway initiatives), the concept of Edge Native has entered into a new phase of cross operators (multi-operators). More exactly, all the works done in GSMA OPG/OPAG and CAMARA are aiming to implement an interconnected and interoperable global Edge Computing ecosystem. Edge Applications can be thus deployed across operator networks and often even outside their trust domains. Consequently, in this heterogeneous ecosystem of stakeholders, even if the role of standards is key, the paper outlined the key role of industry communities and open-source projects.

In this regard, it is worth noting that the definition and implementation of Service APIs in projects like CAMARA (tightly linked with GSMA and Open Gateway) are promising to deliver the needed functionalities to edge native application developers and customers, without the burden of requiring them to become experts of the underlying edge infrastructure. The ETSI MEC standard (synergized with 3GPP specifications) can offer a footprint for interoperability, API basic design principles to ensure universal adoption, and possibly also some guidelines for API abstraction, while the actual implementations of Service APIs from open-source projects can guarantee wide adoption from developers' communities.

In this perspective, collaboration between open source and standards will be needed, as they can provide complementary benefits to the community and finally the whole edge computing market. In the view of a full exploitation of edge capabilities, the adoption of edge native design principles from application development communities will need joint efforts from open source and standards, done also by "non-traditional" collaborations, harmonization and engagement with all industry projects and organizations.

Annex A: MEC Support for Application Development

With the ISG's establishment of the MEC API framework, facilitated through the creation of ETSI GS MEC 009 [14] on API design principles and guidelines, and the associated MEC service APIs, it became increasingly apparent with the ISG that there could be further opportunities for software developer engagement, with a particular focus on potential MEC application developers.

To provide focus to such activities the Deployment and Ecosystem Development Work Group (aka DECODE WG) was established in December 2018, with the aim of:

- Showcasing MEC technology and standards
- Increasing operator adoption and
- Engaging with application developers

The group's first deliverable in support of developer engagement is the creation of OpenAPI™ Specification [20] compliant descriptions for all the service APIs specified by the ISG, publicly available through ETSI Forge[15]. These comprehensive descriptions inherently include each MEC API information model and its service endpoints. They are both human readable and perhaps most importantly are machine readable. This enables content validation and also auto-creation of communication stubs for both the client and server, which is highly powerful when commencing application development. The files are provided with a BSD Clause 3 license for use. Further developer support has been provided through the creation of the MEC Tech Series, which provides short informative video podcasts on a wide range of MEC related topics through the ETSI YouTube channel[16].

Another area in scope of DECODE are the ETSI Plugtests, where MEC has been a joint headliner with NFV at the biannual event. Both interoperability testing and API conformance are in scope. The events provide an excellent opportunity for developers to test their MEC applications in a friendly environment with a wide range of manufacturers' product offerings. The Plugtests are also very well aligned with DECODE's objective to enable operator adoption by demonstrating interoperability between the different MEC deployments.

To facilitate such testing efforts, the group developed and published a test framework, ETSI GS MEC-DEC 025 [21]. This has been used as the basis to develop API Conformance specifications, with associated test suites, GS MEC-DEC 032 [22]. Like the OpenAPI based descriptions, the test suites are publicly available and accessible via ETSI Forge repositories. The suites have been developed in both TTCN-3 and the Robot Framework to cater for both the Telecommunication and IT communities.

For more mature MEC solutions that are available in the public domain, the MEC ecosystem wiki[17] has been developed to showcase those as part of the overall MEC wiki. This provides an excellent reference for developers starting out on MEC application development journey. The wiki also provides links and information on the ongoing and completed proof of concepts and MEC deployment trails.

In 2021, the DECODE WG launched the ETSI MEC Sandbox[18], which is an online environment for developers to interact with MEC Service APIs and learn how they may enable their Edge Native applications. The MEC

[15] https://forge.etsi.org/rep/mec
[16] https://mecwiki.etsi.org/index.php?title=MEC_Tech_Series
[17] https://mecwiki.etsi.org/index.php?title=MEC_Ecosystem
[18] https://try-mec.etsi.org/

Sandbox emulates an Edge System set in the city of Monaco, offering a variety of selectable edge configurations including 4G, 5G, and Wi-Fi access networks. Developers interact with "live" MEC service APIs and experience their run-time behavior, including how to discover MEC services and how to offer new MEC services into the MEC System.

