H2020 5G Dive Project
Grant No. 859881

D1.3: 5G-DIVE Final Architecture

Abstract

This deliverable focuses on the complete version of the 5G-DIVE architecture, detailing the architectural components and functions defining the DEEP platform and its integration with the 5G-CORAL architecture.

Document properties

Document number	D1.3
Document title	5G-DIVE Final Architecture
Document responsible	UC3M
Document editor	Carlos Guimarães (UC3M)
Editorial team	Carlos Guimarães (UC3M), Milan Groshev (UC3M), Ivan Paez (ADLINK), Luca Cominardi (ADLINK), Timothy William (NCTU), Javier Sacido (TELCA), Aitor Zabala (TELCA), Matteo Pergolesi (TELCA), Luis M. Contreras (TID), Alberto Solano (TID), Samer Talat (ITRI), Chenguang Lu (EAB), Chao Zhang (ULUND), Filipe Conceição (IDCC), Ibrahim Hemadeh (IDCC)
Target dissemination level	Public
Status of the document	Final
Version	1.0

Production properties

Reviewers	Matteo Pergolesi (TELCA), Milan Groshev (UC3M), Samer Talat (ITRI), Antonio de la Oliva (UC3M), Tzu Ya Wang (III)

Document history

Revision	Date	Issued by	Description
1.0	29/06/2021	Carlos Guimarães (UC3M)	Final version

Disclaimer

This document has been produced in the context of the 5G-DIVE Project. The research leading to these results has received funding from the European Community's H2020 Programme under grant agreement Nº H2020-859881.

Contents

List of Tables

List of Figures

List of Acronyms

3GPP: 3rd Generation Partnership Project

ADS: Autonomous Drone Scouting

AF: Application Function

AI: Artificial Intelligence

AMF: Access and Mobility Management Function

AP: Average Precision

API: Application Programming Interface

BASS: Business Automation Support Stratum

BSS: Business Support System

CEF: Charging Enablement Function

DASS: Data Analytics Support Stratum

DCCF: Data Collection Coordination Function

DDS: Data Distribution Service

DEEP: 5G-DIVE Elastic Edge Platform

EBI: EastBound Interface

EFS: Edge and Fog computing System

ETSI: European Telecommunications Standards Institute

GSMA: Global System for Mobile Communications

IaaS: Infrastructure-as-a-Service

ICT: Information and Communication Technology

IESS: Intelligence Engine Support Stratum

KPI: Key Performance Indicator

MEC: Multi-access Edge Computing

MEO: Multi-access Edge Computing Orchestrator

MEP: Multi-access Edge Computing Platform

MEPM: Multi-access Edge Computing Platform Manager

MFAF: Messaging Framework Adaptor Function

ML: Machine Learning

mMTC: massive Machine-Type Communication

MNO: Mobile Network Operator

MTLF: Model Training Logical Function

NBI: NorthBound Interface

NDN: Named Data Networking

NEF: Network Exposure Function

NF: Network Function

NFV: Network Function Virtualization

NFVI: Network Function Virtualization Infrastructure

NPN: Non-Public Network

NRF: Network Repository Function

NSD: Network Service Descriptor

NSSF: Network Slice Selection Function

NWDAF: Network Data Analytics Function

OAM: Operations, Administration and Maintenance Function

OCS: Orchestration and Control System

OP: Operator Platform

OSS: Operations Support System

PaaS: Platform-as-a-Service

PCF: Policy Control Function

PLMN: Public Land Mobile Network

PNI: Public Network Integration

RDBMS: Relational DataBases Management System

SA: System Aspects

SBI: SouthBound Interface

SCP: Service Communication Proxy

SDO: Standards Development Organization

SLA: Service Level Agreement

SLAM: Service Level Agreement Manager

SLAT: Service Level Agreement Template

SLI: Service Level Indicator

SLO: Service Level Objective

SMF: Session Management Function

SSP: System & Service Provider

TLS: Transport Layer Security

TSG: Technical Specifications Group

UC: User Client

UDM: Unified Data Management

UDR: Unified Data Repository

UNI: User-Network Interface

UPF: User Plane Function

URI: Uniform Resource Identifier

VI: Virtualization Infrastructure

VIM: Virtualization Infrastructure Manager

VNF: Virtual Network Function

VNFD: Virtual Network Function Descriptor

VSB: Vertical Service Blueprint

VSD: Vertical Service Descriptor

VSI: Vertical Service Instance

VSS: Vertical Support System

WBI: WestBound Interface

WP: Work Package

ZDM: Zero-Defect Manufacturing

Executive Summary

This deliverable reports on the work carried in 5G-DIVE Work Package (WP) 1 with respect to the design of the 5G-DIVE baseline architecture. It departs from the initial design reported in D1.1 [1] and refines the proposal of the 5G-DIVE Elastic Edge Platform (DEEP) and its supporting strata, according to the outcomes and results of the other project work packages (in particular, WP2 and WP3).

The main achievements of this deliverable include:

- A self-contained description of the 5G-DIVE baseline architecture, detailing the DEEP platform and its supporting strata, and their main building blocks, in order to enable a paradigm shift from an Infrastructure-as-a-Service (IaaS) service model towards a Platform-as-a-Service (PaaS) service model.
- The identification of the key innovations implemented by the DEEP platform to empower the vertical with enhanced and abstracted automation and intelligence capabilities.
- Integration options of the DEEP platform with the underlying Edge and Fog Computing Infrastructures, focusing on additional details regarding the integration with the 5G-CORAL architecture.
- A mapping of the 5G-DIVE solution and DEEP platform to 3GPP standards, showcasing the alignment of the work with different standardization activities.

All the findings in this deliverable have been continuously fed into the ongoing work within WP2 and WP3.

1. Introduction

This deliverable presents an updated and complete description of the 5G-DIVE architecture and, in particular, on the 5G-DIVE Elastic Edge Platform (DEEP). It departs from its initial definition reported in D1.1 [1], providing a refined and final design motivated by the research, implementation and integration tasks conducted in the project. Namely, it leverages on high-level conceptual discussions from WP1 as well as implementation insights from WP2 and integration outcomes within the vertical pilots and use cases from WP3. Moreover, it extends the previous initial design with additional and more detailed descriptions of its supporting strata, workflows for the main procedures, and integration with the underlying Edge and Fog computing infrastructures.

For such purposes, the remaining of this deliverable is structured as follows:

- Section 2 presents the final design of the 5G-DIVE solution, including a high-level view of the DEEP platform and its supporting strata.
- Section 3 describes the main innovations introduced by the 5G-DIVE solution with respect to its abstraction, intelligence, and automation capabilities.
- Section 4 overviews the integration of the DEEP platform with a selected Edge Computing Platform, namely 5G-CORAL platform.
- Section 5 presents a mapping of the DEEP platform to the relevant 3GPP standards.
- Finally, Section 6 enumerates the main conclusions, pointing out for the future work.

2. Final Design of 5G-DIVE Solution

This section presents the final design of the 5G-DIVE solution and of its supporting DEEP platform. It departs from the initial definition, as presented in D1.1 [1], not only positioning it over a complete end-to-end (E2E) architecture (Section 2.1) and presenting the refinements over the main supporting strata (Section 2.2) but also describing the workflows with respect to baseline operations performed over the DEEP platform (Section 2.3).

2.1. End-to-End 5G-DIVE Architecture Overview

In this section, we first provide a complete overview of the 5G-DIVE architecture over an E2E solution, comprising not only different technology domains but also different administrative domains. When positioned over such E2E solution, the 5G-DIVE architecture makes available an *Intelligent Edge and Fog 5G E2E* solution to the vertical industries, as depicted in Figure 2-1.

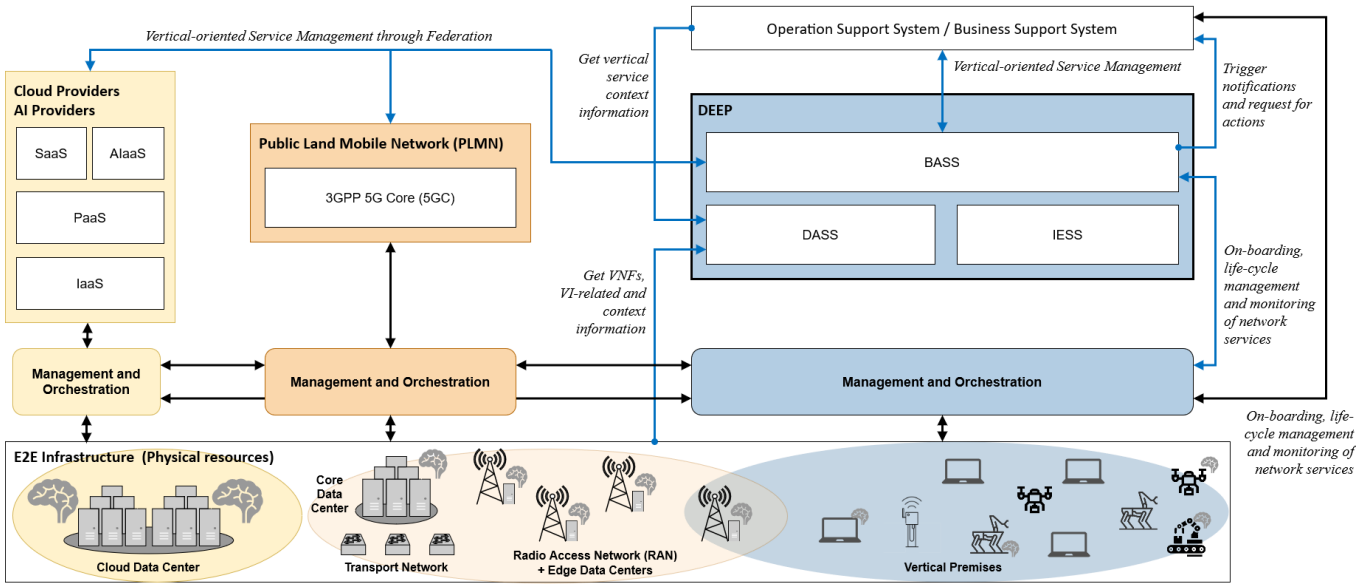


FIGURE 2-1: E2E 5G-DIVE ARCHITECTURE OVERVIEW

In 5G-DIVE architecture, the DEEP platform is positioned as a vertical-oriented platform, acting as the entry point to handle all the interaction with the vertical and its OSS/BSS. In doing so, it aims to abstract the vertical from the extremely heterogeneous infrastructures under different administrative domains and owned by several stakeholders. Moreover, it is the common integration point for different administrative and technological domains. This is a critical aspect to aid in the digital transformation of several vertical industries since verticals still lack overall knowledge about Information and Communications Technologies (ICTs) to manage them on their own. Thus, by hiding the complexity from the vertical, 5G-DIVE is contributing to the adoption of 5G and supporting technologies by different vertical industries.

In the scope of the vertical industries addressed in 5G-DIVE, Figure 2-1 presents possible extensions of the vertical domain through multi-domain interactions of the DEEP platform with three other technological domains: *(i)* 3GPP 5G Core provider; *(ii)* Cloud provider; and *(iii)* AI provider.

1. **3GPP 5G Core Provider:** 5G is introducing the concept of private 5G networks, also known as Non-Public Networks (NPNs). These can be of two types [2][3]: *(i)* standalone NPNs; and *(ii)* Public Network Integrated-NPNs (PNI-NPNs). While in the former the vertical implements a fully operational 5G operational network, in the latter the vertical leverages on functions and capabilities made available by the Public Land Mobile Network (PLMN) (different degrees of integration are envisioned). As such, the DEEP platform interacts with the PLMN platform, making use of exposed APIs by the latter, to request the deployment and provisioning of a specific PNI-NPN option that fits the requirements of the vertical.
2. **Cloud and Edge Providers:** Although the 5G-DIVE solution envisions that the vertical manages a set of local resources under its domain (that we refer as the Fog infrastructure), they might be insufficient to fulfil the computational demands of the vertical services. Moreover, the usage of third-party resources might be cheaper and, therefore, preferrable by the vertical. Thus, vertical services need to be federated across external Edge or Cloud infrastructures. To do so, the DEEP platform intermediates the deployment of E2E vertical services across different administrative domains, considering the privacy of the services and data as well as the communication requirements. Moreover, the DEEP platform can also intermediate resource federation, where resources are leased by the external provider to the sole usage of the vertical.
3. **AI Provider:** Other than computational resources as described in the previous point, the DEEP platform also intermediates the usage of AI services provided by an external provider. As such, in case the DEEP platform is not able to fulfil any intelligence capability requested by the vertical, it can delegate its provisioning to third-party providers, either for training or run-time purposes. Note that, exposing the private and sensitive data of the vertical is a critical aspect that the DEEP platform must take into consideration. In case of any conflict that makes it impossible to be provided either locally or externally, the vertical needs to provide additional requirements or to waive those intelligence capabilities.

The positioning of the DEEP platform in an E2E solution is in line with the role of *System & Service Provider* as introduced in D1.2 [4], in which relationships with other stakeholders are described in Figure 2-2. For more details, please refer to D1.2.

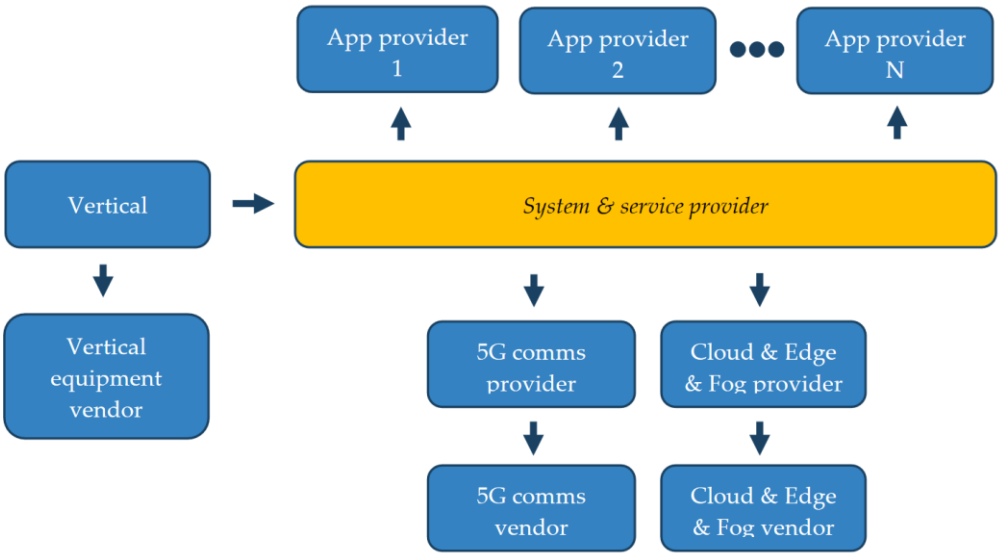


FIGURE 2-2: *SYSTEM & SERVICE PROVIDER* STAKEHOLDERS' FLOWS

The underlying Edge Computing Infrastructure might comprise different technology domains and/or belong to different administrative domains. In the scope of the integration with the DEEP platform, an Edge Computing Infrastructure is comprised of an orchestrator and virtualization infrastructure. Thus, the DEEP platform interacts with the *Management and Orchestration* platforms of such infrastructure(s), as depicted in Figure 2-3, embody different integration schemes in order to provide flexibility to a wide range of scenarios. Each of these options are briefly introduced as follows:

- **Option A:** The DEEP platform interacts with one or more *Management and Orchestration* platform(s), each managing a different set of resources of isolated technological / administrative domains.
- **Option B:** The DEEP platform interacts with one or more *Management and Orchestration* platform(s), all managing the same set of resources of a given technological / administrative domains.
- **Option C:** The DEEP platform interacts with one or more *Management and Orchestration* platform(s), each managing resources of a given technological / administrative domains but at different levels.