Currently, the Sandbox offers a variety of **MEC services** including:

- ETSI GS MEC 011 – MEC Application Support and MEC Service Management [12]
- ETSI GS MEC 012 – Radio Network Information Service [23]
- ETSI GS MEC 013 – Location Service [24]
- ETSI GS MEC 015 – Bandwidth Management and Multi-Access Traffic Steering Services [25]
- ETSI GS MEC 016 – Device Application Interface [26]
- ETSI GS MEC 021 – Application Mobility Service [9], including multiple MEC platform points of presence
- ETSI GS MEC 028 – WLAN Access Information Service [27]
- ETSI GS MEC 030 – V2X Service [28]

Additional services may be introduced into the MEC Sandbox in the future. To learn more about the Sandbox, please visit about the MEC Sandbox wiki[19]. Examples of Edge Native applications utilizing the MEC Sandbox in their development can be found in the results of recent MEC Hackathons:

- ETSI/Linux Foundation - Edge Hackathon 2022[20]
- Edge Computing World - MEC Hackathon 2021[21]

In summary, it can be seen that DECODE WG, through all of its activities and deliverables, provides significant support for potential MEC application developers as they embrace edge native design and development.

[19] https://mecwiki.etsi.org/index.php?title=MEC_Sandbox
[20] https://mecwiki.etsi.org/index.php?title=ETSI_-_LF_MEC_Hackathon_2022
[21] https://mecwiki.etsi.org/index.php?title=MEC_Hackathon_2021_Edge_Computing_World

Annex B: Service API exposure and interoperable API consumption

As explained in the paper, MEC App registration (introduced by ETSI GS MEC 011 [12]) supports various cases i.e., a MEC App instance instantiated by MEC management, and a MEC App instance not instantiated by MEC management (in both sub-cases where there is or there isn't an EAS profile available).

This is key for application developer perspective, also because the MEC platform (as explained in section 4.3) offers an environment to run MEC applications and also enable them to *provide* and *consume* MEC services [12]). This essentially allows any new MEC services (e.g., produced by a service producing MEC App 1) to be discoverable and consumable at application level (e.g., a further MEC App 2 consuming at higher level those services). Let's explain this with more background.

In fact, it can enable a number of interoperability scenarios:

1. A MEC application (instantiated by MEC management) may discover and consume a MEC service registered to the MEC platform (where the service can be e.g., produced by a MEC App, which on its turn can be or not instantiated by MEC management).
2. A MEC application (not instantiated by MEC management, but with an EAS profile available) may discover and consume a MEC service registered to the MEC platform (where the service can be e.g., produced by a MEC App, which on its turn can be or not instantiated by MEC management).
3. A MEC application (not instantiated by MEC management, and without an EAS profile available) may discover and consume a MEC service registered to the MEC platform (where the service can be e.g., produced by a MEC App, which on its turn can be or not instantiated by MEC management).

The second scenario is particularly interesting for ETSI MEC and 3GPP interworking, as it opens the possibility for EAS to invoke MEC services, and dually exposing Service APIs provided by EES or EAS to MEC Applications. These dual possibilities are also coherent with 3GPP TR 23.700-98 [29], where Solution #11 (to KI#5) specifies a CAPIF deployment option for: a) allowing EASs to invoke MEC services, and b) exposing Service APIs provided by EES or EAS (as described in KI#2) to MEC Applications.

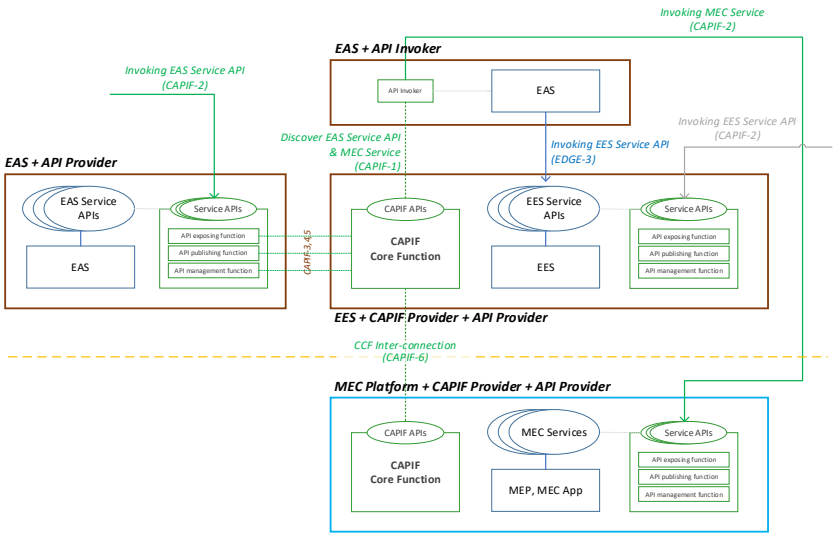

Figure B.1: A deployment option for 3GPP alignment with ETSI MEC using CAPIF (Ref. TR 23.700-98 [29])

As a further variant of the above scenario, the reader may notice also that a MEC application can be running also outside the domain where the MEC service is produced (example in the Figure B.2 below). This can be a further enablement for third parties, which can implement middleware services to be exposed to edge native application developers.

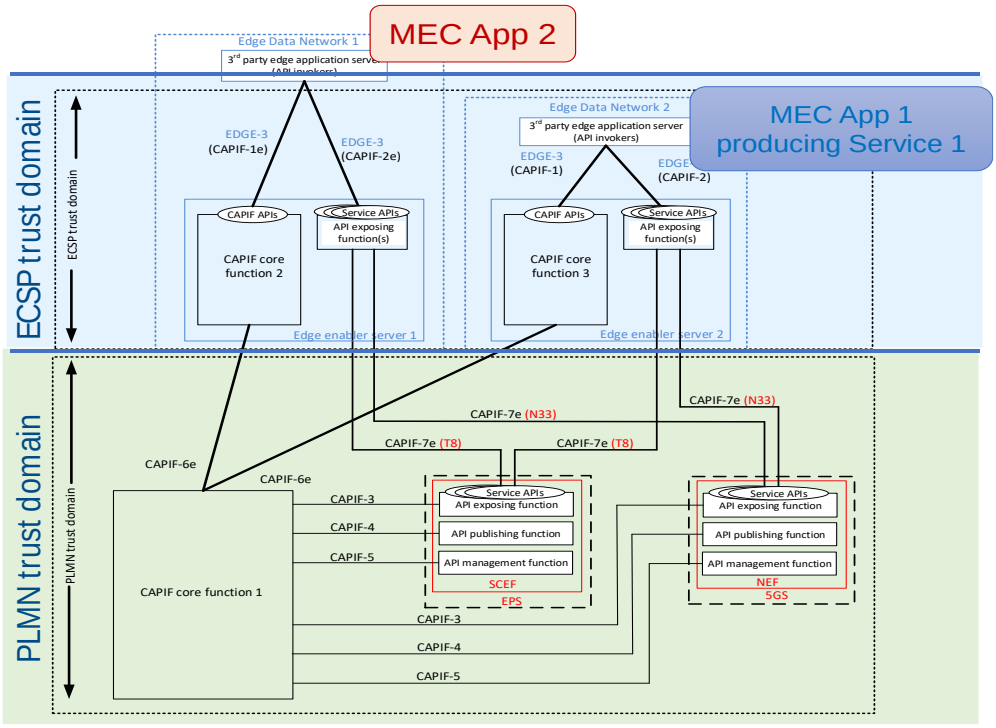

Figure B.2 Example of MEC application running outside the ECSP trust domain, and consuming (via CAPIF) a MEC service produced by another MEC application in the ECSP trust domain (ref. TR 23.558 [37])

Finally, the third scenario is particularly interesting for ETSI MEC interworking with other management systems, e.g., Kubernetes-based edge native applications. In fact, providing that a MEC application (not instantiated by MEC management, and without an EAS profile available) is registered to the MEC system, it can consume any services registered to the MEC platform. This is indeed a powerful mechanism, which allows further interoperability among various systems, and cross-system API consumption enabled by the MEC application registration procedure in ETSI MEC.

In summary, putting all together the three scenarios, any MEC applications to consume any new MEC services (e.g., produced by a service producing MEC App 1), which are indeed discoverable and consumable at application level thanks to the registration procedure (which is of course optional for the developer).