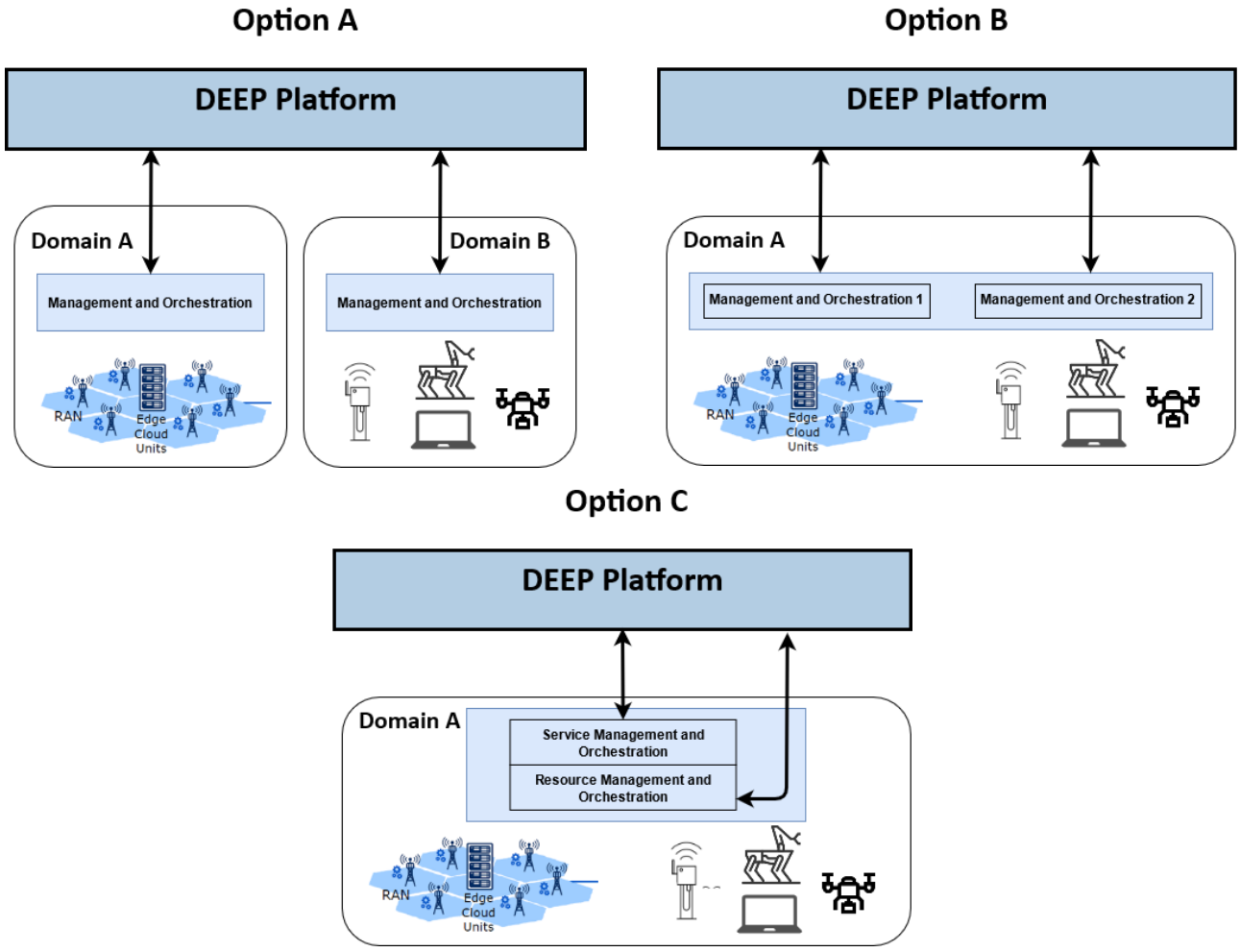


FIGURE 2-3: INTEGRATION OPTIONS WITH UNDERLYING EDGE COMPUTING INFRASTRUCTURES

Implementation details for Option A are reported in D2.3 [5] as the selected approach for the integration with the pilots' use cases.

2.2. DEEP Platform

The internal architecture of DEEP platform, including its three supporting strata (DASS, IESS, and BASS), is depicted in Figure 2-4, as well as the interfaces between the different components of the DEEP, the VSS/OSS/BSS, and the Edge Computing Infrastructure.

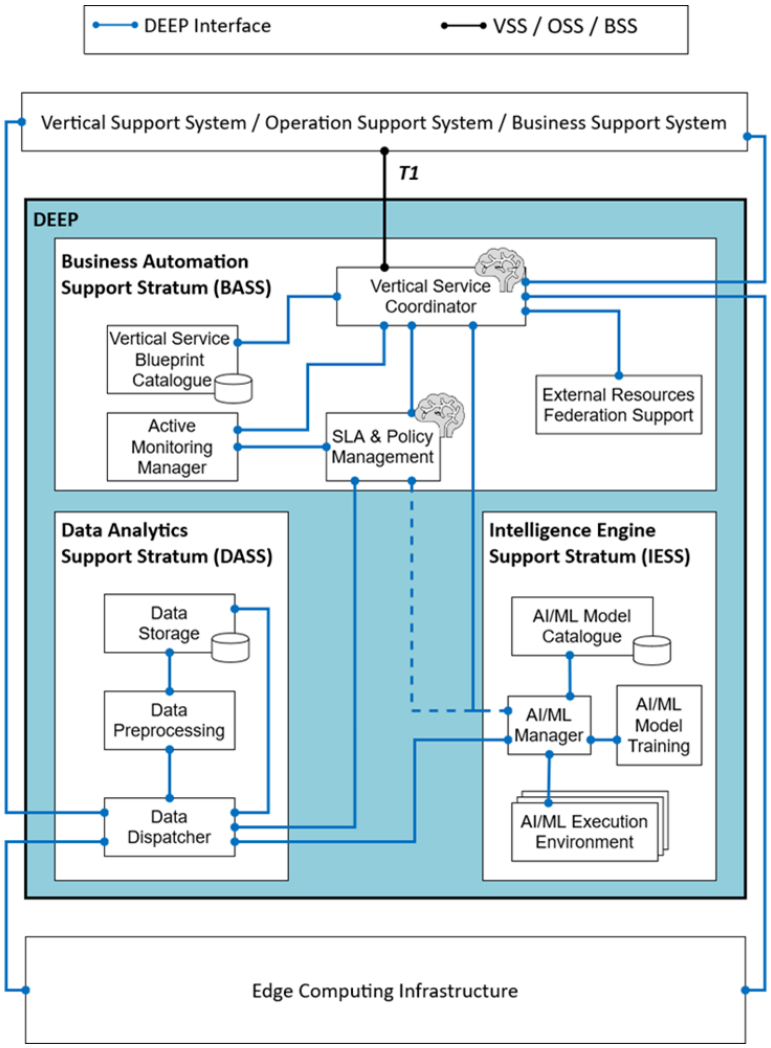


FIGURE 2-4: DEEP INTERNAL ARCHITECTURE

A few improvements must be highlighted in relation to the initial architecture design presented in D1.1 [1], which has been refined with inputs from the implementation stage and the use cases integration. The major changes are summarized below, and discussed in more details in the following subsections:

- Refinement of the internal interaction between DASS components.
- In the BASS, the *Active Monitoring* component has now evolved to an *Active Monitoring Manager*. Several of its internal connections were also revised.
- The *SLA & Policy Management* is extended with direct interfaces towards the IESS.
- In the IESS, a more fine-grained component design has been devised, where the *AI/ML Manager* was introduced to take some of the tasks initially assign to the *AI/ML Model Training*.

2.2.1. Data Analytics Support Stratum (DASS)

The DASS offers the necessary support for pre-processing, storing, and sharing data generated by a variety of sources: (i) application metrics, namely related to the vertical service operations; and (ii) infrastructure metrics, namely contextual information from both the virtualization infrastructures and VNFs. As such, it implements an internet-scale data-centric protocol that unifies data-sharing

between any kind of device including those constrained with respect to the node resources, such as computational resources and power, as well as the network resources. The DASS comprises three main elements, as depicted in Figure 2-5:

1. **Data Dispatcher:** gathers data from the different sources and delivers it to all the subscribed entities, enforcing authorization mechanisms if required, the data dispatcher can send raw data to be pre-processed or directly stored in the data storage element.
2. **Data pre-processing:** transforms raw data into an understandable and common format (e.g., data encoding/decoding, transcoding, querying, filtering, grouping, normalization, anonymization, and compression), the resulting data can be directly stored in the data storage element.
3. **Data storage:** stores gathered data and processed data. The decision of whether to store the data is up to its source which, optionally, also provides its lifespan. The data storage can store raw data coming from the data dispatcher or processed data from the pre-processing element.

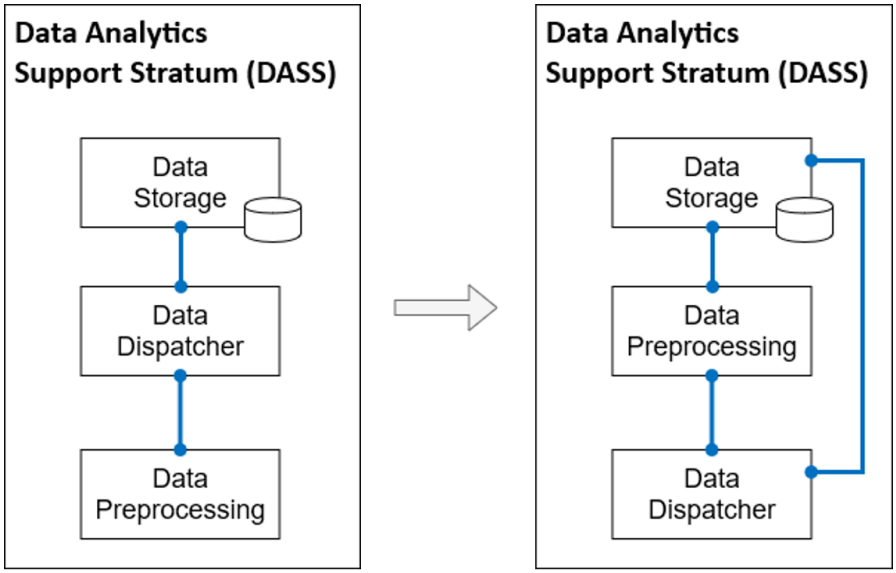


FIGURE 2-5: DASS REFINEMENT FROM ITS ORIGINAL DESIGN

The following refinement were implemented:

- Throughout the implementation of the DASS, it arose the need for the DASS to store the preprocessed data. Thus, the original design was updated to include a direct relationship between the data pre-processing element and the data storage element of the DASS. This resulted in the ability to store and support multiple data encoding, such as JSON, properties, relational, raw bytes, data streams, timeseries, etc. These diverse data formats may come from different data sources through the data dispatcher element.

Implementation details of the DASS can be found in Section 2.2.1 of D2.3 [5].

2.2.2. Intelligence Engine Support Stratum (IESS)

Envisioned as an AI/ML engine, the IESS supports, simplifies and automates the integration of AI/ML features for the vertical services themselves and/or for business automation tasks. It leverages data-driven AI/ML models to support complex decision-making systems, perform classification, forecast

demands, predict events, find patterns and anomalies, among others. To do so, the IESS follows an intent-driven approach, offering functions to interact with several AutoML and AutoAI frameworks, integrating a semi-automated selection of the appropriate framework based on context information. The IESS comprises four main elements, as depicted in Figure 2-6:

1. **AI/ML Manager:** coordinates all the requests for AI/ML capabilities, manages the available AI/ML models contained in the IESS catalogue, and packages the trained AI/ML models.
2. **AI/ML Model Catalogue:** consists of a repository of trained and untrained AI/ML models made available by the DEEP platform. These are annotated with metadata describing their features, requirements, and suitable applications.
3. **AI/ML Model Training:** handles the procedures for training an AI/ML model, from the selection of one or more AI/ML algorithms, their hyperparameters and training dataset, up to the cross-validation of their accuracy.
4. **AI/ML Execution Environment:** comprises the runtime environment to train the selected AI/ML model(s).

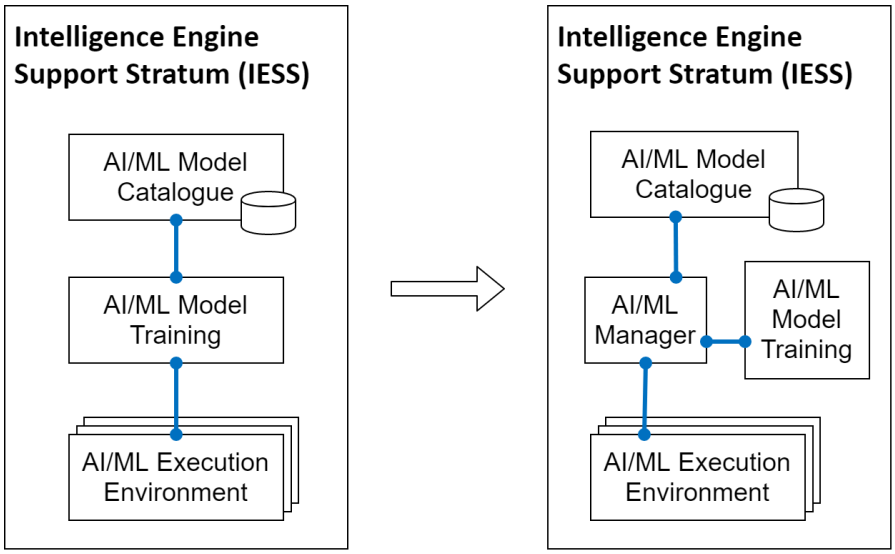


FIGURE 2-6: IESS REFINEMENT FROM ITS ORIGINAL DESIGN

Thus, the IESS manages the lifecycle of the model training procedure, by deploying the required resources, executing the training algorithms, and collecting the results for the cross-validation of the trained AI/ML models. Furthermore, the IESS provides functionalities to package the trained models into inference applications meant to be deployed alongside the vertical service and serving prediction results. Finally, the IESS includes a catalogue containing AI/ML algorithms, pre-trained models, runtime environments in order to enable the re-use of these artifacts and their sharing.

Throughout the implementation of this supporting stratum in WP2, it arose the need for the IESS to better decouple and modularize its functionalities. Figure 2-6 shows graphically the evolution of the IESS design. In the original version, depicted on the left part of the figures, the *AI/ML Model Training* component was in charge of managing the training procedures and of all the interactions to store and retrieve information from the *AI/ML Model Catalogue*. Furthermore, it also coordinated and requested the creation of *AI/ML Execution Environments*. The *AI/ML Model Training* was clearly overloaded with

heterogeneous functionalities and its implementation resulted difficult, with too broad boundaries. In the updated version of the design, depicted on the right part of Figure 2-6, the management functions have been decoupled from the *AI/ML Model Training* introducing the *AI/ML Manager*, that is in charge of exposing all the functionalities of the IESS as well as coordinating them. In the meantime, the *AI/ML Model Training* and the *AI/ML Execution Environments* have been better defined, with more clear boundaries in terms of functionalities and tasks. The changes greatly simplified the development and the internal workflows of the IESS.

Implementation details of the IESS can be found in Section 2.2.3 of D2.3 [5].

2.2.3. Business Automation Support Stratum (BASS)

The BASS provides the interface to plug OSS/BSS systems into the DEEP platform and it includes several components for the translation of business requirements into technical requirements. Internally, the BASS is composed of several components, each one with a dedicated task. The Vertical Service Coordinator takes care of deploying vertical services and managing their lifecycle. It directly interacts with the Edge Computing infrastructure and takes control of decisions with the support of the other components. The BASS comprises five main elements, as depicted in Figure 2-7:

- **Vertical Service Coordinator:** handles vertical-oriented requests, translates them into network services, and coordinates their E2E deployment. It can be enhanced with intelligence capabilities.
- **Vertical Service Blueprint Catalogue:** consists of a repository containing vertical service blueprints, providing a vertical service template with service-specific parameters.
- **SLA & Policy Management:** monitors the SLAs and policies for different vertical services and, in case of a violation, triggers corrective actions. It can be enhanced with intelligence and forecasting capabilities so that violations are predicted and dealt with preemptively.
- **Active Monitoring Manager:** manages the deployment of monitoring probes required for performance monitoring, including a catalogue of available probes.
- **External Federation:** handles the deployment of E2E vertical services across different administrative domains.

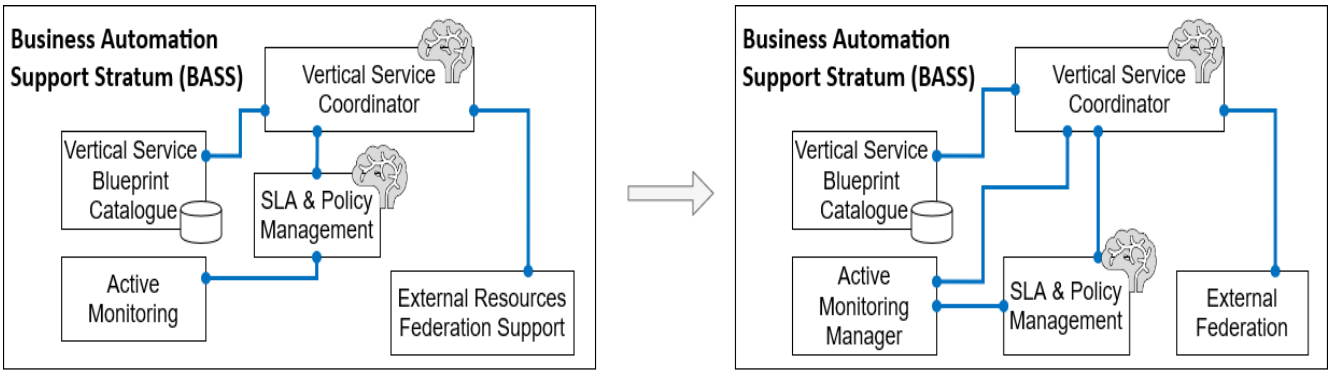


FIGURE 2-7: BASS REFINEMENT FROM ITS ORIGINAL DESIGN

Throughout the implementation of this supporting stratum in WP2 and the continuous analysis of its set of functionalities, it arose the need to refine the design of several elements and their inter-

connections, as depicted in Figure 2-7. These are mostly related to the SLA and monitoring features, which have been made independent in their relationship with the *Vertical Service Coordinator*. In particular, the following refinements were implemented:

- The *Active Monitoring* has been renamed to *Active Monitoring Manager* to highlight its role in the control of all the monitoring operations. The module offers a set of monitoring probes that, once activated by the *Vertical Service Coordinator*, collect metrics about the desired vertical service. Nevertheless, this component is not a mere catalogue. It also manages and tracks the monitoring probes with full control on their lifecycle. Finally, it provides "pointers" to the probes in order for other components to retrieve data via DASS (data plane).
- A direct interface has been added between the *Vertical Service Coordinator* and the *Active Monitoring Manager*. In the original design, the interactions with the *Active Monitoring* module were mediated by the *SLA & Policy Management* module, since the latter was the main beneficiary of the monitoring services. Anyway, during the implementation, the monitoring services gained importance as an independent feature of the BASS, in order to check the status of a service.
- The *SLA & Policy Management* makes use of the monitoring services and is given a direct interface to the *Active Monitoring Manager*. The latter provides information about monitoring probes and provides pointers to the data streams, in order to retrieve them from the DASS.
- The *SLA & Policy Management* has been given a direct interface to the IESS (see Figure 2-4 for this change). Indeed, SLA enforcement engines may feature artificial intelligence and machine learning. The *SLA & Policy Management* can now directly request the IESS for AI/ML services without the mediation of the *Vertical Service Coordinator*. This simplifies the workflows since the request for SLA enforcement engines is a special case with respect to regular IESS requests coming from vertical services.

Implementation details of the BASS can be found in Section 2.2.2 of D2.3 [5].

2.3. Workflows

DEEP platform consists of various workflows to cover different demands for verticals. The workflows mainly cover four parts, representing the basic operations offered by the DEEP platform.

- **Vertical service lifecycle management:** this workflow provides request, creation and deployment of a service, termination of service and update (e.g., scaling, migration, etc) of a service.
- **Training AI/ML applications:** this workflow provides data spanning multiple training, algorithms, and AI/ML components to enhance the services of the applications.
- **Data gathering:** this workflow collects input data and dispatches it to the IESS for training purposes and/or to the BASS to monitor the performance of running vertical services and to validate the fulfilment of SLAs.
- **Federation support:** this workflow support leasing resources or services from third-party domains.

All the aforementioned workflows are elaborated in the following subsections. Other workflows can be envisioned, that might depend on the vertical service integration and its business automation requirements.

2.3.1. Vertical Service Lifecycle Management

Verticals are able to deploy their vertical services by interfacing with the DEEP platform. For instance, a public safety agency interacts with the DEEP platform to create and deploy the required autonomous drone service, through vertical-oriented interfaces that reflect its business needs. Figure 2-8 depicts the adopted workflow for a vertical service instantiation, triggered with every incoming request from the vertical, which is described as follows:

1. The vertical fetches the Vertical Service Blueprints from the *Vertical Service Blueprints Catalogue* element of the BASS. This catalogue covers different service demands for applications and networks utilized by the vertical, such as a drone service or a digital twin service.
2. The *Vertical Service Coordinator* element of the BASS receives the specific vertical service request from the vertical, described in the form of a Vertical Service Descriptor. Obviously, the BASS is capable of addressing the requests as detailed in Section 2.2.3.
3. (optional) the *Vertical Service Coordinator* configures the required mechanisms in the *SLA & Policy Management* to verify the policies and ensure the SLA associated with the vertical service.
4. (optional) The *SLA & Policy Management* and/or the *Vertical Service Coordinator* request monitoring probes to the *Active Monitoring Manager*, if required. An example of a monitoring probe for the drone vertical service consists of monitoring the drone battery status, triggering an alarm when it is less than a certain threshold.
5. The *Vertical Service Coordinator* maps and translates the Vertical Service Descriptor into one or more Network Services Descriptors. In particular, the vertical service requirements defined in the Vertical Service Descriptor will be translated to specific network services.
6. The *Vertical Service Coordinator* requests the deployment and instantiation of each network service from the infrastructure to guarantee the services for the verticals.

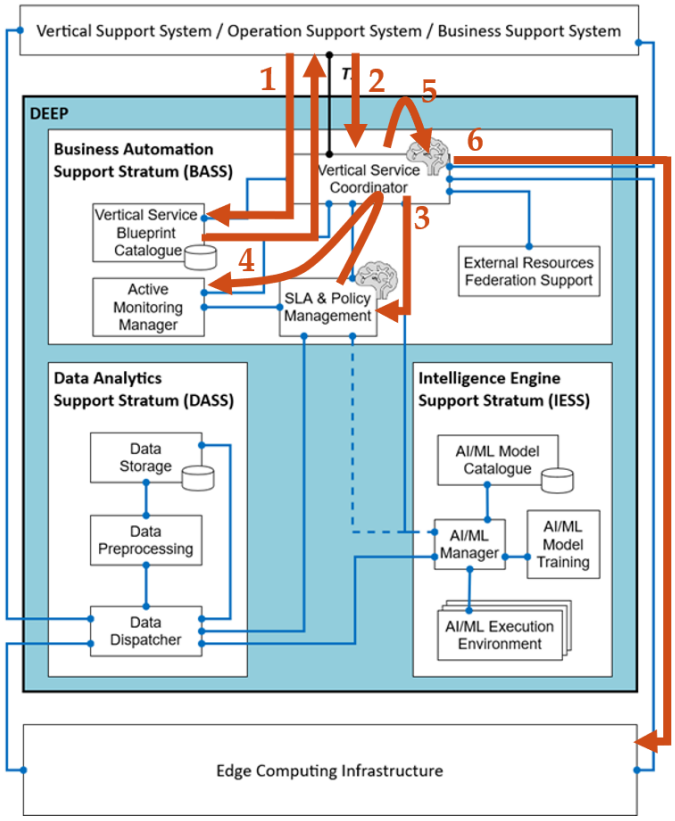


FIGURE 2-8: VERTICAL SERVICE REQUEST WORKFLOW

2.3.2. Training of AI/ML Applications

In the vertical industry, adoption of the edge AI/ML solutions is necessary to accommodate the increasing workloads on the cloud, rapid growth in the number of intelligent applications, and many other enhancing services. In particular, AI and ML are promising solutions to transit towards an intelligent and automated operation [7][8]. However, such solutions will need to conjugate the variety of data spanning across multiple technologies domains, algorithms, and components. The DEEP platform, through the IESS, is able to facilitate such tasks by training AI/ML models and packaging them as inference applications that can be used to support the operations of the vertical service. The workflow for training AI/ML models is shown in Figure 2-9 with the following steps:

1. The BASS receives a vertical service request containing AI-based intent parameters, including pointers for training data.
2. The *Vertical Service Coordinator* requests the IESS for an AI/ML application that fulfils the desired purpose:
    a. The *AI/ML Manager* selects one or more AI/ML models that are suitable for the task.
    b. The *AI/ML Manager* subscribes to relevant data from the DASS.
    c. The *AI/ML Model Training* trains the selected model(s) until accuracy is above a threshold.

3. The *AI/ML Manager* packages the AI/ML model as an inference application, returning it to the BASS. This will allow the vertical service to retrieve inference results and to reutiliz the trained model for other new applications adopted in the vertical.

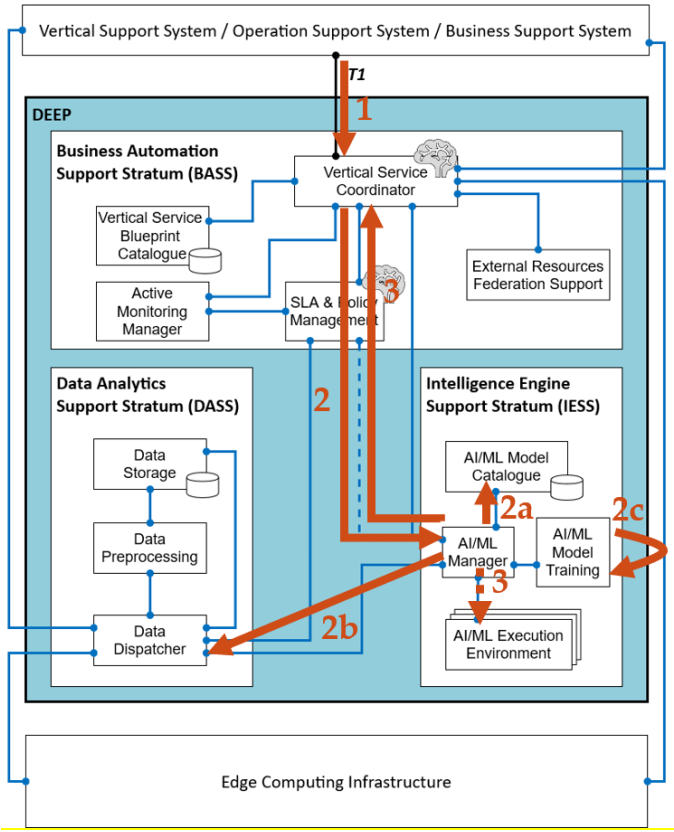


FIGURE 2-9: TRAINING AI/ML APPLICATIONS WORKFLOW

2.3.3. Data Gathering

The data gathering is an essential step for the verticals to monitor the operational performance of their vertical services in order to collect input data for the training and cross-validation of AI/ML models, and in order to use such data as input for business automation tasks within the DEEP platform. Data gathering workflow is shown in Figure 2-10, which steps are described as follows:

1. The DASS acts as the enabler for gathering data from the different sources, whether it comes from the vertical itself or the underlying Edge Computing Infrastructure:
    - Publish/Subscribe, Queries and/or Push-based communication models are possible to be used.
2. (optional) The DASS applies pre-processing mechanisms to the data, such as data cleaning, data filtering, data grouping, data normalization, data anonymization, data transformation, data compression, among others.
3. (optional) The DASS stores the data if flagged by the publisher, and only during its lifespan (if any)

4. The DASS dispatches data to all interested parties, either IESS and BASS elements.

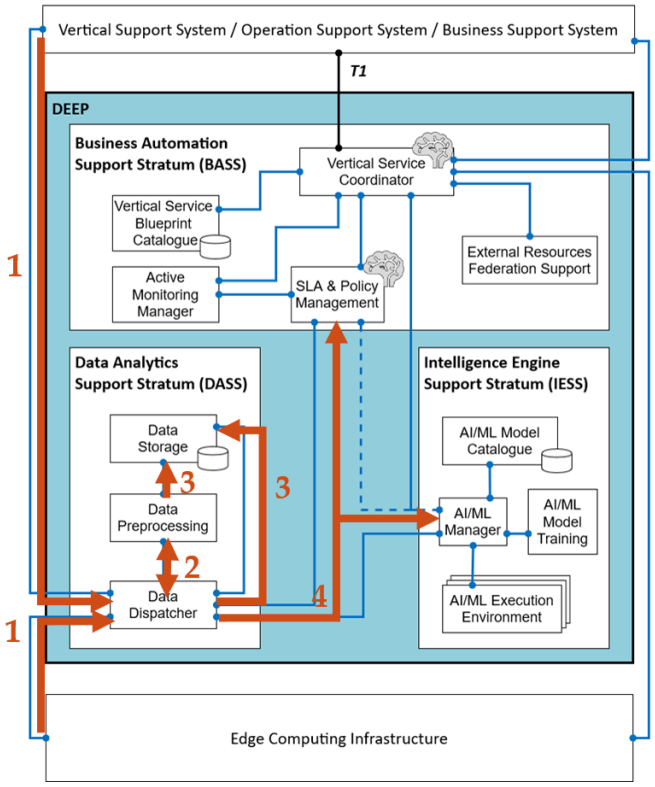


FIGURE 2-10: DATA GATHERING WORKFLOW

2.3.4. Federation Support

It is important to consider leasing resources or services from third-party domains that are not available on the PN (MNO) and NPN (vertical). This requires replacing static agreements with dynamic and on-demand agreements. In order to provide flexible support of such scenarios, Figure 2-11 depicts the internal workflow within the DEEP platform, which steps are described next.

1. The BASS receives a vertical service request described by its Vertical Service Descriptor.
    - The Vertical Service Descriptor can explicitly identify which part of the service should be federated or, alternatively, the BASS can detect this need taking into consideration the available resources, costs, and requirements.
2. The BASS executes the steps on the Vertical Service Request workflow mentioned earlier.
3. The *Federation Support* maps the descriptor parameters to a set of resources to be requested towards federated domains, including the required information for the stitching of the network services.
4. The BASS requests their deployment and instantiation, either locally or federated.

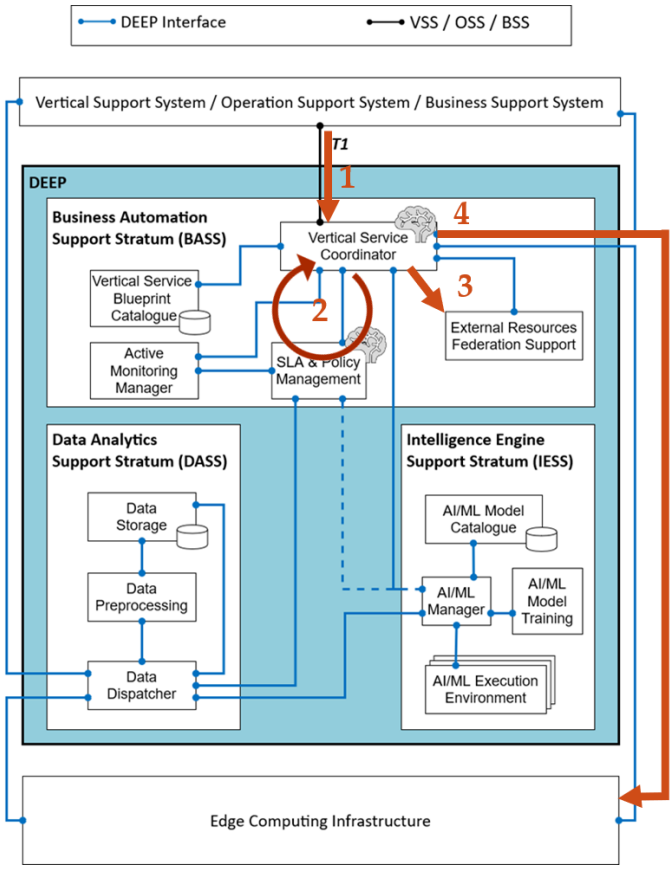


FIGURE 2-11: FEDERATION SUPPORT WORKFLOW

3. DEEP Innovations

The DEEP platform goal is two-fold: *(i)* hide the underlying complexity to the vertical; and *(ii)* automate the provisioning and lifecycle management of vertical services. In doing so, the DEEP platform acts on behalf of the vertical to enforce automation of its business processes, allowing the vertical to focus on its knowledge domain. These features are embodied in a set of innovations, as described in the remaining of this section.

3.1. Vertical Service Abstraction

The DEEP platform abstracts the technical details of the vertical service definition and management, in order to switch the focus of its users and stakeholder on business aspects.

In Section 3.1.1, we define the user roles acting on the platform and their characteristics. While some users retain a technical role in order to enrich the catalogue of services and features offered by the platform, the design focuses on technical users. In Section 3.2.2 we present the main abstraction tools designed for the DEEP platform.

3.1.1. Role Based Access Control

The users interacting with the BASS can be assigned one or more roles, determining the resources and operations they have access to. There are three main roles in the BASS:

- Vertical Developer
- Vertical Operator
- Region Operator

The role of Vertical Developer is reserved for users that have strong technical skills and deep knowledge of how a vertical service works. They are able to design and prepare a vertical service for deployment on the DEEP platform. On the other side, the Vertical Operator role is reserved for users that have less technical skills but still some knowledge about the service configuration, management and lifecycle. They are competent enough to customize the deployment of a service by modifying some parts of it, supplying additional information that affects the business aspects. Finally, users with the role of Region Operator can perform administrative tasks in order to manage and control the resources the BASS can operate on.

Each role has access to a limited set of data elements on the BASS. The Vertical Developer has full access to the Vertical Service Blueprint documents (see Section 3.1.2). The Vertical Service Operator has read-only access to blueprints while having full access to the creation of Vertical Service Descriptors and Vertical Service Instances (see again Section 3.1.2). Finally, the Vertical Region Operator has full access to regions, representing the computing, networking, and storage resources the BASS relies on to deploy the vertical services. The other two roles have read-only access to regions.

A user of the BASS can be associated with more than one role at the same time, giving him the ability to operate more aspects of the system. Anyway, the role assignments should always be minimal per each user and tailored to their skills and responsibilities.

3.1.2. Vertical Service Descriptor and Vertical Service Blueprint

Vertical services can be described on the BASS with the Vertical Service Descriptor (VSD), a document with a structured data format that is at the same time relatively easy to write for a human and to be processed by a machine. It is relatively easy to write and understand by a human user since it is declarative and it describes a goal or desired state to be met by the infrastructure without the need to specify the actions to reach that goal. The BASS takes care of translating the blueprint into actions for the underlying distributed Edge infrastructure in order to create a vertical service instance, made of virtual resources like containers, virtual machines, networks, etc. Furthermore, the BASS can deploy a VSD on several resource orchestrators, effectively creating an abstraction layer between the vertical user and the underlying infrastructure. This allows the definition of services with a common format with the possibility to deploy them on several infrastructures, each one deployed with a specific technology.

While the VSD provides the clear advantages of declarativity and abstraction for the definition of a service, it has the problem of being a rigid model, since it describes a single instance of the vertical service. In order to solve this, we introduce the Vertical Service Blueprint (VSB) is a template of a vertical

service that can be partially customized. The goal of the introduction of the VSB model is twofold: to make the Vertical Service Descriptor (VSD) more flexible and to abstract this flexibility through parameters that are simple and safe to modify. The VSB provides a more generic definition of the vertical service with respect to the VSD in order to separate as much as possible the functional definition of the service from the configuration needed to run it on a specific environment. The deployment-specific configuration is separated and encoded in the form of a list of parameters. From a VSB, multiple versions of the same service can be created depending on the values set for the parameters. Figure 3-1 shows the conceptual process of designing, configuring and deploying a service on the DEEP platform.