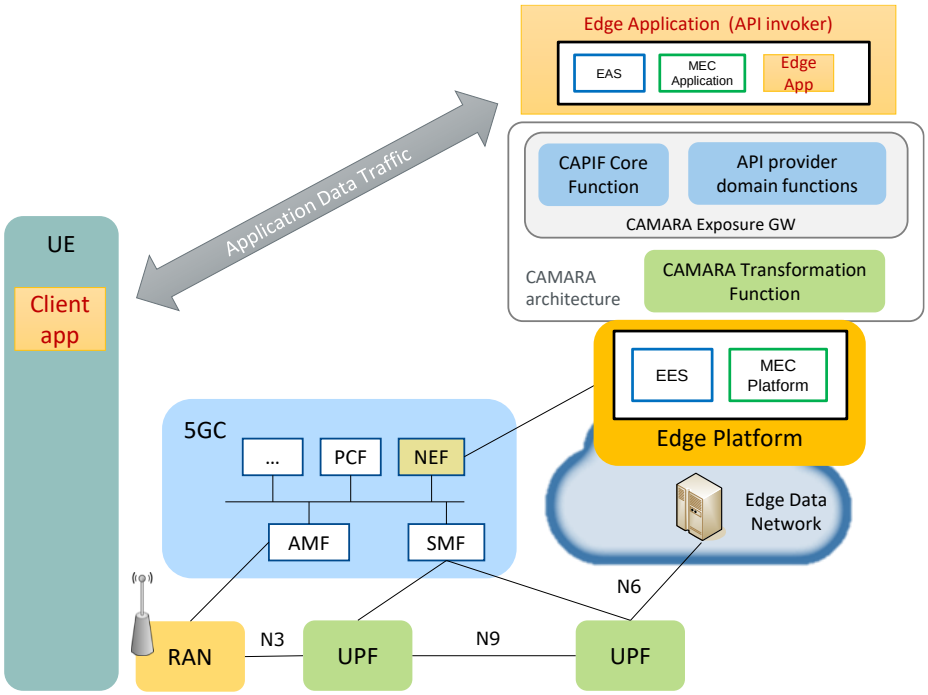

Figure B.3: Option for Edge Native applications to consume MEC services in a MEC federation (via CAPIF framework and the CAMARA architecture).

A further advantage of this interoperable mechanism can be found also in the framework of MEC federation, where the usage of CAPIF as exposure gateway (which could be adopted also as reference implementation in the CAMARA architecture) is also interesting to allow cross-system API consumption, where e.g., multiple MEC platforms from different MEC systems are exposing their capabilities and services to authorized applications in the federation. Figure B.3 shows a convenient example, with a co-located implementation of EES and MEC platform, where any edge native applications are seen as API invokers and can consume services exposed (via the CAMARA architecture) by the underlying platform. This option also facilitates the synergies with ETSI MEC and GSMA OPG architecture, as API exposure can be exploited also in the MEC federation for edge native application development.

Annex C: Definition of abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
CAPIF	Common API Framework
CI/CD	continuous integration and continuous delivery/continuous deployment
CDN	Content Delivery Network
EAS	Edge Application Server
EES	Edge Enabler Server
HAP	High-Altitude Platform
IoT	Internet of Things
KPI	Key Performance Indicator
MEC	Multi-access Edge Computing
ML	Machine Learning
NF	Network Functions
RAN	Radio Access Network
RNIS	Radio Network Information Service
RSU	Road-Side Unit
SCADA	Supervisory control and data acquisition
UAV	Uncrewed Aerial Vehicle
vCPE	virtual CPE
V2X	Vehicle-to-Everything
VIS	V2X Information Services
VR	Virtual Reality

Annex D: References

[1] Eclipse Foundation, Edge Native Working Group, https://edgenative.eclipse.org/

[2] 5GDNA, Edge Native Technical Architecture White Paper 1.0, https://www.huawei.com/en/news/2021/2/edge-native-paper

[3] State of the Edge, Open Glossary of Edge Computing, https://github.com/State-of-the-Edge/glossary/blob/master/edge-glossary.md#edge-native-application