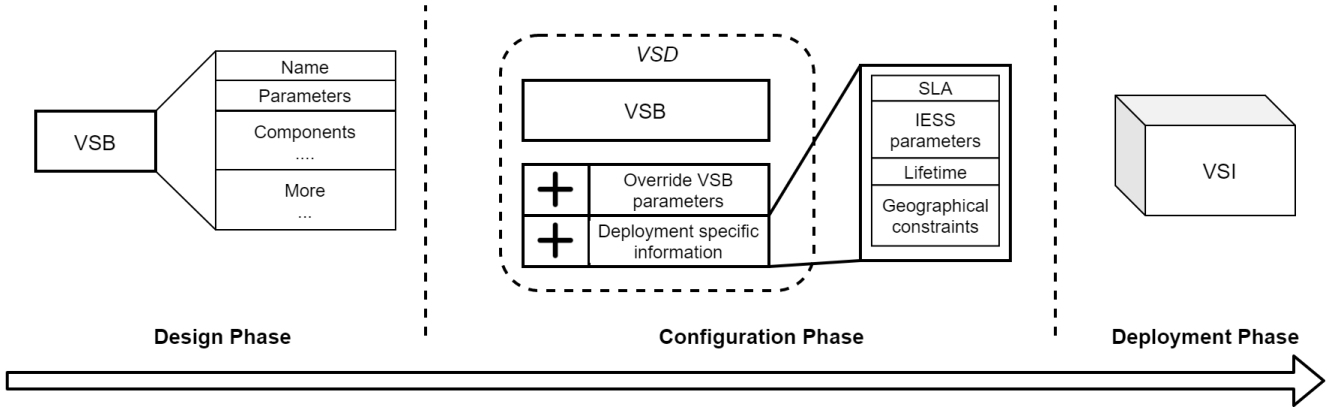


FIGURE 3-1: THE LOGICAL PHASES OF A VERTICAL SERVICE IN THE BASS

On the left side of the picture, we can see the design phase of the service. The VSB defines the main components of the service, their dependencies, their requirements, and any other information useful to describe its functionality. During the configuration phase, shown in the centre of Figure 3-1, the service definition is completed with the override of one or more of the VSB parameters and also augmented with additional information to adapt the service to the requirements and constraints of the specific deployment.

The additional information includes:

- SLA and quality of service
- Parameters for AI/ML requests to the IESS
- Lifetime
- Geographical constraints

The combination of the service definition provided by the VSB, the parameters override, and the additional information generates a customized VSD ready to be deployed by the DEEP platform. The result of the deployment is the creation of a vertical service instance (VSI), as shown on the rightmost part of the figure above. Potentially, multiple instances can be created from the same VSD achieving an additional level of flexibility.

The introduction of the VSB in the BASS introduces several advantages with respect of using only the VSD. First, it realizes a separation between the functional description of the service and its qualitative characteristics from a business-oriented point of view. As discussed in Section 3.1.1, the Vertical Operator role is reserved for those users with limited technical skills but interested in deploying and managing several instances of a vertical service. Thanks to the VSB model, a user with the role of Vertical

Operator can create multiple versions of the same service without knowing its internal details. For example, given a VSB for a video streaming service, it is possible to create two versions of it: service A that streams videos at a quality of 720p and service B, that streams videos at 1080p. The service functionality (i.e., streaming a video) as well as its components and interactions stay the same, but the two flavours present a distinct behaviour in terms of the quality of the transmitted video stream. To achieve the scenario just described, a Vertical Operator would just need to set the proper value of a parameter (e.g., video quality) without the need to rewrite the whole VSD for each service version.

The VSB also improves the portability of the service. The parameters allow adapting the service to different infrastructures meeting their requirements and constraints. Likewise, the flexibility is improved, since, as shown in the previous examples, the parameters can also be used by the user to tune the service for their needs.

The VSB finally improves the shareability of services and the collaboration between developers. Indeed, the more generic description of the service in the VSB avoids the disclosure of sensitive information (e.g., infrastructure network configuration and topology, secrets, user passwords, tokens). The design of a new service can start by taking as an example and inspiration a VSB created for a similar service and solutions to common problems can be openly discussed and improved through collaboration. In order to enable this, the BASS includes a catalogue of VSB documents. Stored blueprints are not public by default but they can be shared upon request of their owner.

In the BASS, users with the role of Vertical Developer can create, update, and delete VSBs, while users with the role of Vertical Operator can only access them read-only. Anyway, after selecting a VSB from the catalogue, the Vertical Operator can create a VSD from it, as shown by the workflow in Figure 3-2.

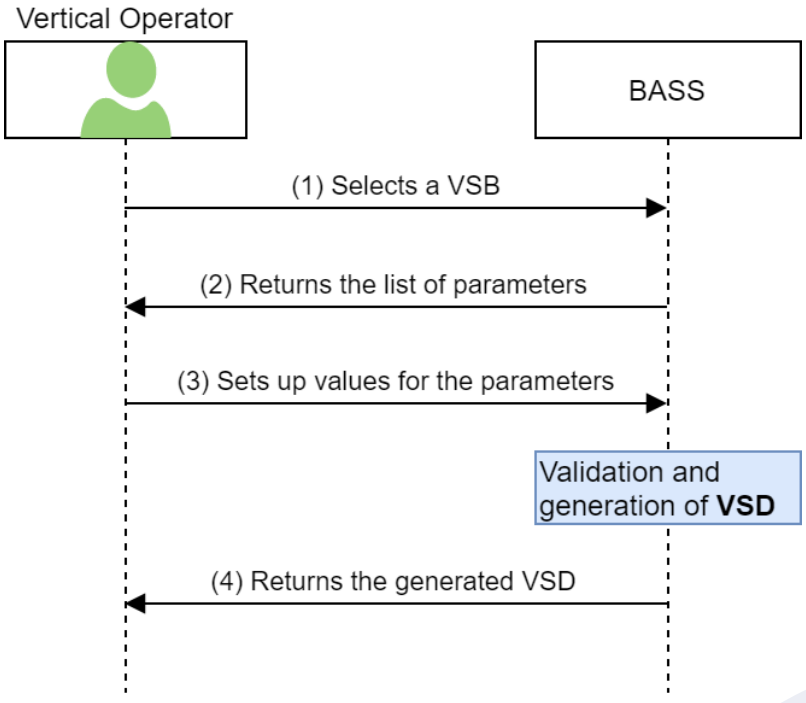


FIGURE 3-2: GENERATION OF A VSD FROM A VSB

At step 1 the Vertical Operator selects a VSB from the catalogue. The BASS returns the list of parameters for that VSB, to be filled with the desired values (step 2). At step 3, the Vertical Operator fills the parameters and submits the new values. The BASS validates the parameter values and generates a new VSD. Finally, the newly generated VSD is returned to the Vertical Operator (step 4).

The concepts of Vertical Service Descriptor and Vertical Service Blueprint have been introduced in other EU projects with partially similar goals. 5G-TRANSFORMER [9] introduced the concept of blueprint in order to define vertical services in a simpler way with respect to the Network Service Descriptor and Virtual Network Function Descriptor data models proposed by ETSI [10][11]. Anyway, to enable the deployment, it is required to provide the NSD and VNFD documents alongside the blueprint. 5G EVE [12] improves the solution from the usability point of view by introducing a software module dedicated to the automatic translation from the blueprint to ETSI formats and to the composition of blueprints in order to enable the modularization and reusability of network functions and services. Anyway, the blueprint concept proposed by the just mentioned projects is focused on supporting the experimentation of vertical services on 5G enabled testbeds and does not provide the flexibility necessary to fit production-oriented environments. Furthermore, their solution requires a NFV enabled infrastructure and the presence of at least an NFV orchestrator, an architectural model different from the DEEP architecture design. At the same time, the tight coupling of the 5G-TRANSFORMER solution and 5G EVE solution to the ETSI NFV data models introduces several complexities: first, the ETSI data models are not fully implemented by any NFV orchestrator requiring an additional step of model translation, second, the ETSI data models, while being extremely flexible and powerful, are also very complicated. Finally, our goal to hide technical complexity from the Vertical Operator by presenting him a simpler interface, more oriented to a business language, is not met by the aforementioned solutions, that still include some technical details in the blueprints managed by the vertical user.

3.2. Active Monitoring based on Probes

Although monitoring physical and virtual resource consumption provides useful hints on the performance of the vertical services, by itself this information is not enough to fully validate their performance. For example, by just relying on infrastructure metrics, like the CPU and MEM consumption or the latency in a given link, it is not possible to assess if the service KPIs of a given vertical service are being fulfilled. To do so, monitoring tools must be able to track application-specific metrics, which reflect the real performance of the vertical services. Ultimately, by analysing both application-specific and infrastructure metrics, monitoring tools provide the means to better raise alerts, predict performance issues, and audit if the SLAs are being guaranteed.

In this sense, the DEEP platform goes beyond already available physical and virtual resource monitoring solutions and integrates application-level metrics, even when applications do not explicitly expose such information. This is especially relevant when applications are provided by third-party stakeholders as black-boxes with limited configuration capabilities, and are not involved in the vertical service creation, integration, and deployment stages. To workaround such scenarios, the DEEP platform envisions and supports the use of monitoring probes that can be used to extract meaningful performance data from

such applications, empowering the automation and AI/ML procedures with heterogenous information from different vertical services and technological domains.

Since the envisioned monitoring probes must be aware of the vertical service specificities, they are implemented by the Vertical Developer (as defined in Section 3.1.1) during the preparation of the vertical service (i.e., while designing the VSBs). Each monitoring probe defines the method to get a given application-specific metric, the logic to expose the collected monitoring data to the DEEP platform (via DASS-powered mechanisms), and the relevance to distinct service KPIs. The methods can be implemented following different approaches:

- **Log analysis:** most of the applications expose different levels of logging capabilities that are mainly used for debugging purposes whenever an issue is detected. As such, logs usually provide all types of operations executed by the applications with a timestamp associated. This kind of probes gather logs generated by the applications, parsing and extracting the data to compute specific service KPIs.
- **Traffic inspection:** when distributing a vertical service across several computational resources in the cloud-to things continuum, the network is paramount to enable communication between all its components. As such, inspecting the network messages exchanged between the vertical service components can be used to extract meaningful information about the performance of applications and the vertical service as a whole. These probes can be deployed either on the "client", the "server" or anywhere in the network, although the last option might be limited by any security layer.

As examples of their applications, monitoring probes can be used to: *(i)* resource management and failures/anomalies detection; *(ii)* identification of performance or configuration issues; *(iii)* assessing application health; *(iv)* diagnosis and identification of the root cause of errors; and *(v)* detect application performance bottlenecks and optimize processes.

Upon the definition of the monitoring probes, the Vertical Developer makes them available in the BASS *Active Monitoring Manager* catalogue. Then, during the vertical service deployment stage, the BASS can decide on the deployment of additional monitoring probes aiming for a three-fold purpose: *(i)* assess the defined SLAs by the Vertical Operator; *(ii)* support any IESS requirement for the training and/or execution of intelligence engines; and *(iii)* provide additional insights to the Vertical Operator.

3.3. Business Automation

The DEEP platform aims at translating business requirements into technical requirements for the underlying infrastructure and while performing this task, it attempts to automate some business procedures traditionally carried on through human interactions.

The negotiation of an SLA between a customer and a provider can be a long and tiring process between the two parties. Furthermore, it requires to adhere to principles and best practices proposed by consortiums and SDOs, like [13], in order not to incur trivial and well-known mistakes. In Section 3.2.1,

we explore several approaches based on the 5G-DIVE stakeholders in order to define structured negotiation procedures.

For similar reasons, changing an agreed SLA while it is in place requires human interaction and hence a lot of time. The current network and data services have highly dynamic constraints and need to be updated and reconfigured quickly. A provider should be able to propose new SLAs to its consumers at a similar rate of speed. In Section 3.2.2 we define an SLA model that is suitable to be implemented and managed by machines and in Section 3.2.3 we introduce the concept of SLA enforcement to adapt the changing service requirements to the provider resources.

3.3.1. Stakeholders SLAs

Novel services and applications are no longer based on a single isolated infrastructure. Instead, they are composed of resources from various providers that operate between them to serve the vertical. Such interoperability is a big challenge, and this is also the case with 5G-DIVE.

In this regard, the interdependencies between application, 5G communications and Cloud & Edge need to be considered during the service design, planning, deployment, provisioning, and maintenance, leading to a particular procedure for Service Level Agreement Management (SLAM).

One possible approach could be the one proposed in [14], which tries to create "the best" SLA given a set of SLOs and a set of resource and service configuration possibilities. In 5G-DIVE, the implementation of strategies for establishing a SLA or agreeing on an SLA is the responsibility of the *System & Service Provider* who operates the DEEP platform. For such a purpose, three different components can be identified:

- **Service Level Objective (SLO)**: is a message from the vertical soliciting the type and level of service to be provided.
- **Service Level Agreement Template (SLAT)**: is a statement or advertisement from providers about their guaranteed service levels. In 5G-DIVE we distinguish application providers, 5G communication providers and Cloud & Edge providers.
- **Service Level Agreement (SLA)**: is an agreement that the vertical is prepared to accept. When the *system & service provider* also accepts the contract, it is represented as SLA*, which is equivalent to a signed SLA.

According to Figure 3-3, it involves different stakeholders to make the service provision possible.

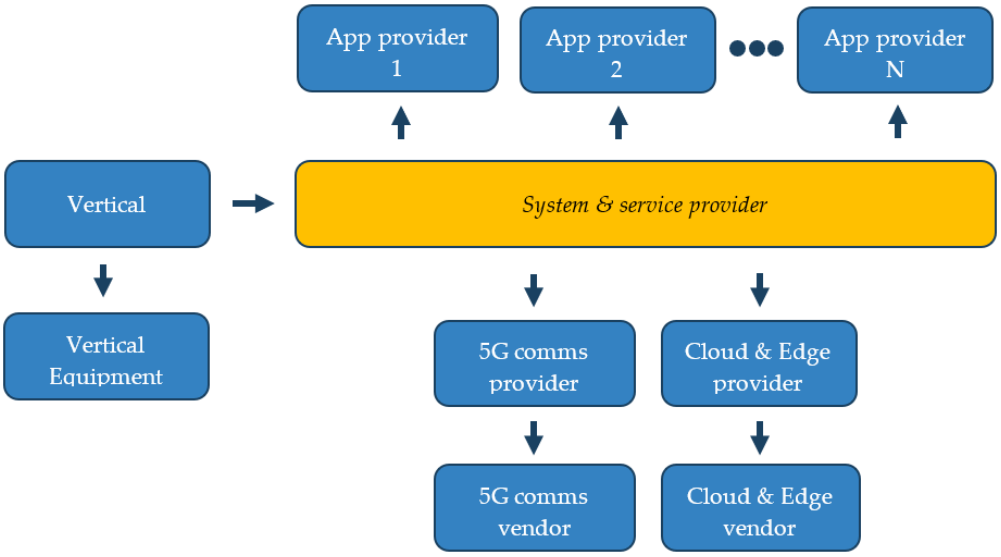


FIGURE 3-3: STAKEHOLDERS' FLOW

From all of them, we focus on those that are relevant for the SLAs composition:

1. *Vertical*: In the case of the Industry 4.0 pilot this stakeholder corresponds to the factory owner that provides the service to the end- user.
2. *System & service provider*: This stakeholder represents the role of a network operator that not only provides network capabilities but also a range of different services such as connectivity, value-added applications and edge computing that result in selling a vertical complete system.
3. *Application provider*: It provides the applications to the vertical, being able to provide a service (e.g., digital twin, ZDM, mMTC, drone location, drone fleet coordination).
4. *5G communication provider:* This stakeholder provides the entire 5G infrastructure, including the 5G Core and 5G-NR coverage at the vertical premises.
5. *Cloud & edge provider*: It provides the virtualization infrastructure required to allow the provision of applications and 5G services. It involves both networking and computing resources.

Depending on the interaction that the previous stakeholders have with the *System & Service provider* we can have two possible approaches:

- **Request approach (Figure 3-4):** In this approach, the vertical initiates the SLA interchange process. The factory owner sends an SLAT(SLO) to the *System & Service Provider* (the stakeholder operating the DEEP platform) who splits the SLO into those that apply for each type of provider (i.e., SLAT(SLO_{app}), SLAT(SLO_{5G}) and SLAT(SLO_{cloud})) and distributes them to the Application, 5G communications and Cloud & Edge providers. Afterward, the *system & service provider* aggregates the signed SLAs sending the final SLAT(SLO*) back to the factory owner.
- **Catalogue approach (Figure 3-5):** This approach supposes the interchange of the SLAT(SLO_{app}), SLAT(SLO_{5G}) and SLAT(SLO_{cloud}) before the factory owner solicits any service from the system & service provider. As a result, when the factory owner sends it SLAT(SLO) the system & service provider can directly signed and send it back saving the processing time.

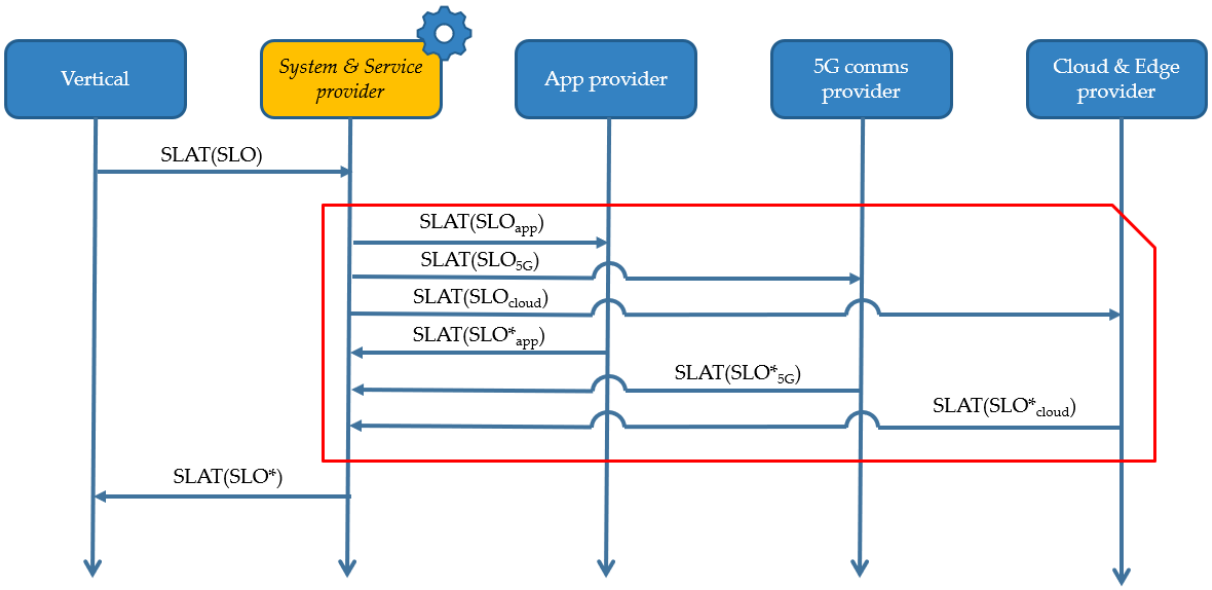


FIGURE 3-4: REQUEST APPROACH IN SLA MANAGEMENT

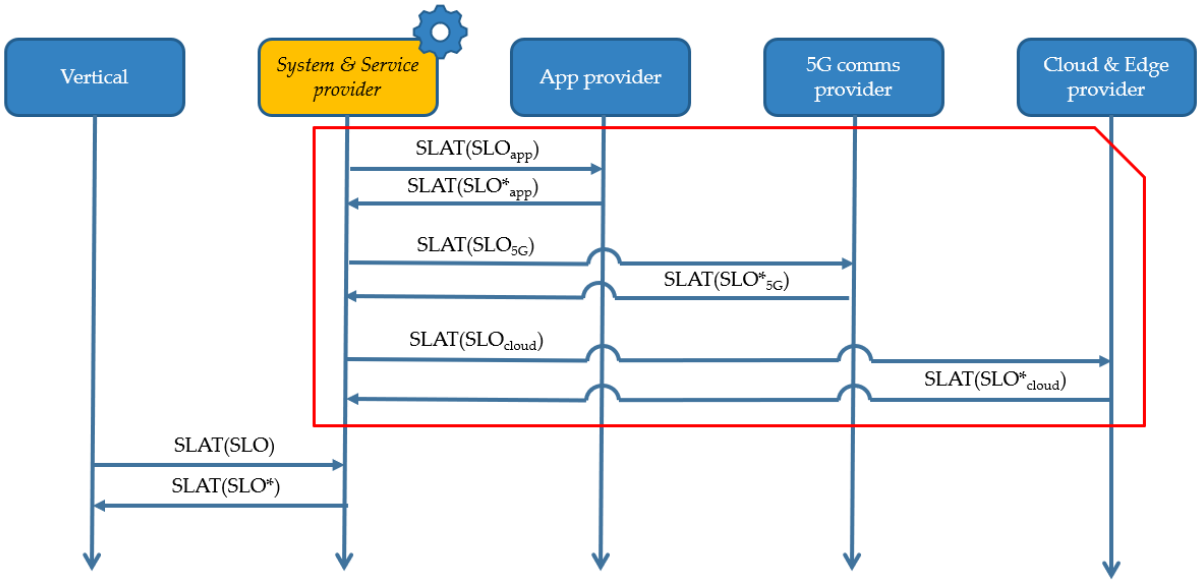


FIGURE 3-5: CATALOG APPROACH IN SLA MANAGEMENT FOR INDUSTRY 4.0 PILOT

Both the request approach and the catalogue are interesting for novel services as those proposed in 5G-DIVE. The request approach serves best stakeholders' flows which count on *system & service providers* that vary their capabilities frequently along the time or do not want to make them publicly available. The catalogue approach serves best stakeholders' flows which need a quick and reliable interchange of information. Due to the intrinsic characteristics of the 5G-DIVE solution, the latter approach seems to be the most convenient as it allows better integration with the BASS platform.

3.3.2. SLA Model

In order to encode the SLA concepts into the BASS, we introduce the following entities, as defined in [15]:

- **Service Level Indicator (SLI):** a measurement of the service quality. It can be seen as a KPI.
- **Service Level Objective (SLO):** an objective the system must meet. It is defined on top of an SLI.
- **Service Level Agreement (SLA):** an agreement on the quality of service defined as a list of SLOs.

The structuring and decomposition of the SLA into many atomic objectives, the SLOs, enables great flexibility in the definition of the quality of service. Each VSB includes a set of SLOs, each one defining a target for an SLI. In the procedure of generating a Vertical Service Descriptor from the blueprint, the Vertical Operator picks the SLOs that should suit better its service instance defining a custom SLA. Different instances of the same service can get a distinct, customized SLA, in order to fit the desired goal for that instance as well as the available cost budget. For example, a service instance for experimentation may not need high availability and reliability since its lifetime spans a short period and the service importance is not critical. A set of SLO with more relaxed targets can be defined in order to decrease the costs for the vertical operator.

As mentioned above, an SLO is defined concerning a SLI. A SLI can involve one metrics coming from the monitoring probes or multiple metrics, aggregated together through a mathematical function (e.g., average). In any case, a SLI should produce a series of quantitative values during the lifecycle of the vertical service. A SLI is also associated with corrective policies that, when applied, would affect the value of the SLA. Some corrective policies could be the vertical or horizontal scaling of a component of the vertical service, or the migration of a component across two regions.

The definition of SLI and SLO is tailored to the Vertical Service and they are included in the blueprint in order to perform a customizable selection for each deployment. Since the Vertical Developer is in charge of composing the Vertical Service Blueprints, and since they have a detailed knowledge of the vertical service, they are in charge of defining the set of SLI and SLO. The Vertical Operator, a user role with a more business-oriented view, is left with the task of simply selecting the desired SLOs for each specific deployment. The selection of the SLOs implies a cost on the infrastructure, that could be shown before initializing the deployment. The Vertical Operator can accept the proposed cost or reconfigure the SLO selection in order to better meet its performance requirements or budget limitations. This is a negotiation process between the user and the infrastructure, in order to dynamically establish the most appropriate SLA for each specific service instance. The negotiation process is inspired by the negotiated SLA management procedure discussed in [14] and [13].

3.3.3. SLA Enforcement

SLA Enforcer provides a framework to implement SLA enforcing applications for the services deployed using BASS. The framework includes the SLOs to enforce, the measured SLIs from the service extracted by using the Active Monitoring component of the BASS, stored in the DASS, and the set of allowed actions that can be triggered to correct an SLA violation. These three elements, when coordinated together, compose a “sense, think, act” closed-loop that will retrieve metrics or SLIs, apply logic depending on the target SLOs and choose the best corrective action that can be applied to the system to ensure the fulfilment of the SLAs.

In Figure 3-6, a simplified architecture of the SLA Enforcement is shown. The SLA Manager is the component in charge of configuring the input and output interfaces, according to the negotiated SLAs and the metrics that involve the SLIs and SLOs, and also the valid corrective actions that can be taken to ensure the proper fulfilment of them.

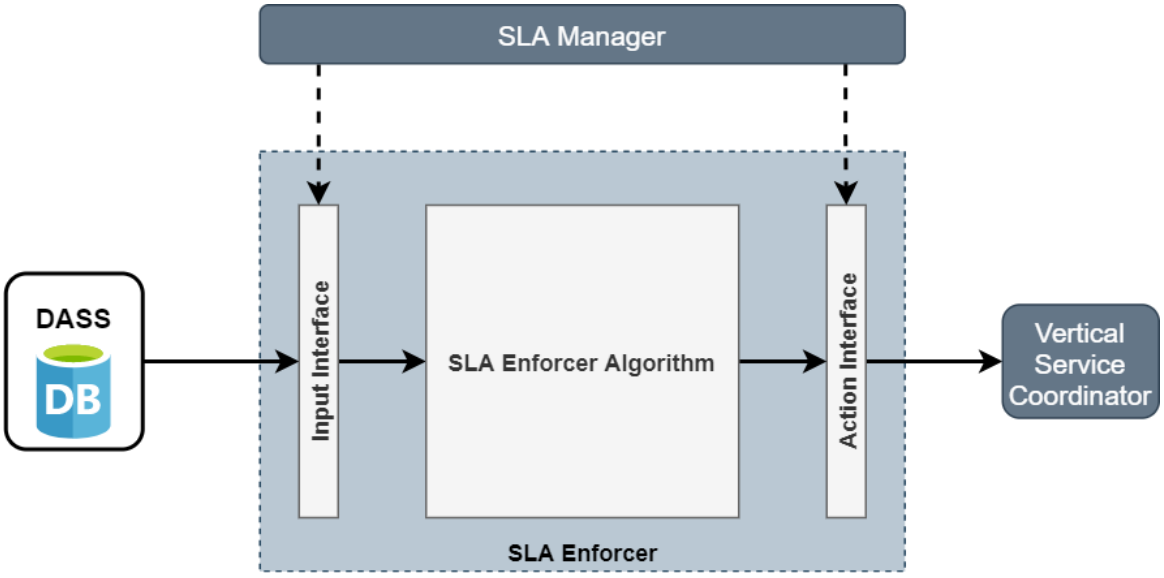


FIGURE 3-6: SLA ENFORCEMENT SIMPLIFIED ARCHITECTURE

The SLA enforcer framework we propose allows verticals to provide their custom algorithm to manage the SLA of their service. It is only required to adapt the engine to the interfaces exposed for reading SLI values, compare them with the SLOs, and perform the corrective actions.

This concept is very similar to what was already proposed in 5Growth [16], in the sense that it identifies an automation closed-loop for the detection and correction of SLA violations. Anyway, here we propose a generic framework, that can potentially support any kind of traditional or AI-based algorithm to implement the SLA enforcement logic, giving more flexibility and potential to develop more complex strategies and policies. Vertical and even third parties can plug their custom SLA enforcement engines into the DEEP platform.

On the other hand, the combination of the *Active Monitoring Manager* BASS component and the DASS constitute the counterpart of the 5Growth Monitoring Platform.

3.4. AI/ML-based Intelligence Support

The DEEP platform comprises tools that facilitate the definition of online/offline AI model training and cross-validation of AI/ML models. The IESS is able to perform automatic AI model training. Verticals do not need to worry about manually tweaking the training parameters and validate the training results, allowing verticals to save time to identify the optimum parameters. With the IESS being able to handle it automatically also means that the barrier for entry for vertical is lower. The vertical operator does not need to be an expert in working with AI/ML model before being able to use it.

Alternatively, rather than manually train and selects for a trained model, verticals can let the IESS to selects a suitable AI/ML automatically for a given intent. This automatic selection of IESS is based on an AutoAI-driven solution. As decision processes need to take into consideration a high number of variables, with complicated non-linear relationships between them, which are complex to manage and to program using traditional approaches. Additionally, they are prone to errors or non-optimal decisions when facing unexpected situations. Instead, AI/ML techniques can analyse complex and uncorrelated data to extract and understand its meaningful value, which when applied to decision processes lead to better and more optimal decisions.

Take the *Intelligent Image Processing for Drones* use case (ADS-UC2) as an example. In ADS-UC2, an AI model is used for the detection and localization of Person in need of Help (PiH). This process relies on the detection of 'flag', and 'person' objects. In this scenario, the IESS can be used to perform automatic AI model training for the detection of 'flag', and 'person'. With the tools provided by the DEEP platform, verticals can choose to provide pointers for a publicly available dataset or a custom dataset. Other than the dataset, verticals can also provide the desired level of performance metric for the trained AI/ML models. In the case of an object detection scenario like in the one for ADS Use Case 2, a request is put to the IESS to train for an AI object detection model that is able to achieve at least 95% Average Precision (AP) [17] for 'person' object, and at least 80% AP for 'flag' object. More details on the AI/ML training process under the DEEP platform can be found in Section 2.3.2. The trained AI/ML models from the IESS are then stored in a model catalogue. This catalogue serves as a repository for all of the trained models that various verticals used.

Another way to utilize the IESS is to train a model for automated vertical service management (e.g.: auto-scaling, self-healing, and pre-emptive migration). During the runtime of intelligent vertical applications on the DEEP platform, the behaviour of a vertical service application can be monitored. Using this behaviour history, the IESS can be utilized to automatically train for an AI/ML model whose objective is to perform vertical service management. As an example, during the lifetime of the ADS-UC2 vertical, we can collect data such as processing load, processing latency and resource utilization. With IESS, this data can be used to train for an AI/ML model, capable to automatically perform vertical service management tasks. Task like scaling of containers when the processing load and processing latency is high, restarting of services during crashes, performing migration to balance the load across all available resources.

3.5. Data Distribution and Unification

The data distribution and unification innovations are encapsulated in the DASS component. The DASS enables data processing to support highly distributed applications built on top of an open platform called Eclipse Zenoh. The Eclipse Foundation is a leader in building an open-source ecosystem including business models and is driven by European actors. A complete description of Zenoh-based Data Management was provided on D2.1 [18] in Section 8 Appendix Zenoh-based Data Management. The DASS innovations regarding data distribution, fault-tolerant routing infrastructure, and properly dispatch geo-distributed data are part of the zenoh.net layer.

Data management is a complex task comprising different approaches and techniques depending on the scenario and application domains. The technologies that deal with data management can be categorized into two different families according to its static vs dynamic form, and the different storage mechanism:

- **Data-at-rest:** this category refers to the traditional relational databases management systems (RDBMS) where data is organized into tables that can be linked, or related, based on certain key properties. This capability enables the retrieval of entirely new tables from data in one or more tables via queries. More recently, the appereance of NoSQL databases allows the storage of non-structured data. Both SQL and NoSQL database systems are examples of data-at-rest.
- **Data-in-motion:** this category refers to periodically generated or live streams of data. Usually follows the publisher/subscriber messaging pattern. For example, a temperature sensor sends periodical data to a weather station. This pattern provides greater network scalability and more dynamic network topology. The Data Distribution Service (DDS) is an example of such a messaging protocol. Some well-known implementations that support the pub/sub and message queues are e.g. JMS[1], Apache Kafka[2] and Redis[3]. The streamed data can have different formats e.g., video, images, or simply bytes.

Historically the *data-at-rest* was the predominant paradigm for IT systems, and *data-in-motion* was used everywhere else like operational technologies e.g., SCADA systems. This is however changing in recent years as the IT world embraces reactive and event-driven architectures. This introduces the challenge of dealing with *data-at-rest* and *data-in-motion* in a unified manner.

Regarding data distribution, the two main paradigms are cloud-centric approaches and decentralized/distributed ones. The limitations and constraints of traditional cloud-centric approaches are well explained in the literature, and in particular in D2.1 Section 8.1. It is because of such limitations that in 5G-DIVE we adopt a clean state approach for the DASS, designing from scratch a data platform and protocol capable of natively support the harmonization of *data-at-rest* and *data-in-motion* over a decentralized and distributed edge/fog environment, along with the necessary support and hooks to perform data analytics operations.

The data unification effort was focused on the *Data pre-processing* and *Data Storage* components of the DASS. Data unification can be achieved by native support of multiple data encoding, such as JSON, Properties, Relational, Raw, etc., along with transcoding across supported formats. The DASS also defines a canonical query syntax based on URIs syntax that enables filtering and querying for a particular subset of the data. During 5G-DIVE second year, the DASS also provides a storage back-end plug-in API, that facilitates the integration of third parties storage technologies. At the moment of

[1] https://download.oracle.com/otndocs/jcp/7195-jms-1.1-fr-spec-oth-JSpec/
[2] https://kafka.apache.org/
[3] https://redis.io/

writing this deliverable, the DASS supports the management of different storage back-ends technologies:

- In-memory backend: This backend represents main memory storage. It's not persistent, as soon as the application stops, all the keys/values stored in this backend are lost.
- SQL-Based backend: Backend and storages for using an SQL-based database e.g. MySQL, MariaDB, PostgreSQL, SQLite, RocksDB.
- Time series backend: This backend relies on a server and can handle a stream of data e.g. telemetry data.
- File system backend: This backend relies on the host's file system to implement the storage.

The DASS is inspired by the Named Data Networking (NDN) [6] paradigm to support access transparency of both data-at-rest and data-in-motion. Application's data (i.e. data payload) and data names are directly used in the network packet forwarding, allowing consumer applications to request the desired data by its name. In order to publish data in the network, a value (i.e., the data payload) is associated to a key (i.e., the data name), making de-facto a system based on key-value semantics. In doing so, the DASS provides a routing infrastructure capable of delivering the right data in the right place based on the set of available publishers and subscribers. It implements three kind of deployment units: peers, clients and routers.