[4] ETSI White Paper No. 30, MEC in an Enterprise Setting: A Solution Outline, 1st Ed., Sept 2018, www.etsi.org/images/files/ETSIWhitePapers/etsi_wp30_MEC_Enterprise_FINAL.pdf

[5] ETSI White Paper No. 24, MEC Deployments in 4G and Evolution Towards 5G, 1st Ed., Feb 2018, www.etsi.org/images/files/ETSIWhitePapers/etsi_wp24_MEC_deployment_in_4G_5G_FINAL.pdf

[6] T-220019: 5GAA Technical Report "Use Case Implementations for Sensor Data Sharing", Feb 2023, https://5gaa.org/use-case-implementations-for-sensor-data-sharing/

[7] SAE_J3224: SAE "V2X Sensor-Sharing for Cooperative and Automated Driving", Aug 2022, https://www.sae.org/standards/content/j3224_202208/

[8] ETSI GS MEC 003 V3.1.1 (2022-02): "Multi-access Edge Computing (MEC); Framework and Reference Architecture", Link: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/03.01.01_60/gs_MEC003v030101p.pdf

[9] ETSI GS MEC 021 V2.2.1 (2022-02): Multi-access Edge Computing (MEC); Application Mobility Service API. https://www.etsi.org/deliver/etsi_gs/MEC/001_099/021/02.02.01_60/gs_MEC021v020201p.pdf

[10] ETSI GS MEC 002 V3.1.1 (2023-04): Multi-access Edge Computing (MEC); Use Cases and Requirements https://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/03.01.01_60/gs_MEC002v030101p.pdf

[11] ETSI GS MEC 040 V3.1.1 (2023-02), Multi-access Edge Computing (MEC); Federation enablement APIs, https://www.etsi.org/deliver/etsi_gs/MEC/001_099/040/03.01.01_60/gs_MEC040v030101p.pdf

[12] ETSI GS MEC 011 V3.1.1 (2022-09): Multi-access Edge Computing (MEC); Edge Platform Application Enablement https://www.etsi.org/deliver/etsi_gs/MEC/001_099/011/03.01.01_60/gs_MEC011v030101p.pdf

[13] ETSI GS MEC 010-2 V2.2.1 (2022-02): Multi-access Edge Computing (MEC); MEC Management; Part 2: Application lifecycle, rules and requirements management https://www.etsi.org/deliver/etsi_gs/MEC/001_099/01002/02.02.01_60/gs_MEC01002v020201p.pdf

[14] ETSI GS MEC 009 V3.2.1 (2022-07), Multi-access Edge Computing (MEC); General principles, patterns and common aspects of MEC Service APIs, https://www.etsi.org/deliver/etsi_gs/MEC/001_099/009/03.02.01_60/gs_MEC009v030201p.pdf

[15] ETSI White Paper No. 49 (2022-06): MEC federation: deployment considerations https://www.etsi.org/images/files/ETSIWhitePapers/ETSI_WP_49_MEC-Federation-Deployment-considerations.pdf

[16] ETSI White Paper No. 46, MEC security; Status of standards support and future evolutions, 2nd edition – September 2022, https://www.etsi.org/images/files/ETSIWhitePapers/ETSI-WP-46-2nd-Ed-MEC-security.pdf

[17] ETSI GS NFV-SEC 001 V1.1.1 (2014-10): "Network Functions Virtualisation (NFV); NFV Security; Problem Statement", Link: https://www.etsi.org/deliver/etsi_gs/nfvsec/001_099/001/01.01.01_60/gs_nfv-]sec001v010101ppdf

[18] ETSI GS NFV-SEC 009 V1.1.1 (2015-12): "Network Functions Virtualisation (NFV); NFV Security; Report on use cases and technical approaches for multi-layer host administration", Link: https://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/009/01.01.01_60/gs_nfvsec009v010101p.pdf

[19] Network Equipment Security Assurance Scheme – Overview; Version 2.0", 05 February 2021, Link: https://www.gsma.com/security/wp-content/uploads/2021/02/FS.13-NESAS-Overview-v2.0.pdf