- **Peers:** a simple user application producer/subscriber can communicate with other peers in a P2P mesh topology and connect to routers.
- **Client:** a user application that connects to a single router (or a single peer) to communicate with the rest of the system.
- **Router:** an infrastructure component able to route data between clients and peers in a given topology.

One of the key innovations introduced by the DASS is related to its mesh routing capabilities. Routers and peers can be deployed in a mesh topology, as depicted in Figure 3-7. Multiple routers can be deployed at the Edge between a mesh of routers and a mesh of peers for load balancing and fault tolerance purposes, this is achieved via automatic reconnection. Automatic reconnection is when clients, routers and peers try to reconnect to their configured peers after disconnection.

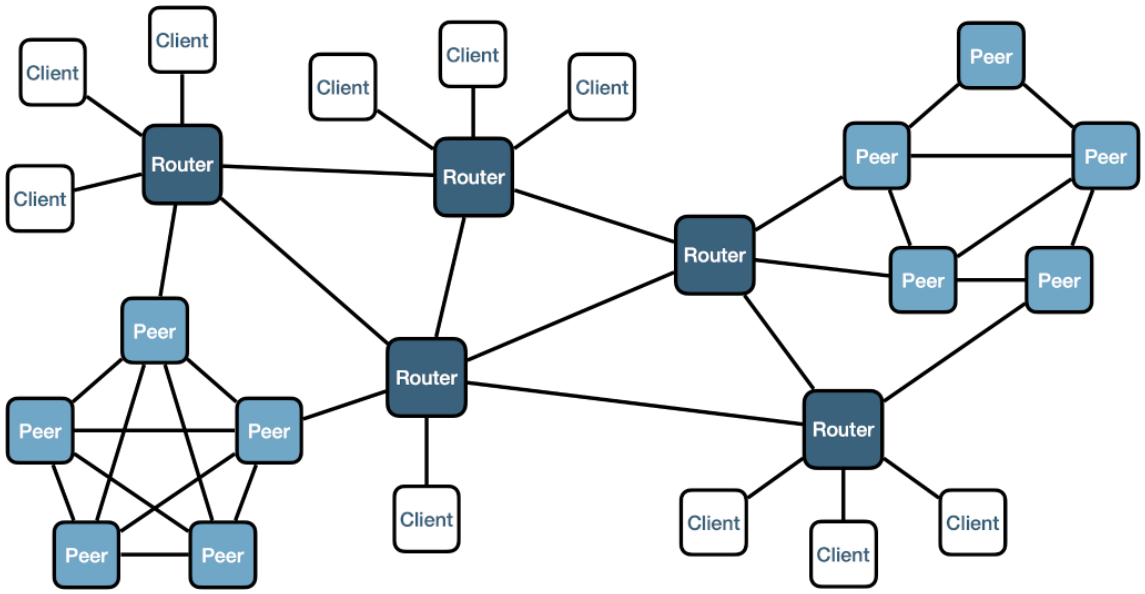


FIGURE 3-7: DASS DEPLOYMENT UNITS

By default, the geo-distributed storages work under eventual consistency. Stronger consistency can be implemented by users leveraging quorum mechanism. More information about the DASS was provided on D2.1 [18] in Section 8 Appendix Zenoh-based Data Management.

3.6. Federation Support and Resource Provisioning

The DEEP platform facilitates the integration of federated services and resources as if they were controlled and managed by the vertical industry domain. This has increasing importance to enable the creation and management of E2E vertical services, which may involve multiple administrative domains. Several examples can be envisioned that rely on such mechanisms, like in the integration of the PN with the NPN (i.e., PNI-NPN) or the usage of third party Cloud and Edge facilities to reduce costs.

3.6.1. *System & Service Providers (SSP)'* Federation

The *system & service provider* (SSP) represents the role of a network operator that not only provides network capabilities but also a range of different services such as connectivity, value-added applications and edge computing that result in selling a vertical complete system.

The regular approach is that, in which the vertical interacts with a single *system & service provider*. However, it is also possible to consider that the vertical interacts with different *system & service providers* to provide its service. This situation is detailed in Figure 3-8.

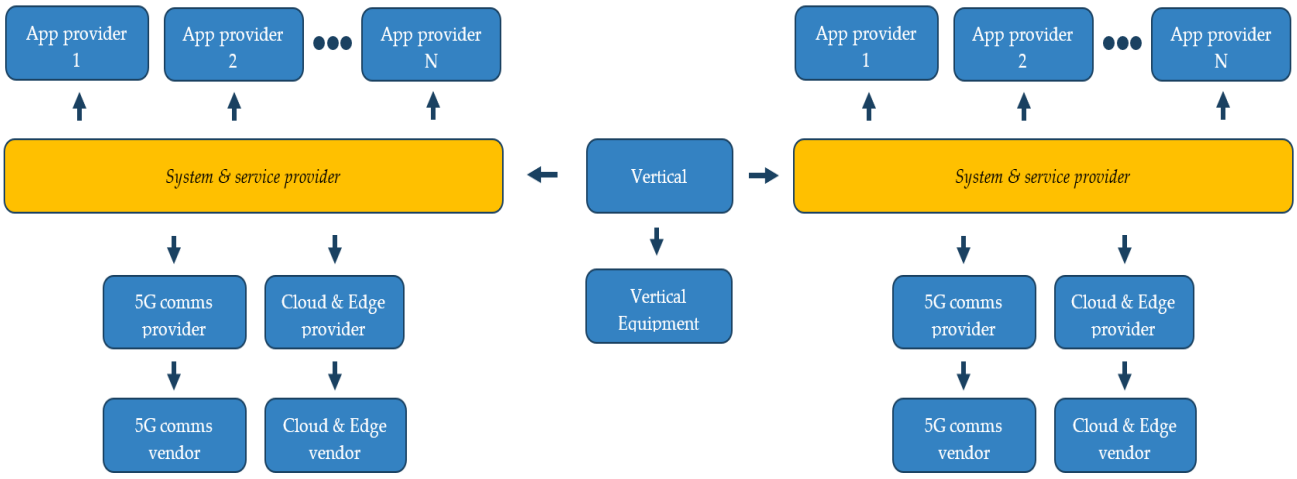


FIGURE 3-8: DUAL CONNECTION TO SSPS

Moreover, it could be the case that an SSP leverages on the infrastructure and capabilities of another SSP for complementing its own service and provide a better offer to the vertical. This can be seen as a recursive interaction or concatenation of SSPs, as illustrated in the Figure 3-9.

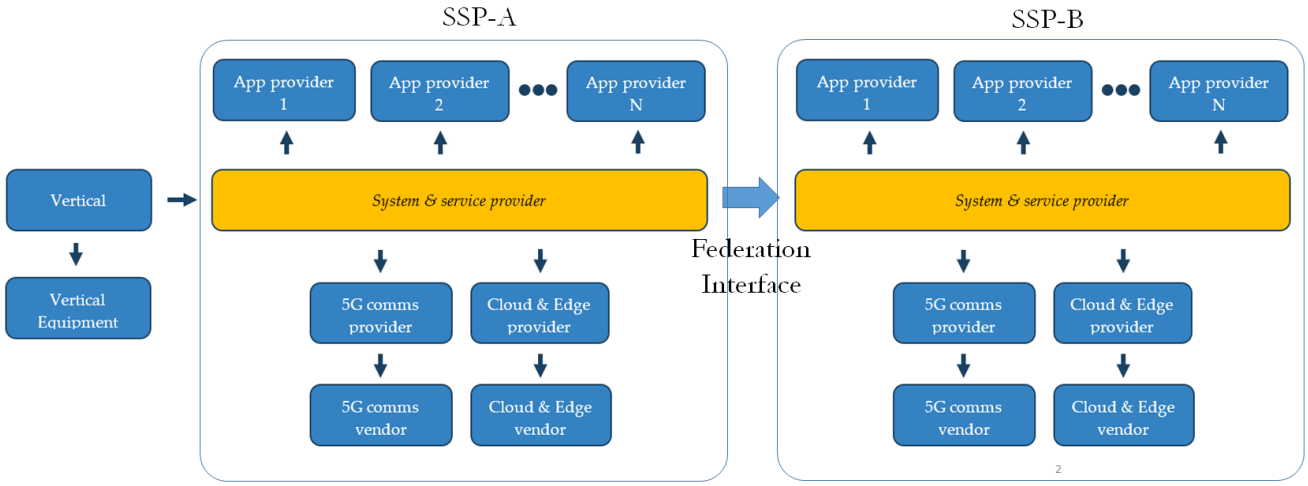


FIGURE 3-9: SSPS' FEDERATION

In this case it can be assumed that SSP-A behaves as a vertical customer to SSP-B.

Taking into account the stakeholders' structure within the project, some other scenarios of federation are made possible opening up the combination to several other possibilities, as follows:

- Different *system & service providers* sharing the 5G communications and Cloud & Edge providers.
- Common application providers and 5G communications and Cloud & Edge providers that are shared by different *system & service providers*.

3.6.2. Federation Support

Additional federation scenarios can be considered from a technology perspective leveraging on different architectural propositions around MEC / Edge Computing. This section overviews some possible

alternatives. All of them are in principle compatible with 5G-DIVE architecture, so the options below do not preclude a preferred alternative with respect the others.

3.6.2.1. GSMA Operator Platform Concept

The GSMA is promoting an initiative in line with the federation of edge computing capabilities from different operators across the world. In the context of 5G-DIVE, the edge computing substrate could leverage on the proposed GSMA Operator platform by using linked facilities from different operators and providers. Furthermore, this could be even extended with the idea of including in such federation also facilities from verticals, complementing the operator's footprint and portfolio of edge capabilities.

In essence, the Operator Platform (OP) for telco edge intends to offer a generic platform.

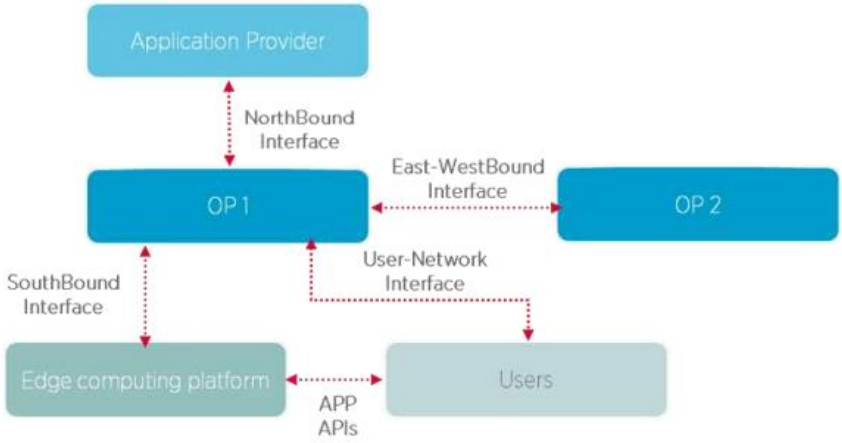


FIGURE 3-10: HIGH-LEVEL ARCHITECTURE OF GSMA'S OPERATOR PLATFORM (TAKEN FROM [19])

The OP architecture is based on the following main boundaries:

- Northbound Interface (NBI): enables service management and fulfilment of enterprise and Application Providers' use case requirements.
- East/Westbound Interface (E/WBI) APIs: the interface between instances of the OP that extend an operator's reach beyond their footprint. The OP architecture allows an OP from one operator to deploy applications provided by third parties on the OP of another operator (making effective the federation).
- Southbound Interface (SBI): connecting the OP with the specific operator infrastructure that will deliver the network services and capabilities to the user.
- User-Network Interface (UNI): enables the User Client (UC) hosted in the user equipment to communicate with the OP.

The OP exposes operator's network functions and resources to third-party applications that can leverage the overall edge capabilities of the federated environment. The OP concept provides edge compute resources on a virtualised basis to another party in the OP ecosystem (for example, an Application Provider, an aggregator, or another operator). Each tenant application is isolated from all other applications in the platform.

Any third party accessing the OP uses a single interface to manage applications deployed in the platform towards the subscribers of multiple operators participating in the federation. The Application Providers and OP do not require knowledge of each other's internal workings and implementation details (e.g., internal topology or cloudlet physical location).

Figure 3-11 provides a view of the OP reference architecture and components. The reader is referred to [19] for further details on interfaces and functionality.

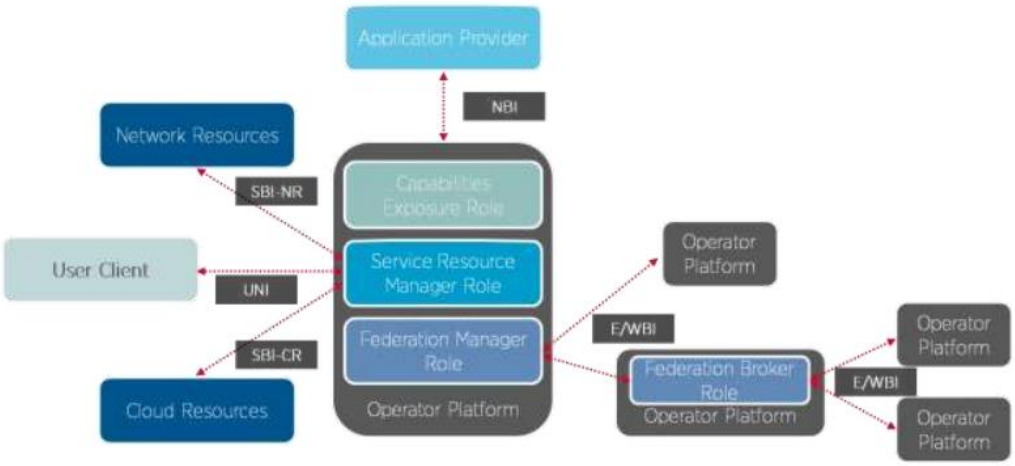


FIGURE 3-11: OP REFERENCE ARCHITECTURE (TAKEN FROM [19])

3.6.2.2. Inter-MEC architecture

With a similar purpose as before, ETSI MEC has been working on the concept of interconnecting MEC capabilities from different operators, known as inter-MEC, and detailed in the MEC 035 report [20] (yet a work-in-progress draft at the time of writing this deliverable). It defines different interconnection scenarios, also including the interconnection MEC-Cloud.

For inter-MEC system communication, the following considerations are assumed:

- A MEC platform can discover other MEC platforms that may belong to different MEC systems.
- A MEC platform can securely exchange information with other MEC platforms that may belong to different MEC systems.
- A MEC application can securely exchange information with other MEC applications that may belong to different MEC systems.

In order to allow such interconnection, the report proposes several alternatives for the federation of the MEC environments. Specifically, two options are proposed:

1. **Federation Manager (Figure 3-12):** responsible for supporting inter-MEC system communication with functionalities such as authorization, authentication, and control access for MEC federation members, application life cycle management, exposure of the catalogue of MEC systems, etc.

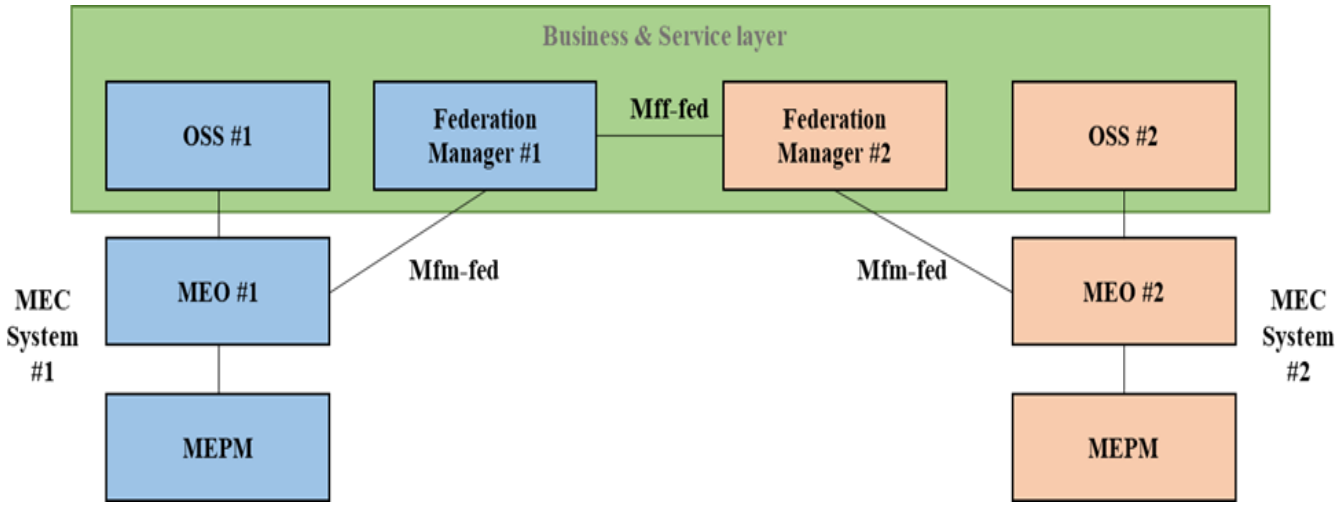


FIGURE 3-12: HIGH-LEVEL FRAMEWORK FOR FEDERATION MANAGER

2. **Federation Broker (Figure 3-13):** considered in order to reduce complexity to reach a high number of federation agreements.

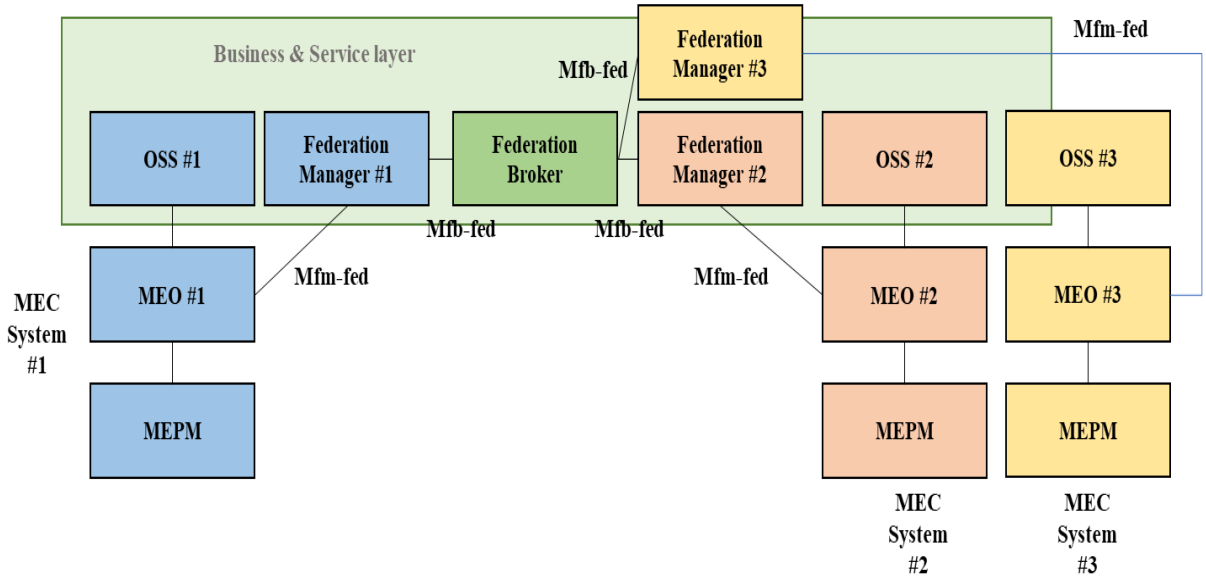


FIGURE 3-13: HIGH-LEVEL FRAMEWORK FOR FEDERATION BROKER

Not explicitly considered in the report, the interconnection can be even considered a further extension by for integrating it with NFV according to MEC 017 propositions [21].

3.6.2.3. Integration of multi-domain MEC environments

The inter-MEC architecture described before faces the federation of MEC environments considering coordination at the system level. However, alternative federation approaches can be considered from a IaaS, PaaS and SaaS perspective, where different actors can provide specific pieces of the MEC architecture. This is the approach taken in [22], investigated in the context of 5G-DIVE project.

A first integration approach would be to consider the usage of infrastructure from a different provider, nominally an infrastructure provider. This situation can be assimilated to an Infrastructure-as-a-Service

(IaaS) offering in the cloud computing business. Since MEC makes use of NFVI environments for hosting the applications and other virtualized functions, this scenario leads as well to integration of NFVI environments, probably requiring the interconnection of the overall NFVI substrates used by the MEC provider. Figure 3-14 represents the administrative boundary among providers in this model.

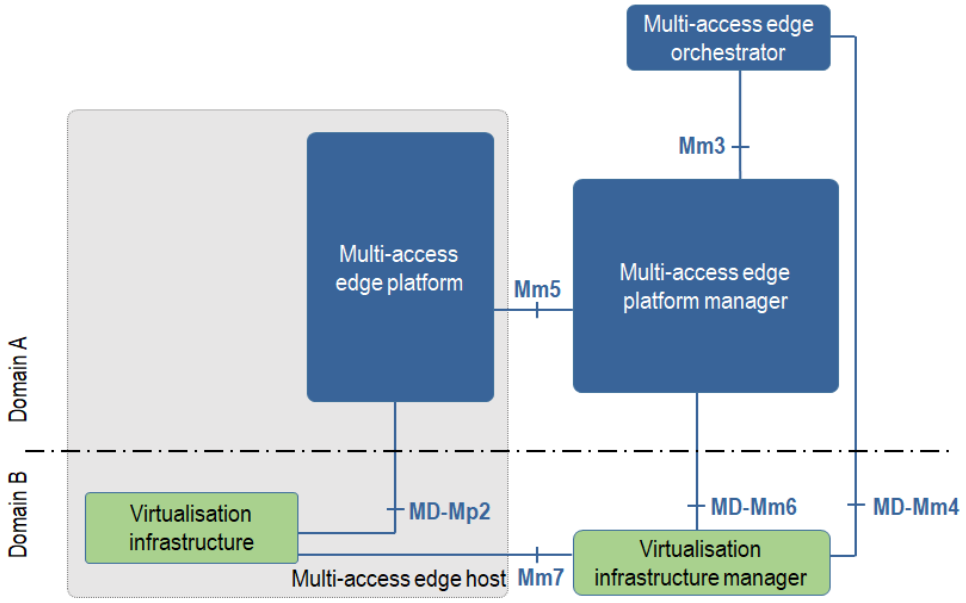


FIGURE 3-14: INTEGRATION AT INFRASTRUCTURE LEVEL

A different approach could be the integration with a domain that implements the MEP and possibly some specific applications. This approach can be perceived as a Platform-as-a-Service (PaaS) offering, also in analogy with the cloud computing world. The integration at PaaS level could present two different sub-scenarios: (i) integration with the platform provider with infrastructure owned by the primary MEC provider, Domain A; and (ii) integration with the platform provider, Domain B, including its infrastructure. Both scenarios are shown in Figure 3-15 and Figure 3-16, respectively.

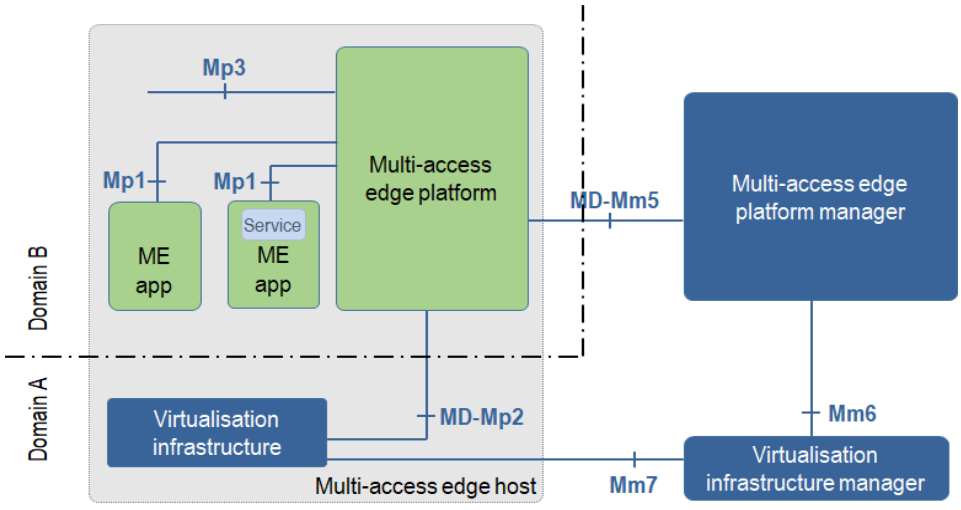


FIGURE 3-15: INTEGRATION AT PLATFORM LEVEL WITH INFRASTRUCTURE OWNED BY DOMAIN A

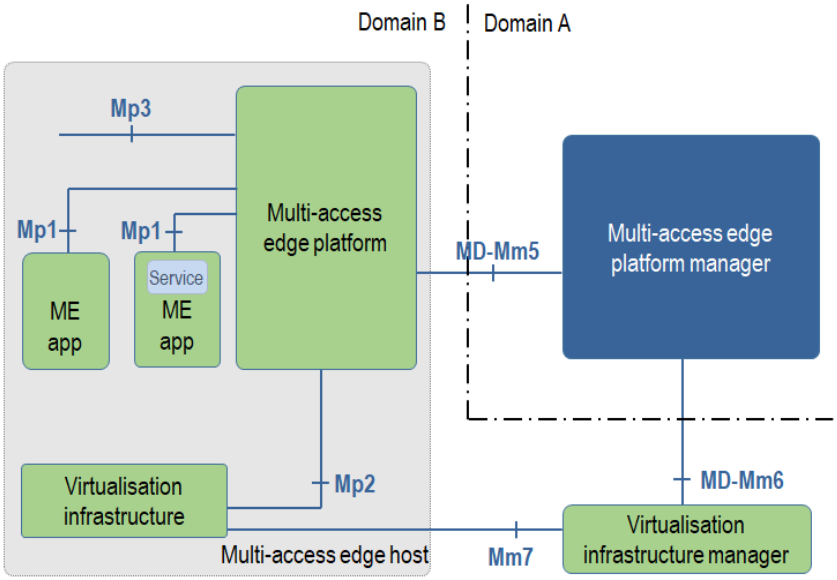


FIGURE 3-16: INTEGRATION AT PLATFORM LEVEL WITH INFRASTRUCTURE OWNED BY DOMAIN B

In this case, the primary domain, Domain A, implements only the MEO function, interconnecting to the MEPM and the VIM of the secondary domain for the orchestration of the applications as provided or enabled by Domain B. Then provider in Domain A focuses on the commercial relation with the MEC customer (and the end-users) and in the decisions about instantiating and running applications in the system.

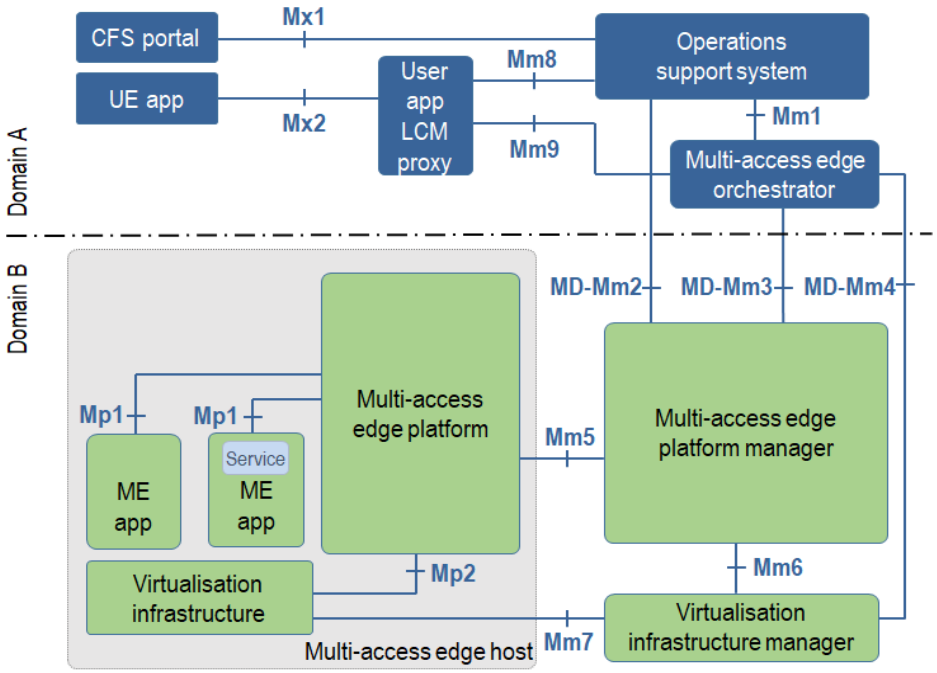


FIGURE 3-17: INTEGRATION AT SERVICE LEVEL

4. Update regarding DEEP Integration with 5G-CORAL

D1.1 [1] provided a general description regarding the integration of the DEEP platform to a 5G-CORAL based system. This section provides an update with more explanations on the interconnection among VSS/OSS/BSS, DEEP and 5G-CORAL edge system. Here, the integration of 5G-CORAL Edge system is also provided as one example, which would help understand how other edge systems can be integrated with the DEEP platform. To facilitate the denotation, we use OSS to represent VSS/OSS/BSS in the rest of this section.

As defined in 5G-DIVE, the DEEP platform is designed as an add-on platform between OSS and the underlying Edge systems, such as 5G-CORAL Edge system. The aim of DEEP is to provide the new capabilities of AI/ML, data handling and business automation via IESS, DASS and BASS, respectively. When interfacing with an Edge system, the southbound interfaces of the DEEP platform need to be supported by the northbound interfaces of the Edge system. If the Edge functions or applications want to utilize DEEP functionalities provided by IESS, DASS and BASS, the Edge functions or applications may need to be upgrade to support the relevant southbound interfaces of the DEEP platform. As an example, the following will provide more information regarding DEEP integration to 5G-CORAL Edge system.

Figure 4-1 shows a simplified picture of the architecture of a standalone 5G-CORAL Edge system [23]. The DEEP platform will be added between OSS and 5G-CORAL Edge system. Therefore, the relevant DEEP interfaces should connect 5G-CORAL Edge system via its T1, T2 and T3 interfaces, which are shown in Figure 4-2.

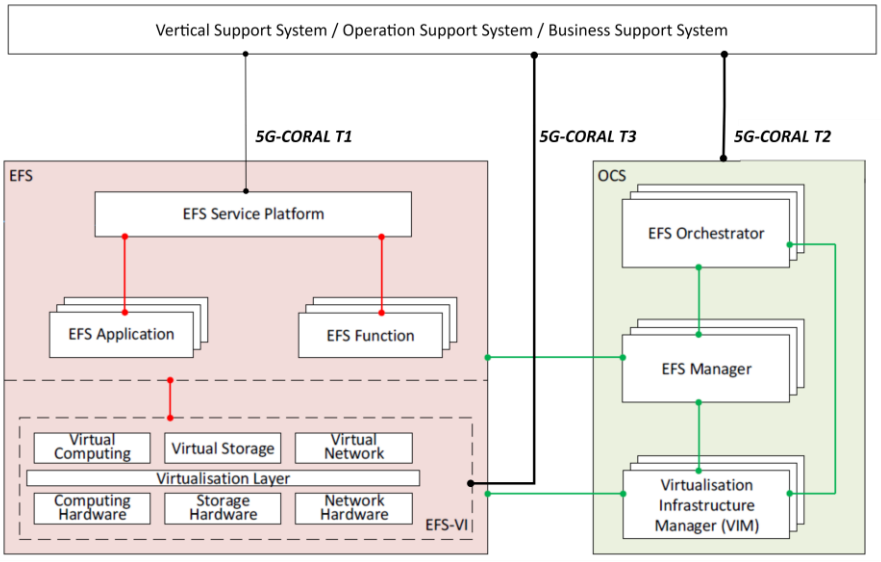


FIGURE 4-1: STANDALONE 5G-CORAL EDGE SYSTEM WITHOUT DEEP

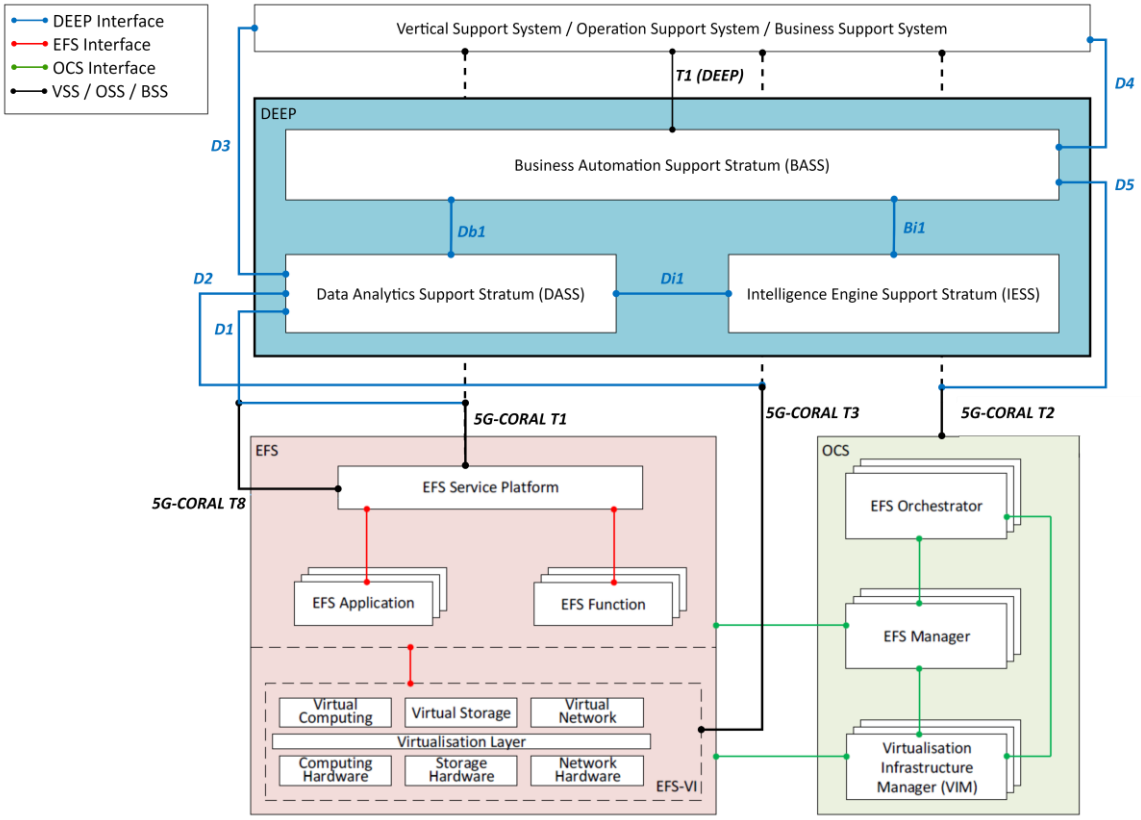


FIGURE 4-2: DEEP INTERGRATION WITH 5G-CORAL EDGE SYSTEM

DEEP D1 interface is connected to 5G-CORAL T1 and T8 interface towards the EFS service platform. The D1-T8 interconnection makes the data exchange possible between EFS applications/functions and DASS in DEEP, which enables EFS application/function to utilize the functionalities provided by DEEP. The original 5G-CORAL T1 interface connects the OSS to the 5G-CORAL EFS service platform, which can be done now through the D3 interface and the D1-T1 interconnection. The original 5G-CORAL T2 interface connects the OSS to the OCS, which can be done now through the D4 interface and the D5-T2 interconnection. The BASS in DEEP platform will also use D5-T2 interconnection to communicate with the orchestrator in OCS. The original T3 interface connects between OSS and the EFS virtualization infrastructure (EFS-VI), which can be done now through the D3 interface and the D2-T3 interconnect. DEEP can also directly use the D2-T3 interconnect to communicate with the EFS-VI. The dashed lines connecting OSS to 5G-CORAL T1, T2 and T3 interfaces represent a backup connection. In case the DEEP platform is down or malfunctioning, OSS can fallback to the standalone mode of 5G-CORAL Edge system without DEEP using these connections. In this way, DEEP is fully integrated to 5G-CORAL Edge system reusing all existing interfaces defined. For completeness, more information regarding these interfaces of 5G-CORAL and DEEP are provided in Table 4-1 and Table 4-2, respectively.