[20] OpenAPI Initiative, https://www.openapis.org/

[21] ETSI GR MEC-DEC 025 V2.1.1 (2019-06), Multi-access Edge Computing (MEC); MEC Testing Framework, https://www.etsi.org/deliver/etsi_gr/MEC-DEC/001_099/025/02.01.01_60/gr_mec-dec025v020101p.pdf

[22] ETSI GS MEC-DEC 032-1 V3.1.1 (2022-04), Multi-access Edge Computing (MEC); API Conformance Test Specification;
Part 1: Test Requirements and Implementation Conformance Statement (ICS), https://www.etsi.org/deliver/etsi_gs/MEC-DEC/001_099/03201/03.01.01_60/gs_MEC-DEC03201v030101p.pdf
Part 2: Test Purposes (TP), https://www.etsi.org/deliver/etsi_gs/MEC-DEC/001_099/03202/03.01.01_60/gs_MEC-DEC03202v030101p.pdf
Part 3: Abstract Test Suite (ATS), https://www.etsi.org/deliver/etsi_gs/MEC-DEC/001_099/03203/03.01.01_60/gs_MEC-DEC03203v030101p.pdf

[23] ETSI GS MEC 012 V2.2.1 (2022-02): Multi-access Edge Computing (MEC); Radio Network Information API https://www.etsi.org/deliver/etsi_gs/MEC/001_099/012/02.02.01_60/gs_MEC012v020201p.pdf

[24] ETSI GS MEC 013 V3.1.1 (2023-01): Multi-access Edge Computing (MEC); Location API https://www.etsi.org/deliver/etsi_gs/MEC/001_099/013/03.01.01_60/gs_MEC013v030101p.pdf

[25] ETSI GS MEC 015 V2.2.1 (2022-12): Mobile Edge Computing (MEC); Bandwidth Management API https://www.etsi.org/deliver/etsi_gs/MEC/001_099/015/02.02.01_60/gs_MEC015v020201p.pdf

[26] ETSI GS MEC 016 V2.2.1 (2020-04): Multi-access Edge Computing (MEC); Device application interface https://www.etsi.org/deliver/etsi_gs/MEC/001_099/016/02.02.01_60/gs_MEC016v020201p.pdf

[27] ETSI GS MEC 028 V2.3.1 (2022-07): Multi-access Edge Computing (MEC); WLAN Access Information API Ihttps://www.etsi.org/deliver/etsi_gs/MEC/001_099/028/02.03.01_60/gs_MEC028v020301p.pdf

[28] ETSI GS MEC 030 V3.1,1 (2023-03): Multi-access Edge Computing (MEC); V2X Information Services API https://www.etsi.org/deliver/etsi_gs/MEC/001_099/030/03.01.01_60/gs_MEC030v030101p.pdf

[29] 3GPP TR 23.700-98 "Study on enhanced Architecture for enabling Edge Applications; (Release 18)"

[30] F. Granelli, C. Costa, J. Zhang, R. Bassoli and F. H. P. Fitzek, "Design of an On-Demand Agile 5G Multi-Access Edge Computing Platform Using Aerial Vehicles," in *IEEE Communications Standards Magazine*, vol. 4, no. 4, pp. 34-41, December 2020, doi: 10.1109/MCOMSTD.001.2000016.

[31] Mahadev Satyanarayanan et al., “Sinfonia: Cross-Tier Orchestration for Edge-Native Applications”, July 2022, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Available at: http://reports-archive.adm.cs.cmu.edu/anon/2022/CMU-CS-22-125.pdf

[32] Eclipse Foundation, “From DevOps to EdgeOps: A Vision for Edge Computing”, White Paper, 2022

[33] OpenAI, “GPT-4 Technical Report”, arXiv:2303.08774, 2023

[34] Zhang, Y., et al. “Google USM: Scaling Automatic Speech Recognition Beyond 100 Languages.” arXiv:2303.01037, 2023

[35] Aharoni et al. “Massively Multilingual Neural Machine Translation.” arXiv:1903.00089, 2019

[36] Borsos et al. “AudioLM: a Language Modeling Approach to Audio Generation”, arXiv:2209.03143, 2022

[37] 3GPP TR 23.558: “Study on application architecture for enabling Edge Applications”

ETSI
06921 Sophia Antipolis CEDEX, France
Tel +33 4 92 94 42 00
info@etsi.org
www.etsi.org