TABLE 4-1: RELEVANT 5G-CORAL INTERFACES FOR DEEP INTEGRATION

5G-CORAL interface	Description
T1	This is the reference point between the EFS service platform entity manger and the Operation Support System/Business Support System (OSS/BSS).

T2	This interface is used for exchange between OSS/BSS and EFS Orchestrator, and supports the following: • EFS Stack Descriptor and EFS Stack lifecycle management, including EFS Stack instantiation, update, scaling, migration, termination, and query (e.g., retrieving summarised information about edge and fog resources associated to the EFS Stack instance); • Policy management and or enforcement for EFS Stack instances, function and application instances, and edge and fog resources (e.g., authorisation, access control, resource reservation, placement, allocation, etc.); • Forwarding of events, accounting and usage records and performance measurement results regarding EFS Stack instances, application and function instances, and edge/fog resources to OSS/BSS, as well as and information about the associations between those instances and edge/fog resources; • Integrating and releasing of resources into/from the target EFS including third-party information and SLAs.
T3	This is the reference point between the EFS virtualisation infrastructure (EFS-VI) and the Operation Support System/Business Support System (OSS/BSS). ETSI NFV has an interface between NFVI and OSS/BSS, however, this interface is not named and is classified under "other references"
T8	This is the refence point between the EFS service platform and the Non-EFS applications, functions and resources. There is no equivalent interface both in ETSI NFV and ETSI MEC.

TABLE 4-2: RELEVANT DEEP INTERFACES FOR DEEP INTEGRATION

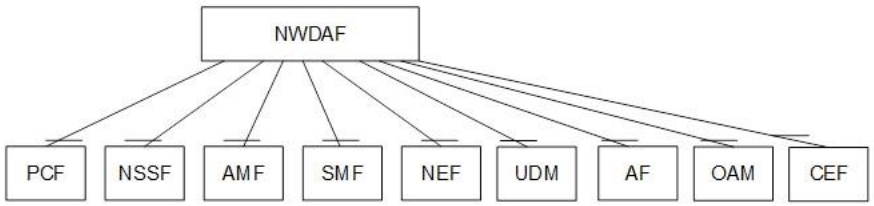
DEEP interface	Description
D1	This interface is used for exchanging data between the VNFs of a given vertical service and DASS, and supports the following: • Sharing information exposed by each VNF • Defining the situation- and context-aware of the exposed information • Defining model and knowledge information with respect to the exposed information
D2	This interface is used for exchange between the virtualization infrastructure and DASS, and supports the following: • Fetching (abstracted) monitoring information regarding the different resources comprising the virtualization infrastructure
D3	This interface is used for exchange between VSS/OSS/BSS and DASS, and supports the following: • Sharing of context data related to the requested vertical services, which may be relevant for training AI/ML applications with the purpose of fulfilling the vertical service intent-based AI capabilities
D4	This interface is used for exchange between BASS and VSS/OSS/BSS, and supports the following: • Provide notifications about SLAs and policy conflicts

	• Providing notifications regarding the automated decisions that were enforced by the BASS regarding the operation and management of a running vertical services • Providing notifications in case the BASS becomes unable to comply with the required actions and, therefore, requires assistance from the VSS/OSS/BSS
D5	This interface is used for exchange between BASS and Orchestrator, and supports the following: • Edge computing infrastructure Stack Descriptor and Stack lifecycle management, including Stack instantiation, update, scaling, migration, termination, and query (e.g., retrieving summarised information about edge and fog resources associated to the Edge stack instance); • Policy management and or enforcement for Edge stack instances, function and application instances, and edge and fog resources (e.g., authorisation, access control, resource reservation, placement, allocation, etc.); • Forwarding of events, accounting and usage records and performance measurement results regarding Edge stack instances, application and function instances, and edge/fog resources to OSS/BSS, as well as and information about the associations between those instances and edge/fog resources; • Integrating and releasing of resources into/from the target Edge including third-party information and SLAs.

5. Architecture Mapping to 3GPP Standards

In this section, a mapping of the DEEP platform to the relevant standards is presented. The DEEP is by design a platform for vertical services automation that leverages on AI/ML to deliver intelligence at the Edge. The intelligence delivered is highly dependent on raw data creation and flows within the system, and with the vertical use cases at its origin. The target connectivity technology in the project is 5G and for this reason, the main focus of this section and mapping relates to 3GPP standards. In this body, the relevant definitions that relate to and impact the 5G-DIVE project are agreed in the Technical Specification Group for Service and System Aspects (TSG SA2), the working group responsible for the development of the overall 3GPP system architecture and services including User Equipment, Access Network, Core Network, and IP Multimedia Subsystem.

A component is defined, the Network and Data Analytics function (NWDAF), that acts as a central entity responsible for data collection (by on-demand requests or via subscription means) and interaction with databases for data storage. The NWDAF can interact with several Core Network functions, as depicted in Fig. 5-1. By covering a wide range of network functions, these interfaces allow for data collection from virtually all aspects of the 5G network.



Legend: PCF - Policy Control Function; NSSF - Network Slice Selection Function; AMF - Access and Mobility Management Function; SMF - Session Management Function; NEF – Network Exposure Function; UDM – Unified Data Management; AF - Application Function; OAM - Operations, Administration and Maintenance Function; CEF- Charging Enablement Function

FIGURE 5-1: AVAILABLE INTERFACES BETWEEN NWDAF AND OTHER NETWORK FUNCTIONS [24]

The NWDAF may or not have AI/ML model training functionalities. These functionalities are named Model Training Logical Function (MTLF).

The data gathered by the NWDAF can be collected from any network function, including the Operations, Administration and Maintenance (OAM) function. Two interfaces are defined for data exchanges between NWDAF and other Network Functions (NFs) as depicted in Figure 5-2 and Figure 5-3.

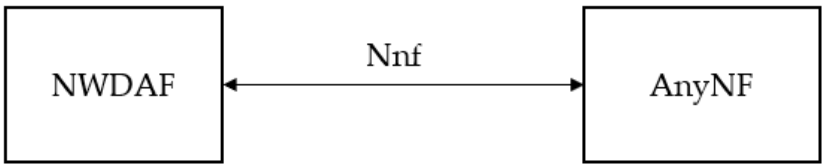


FIGURE 5-2: NNF INTERFACE [25][26]

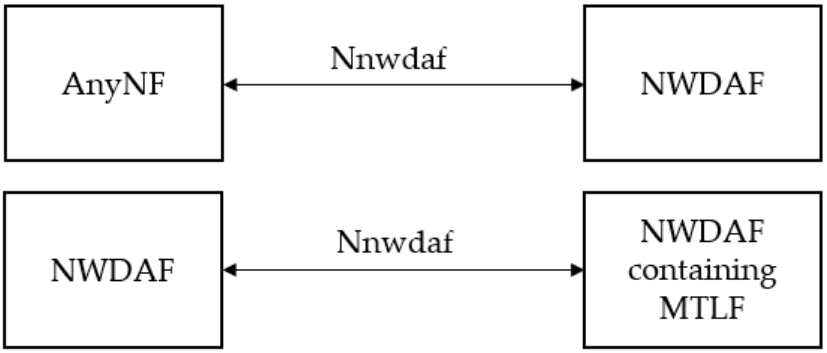


FIGURE 5-3: NNWDAF INTERFACE [25][26]

The Nnf interface is defined for the NWDAF to request the subscription to data delivery for a particular context, to cancel the subscription to data delivery and to request a specific report of data for a particular context.

The Nnwdaf interface is defined for 5GC NFs, to request the subscription to network analytics delivery for a particular context, to cancel the subscription to network analytics delivery, to request a specific report of network analytics for a particular context, and for requesting and subscribing to model training services.

Different components are defined to support OAM data collection. Figure 5-4 depicts these components and the corresponding interfaces. New components are the Data Collection Coordination Function (DCCF), the Messaging Framework Adaptor Function (MFAF) and the Network Repository Function (NRF).

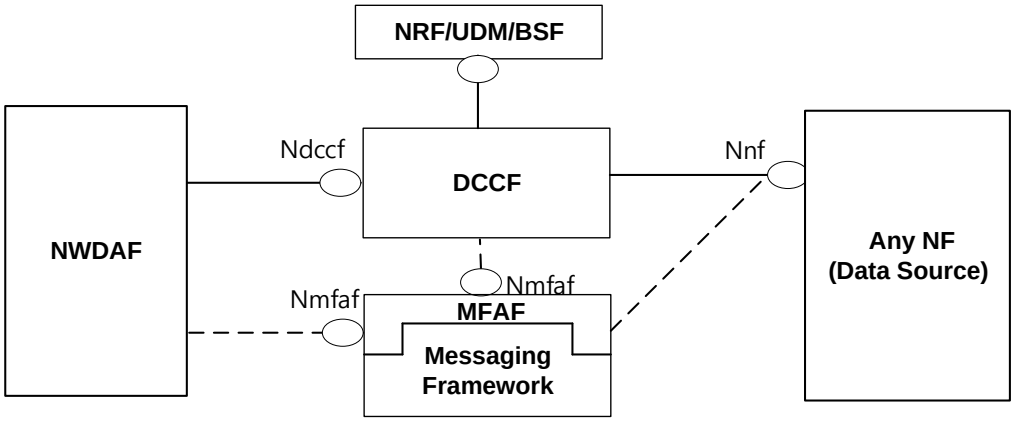


FIGURE 5-4: INTERFACES FOR OAM DATA COLLECTION [25,26]

The DCCF is responsible for:

- Determining which data sources can provide data for a received data request.
- Determining whether data is already being collected from a data source.
- Instructing a Messaging Framework to send data to consumers or notification endpoints.

- Instructing a Messaging Framework to do formatting and processing of the data sent via the Messaging Framework.
- Formatting and processing of data.
- Sending data to consumers or notification endpoints.
- Registering NWDAFs.

These functionalities match the functionalities of the data dispatcher component of the DEEP.

The MFAF is responsible for:

- Interfacing with a DCCF that controls how a messaging framework will process, format, and send data to consumers or notification endpoints.
- Receiving data from Data Sources via services offered by those Data Sources.
- Sending data received from Data Sources to a messaging framework.
- Receiving data from a messaging framework.
- Processing, formatting, and sending data to specified consumers or notification endpoints.

These functionalities match the functionalities of the data pre-processing component of the DEEP.

The NRF supports the following functionality:

- Supports service discovery function. Receive NF Discovery Request from NF instance or SCP, and provides the information of the discovered NF instances (be discovered) to the NF instance or SCP.
- Supports P-CSCF discovery (specialized case of AF discovery by SMF).
- Maintains the NF profile of available NF instances and their supported services.
- Maintains SCP profile of available SCP instances.
- Supports SCP discovery by SCP instances.
- Notifies about newly registered/updated/ deregistered NF and SCP instances along with its potential NF services to the subscribed NF service consumer or SCP.
- Maintains the health status of NFs and SCP.

These functionalities match the functionalities of the data storage component of the DEEP.

The Ndccf interface is defined for the NWDAF to support subscription request(s) for data delivery from a DCCF, to cancel the subscription to data delivery, and to request a specific report of data.

Table 1 summarises the mapping between 5G-DIVE components and interfaces and the relevant 3GPP up to date defined components and interfaces.

TABLE 5-3: MAPPING OF 5G-DIVE COMPONENTS TO 3GPP COMPONENTS

Component Mapping		Interface mapping	
5G-DIVE component	**3GPP component**	**5G-DIVE interface**	**3GPP interface**
AI/ML manager	NWDAF	Di1	Nnf / Nnwdaf
AI/ML Model training	NWDAF with MTLF	Bi1	Nanf
Data Storage	UDR	---	---
Data pre-processing	MFAF	---	---

Data Dispatcher	DCCF	---	---
SLA & Policy Management/Vertical Service Coordinator	UPF/AMF	---	---

6. Conclusions

This deliverable presented an update of the 5G-DIVE solution, focusing on several refinements of the DEEP platform and its key innovation in terms of abstraction, intelligence, and automation. It then details the integration of the DEEP platform with 5G-CORAL as the embodiment of the underlying Edge Computing Platform and also presents the mapping of the 5G-DIVE solution with relevant 3GPP standards.

The refinements applied in the DEEP platform were motivated by advances in the platform implementation, deployment, and integration with the selected use cases in the project. These aimed mostly to make operations more straightforward and flexible, reducing the complexity of the DEEP platform. The workflows for basic operations envisioned to be performed by the DEEP platform are described in this deliverable, with the proposed refinements making it simpler to accommodate other workflows and utilization scenarios.

The DEEP platform also defines a set of innovations aiming at hiding the underlying complexity to the vertical and automating the provisioning and lifecycle management of vertical services. Thus, the DEEP platform can act on behalf of the vertical to enforce automation of its business processes, allowing the vertical to focus on its knowledge domain. The key innovations are: *(i)* vertical service abstraction; *(ii)* support for AI/ML-based intelligence; *(iii)* data distribution and unification; *(iv)* active monitoring; *(v)* federation support.

Finally, the 5G-DIVE solutions proposes the deployment of the DEEP platform as an add-on on top of Edge computing infrastructure. Such concept is verified through its high-level integration with 5G-CORAL, which is completely transparent for the latter.

As future work, an evaluation on how the 5G-DIVE architecture serves the selected Industry 4.0 and Autonomous Drone Scouting use cases, identifying the main benefits for these vertical industries and drawing the final conclusions with respect of the future exploitation of the DEEP innovations.

7. References

[1] 5G-DIVE. "D1.1: Architecture and detailed analysis of vertical use cases", March 2020 [Online]. Available: https://5g-dive.eu/wp-content/uploads/2021/05/D1.1_Final_updated-with-review-comments.pdf

[2] 5G-ACIA. "5G Non-Public Networks for Industrial Scenarios (White Paper)", July 2019 [Online]. Available: https://5g-acia.org/wp-content/uploads/2021/04/WP_5G_NPN_2019_01.pdf

[3] GSMA. "5G IoT Private \& Dedicated Networks for Industry 4.0 - A guide to private and dedicated 5G networks for manufacturing, production and supply chains", October 2020 [Online]. Available: https://www.gsma.com/iot/wp-content/uploads/2020/10/2020-10-GSMA-5G-IoT-Private-and-Dedicated-Networks-for-Industry-4.0.pdf

[4] 5G-DIVE. "D1.2: 5G-DIVE Techno-economic Analysis", November 2020 [Online]. Available: https://5g-dive.eu/wp-content/uploads/2021/01/D1.2-5G-DIVE-TEA.pdf

[5] 5G-DIVE. "D2.3: Final Specification of 5G-DIVE Innovations", June 2021 [Online].

[6] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. C. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang. "Named data networking." ACM SIGCOMM Computer Communication Review 44, no. 3, 2014

[7] Y. Wu, "Cloud-Edge Orchestration for the Internet-of-Things: Architecture and AI-Powered Data Processing", IEEE Internet of Things Journal, August 2020

[8] D. M. Gutierrez-Estevez et al., "Artificial Intelligence for Elastic Management and Orchestration of 5G Networks", IEEE Wireless Communications, vol. 26, no. 5, pp. 134-141, October 2019.

[9] 5G-TRANSFORMER. "D3.1: Definition of vertical service descriptors and SO NBI" [Online]. Available: http://5g-transformer.eu/wp-content/uploads/2018/03/D3.1_Definition_of_vertical_service_de-scriptors_and_SO_NBI.pdf

[10] ETSI. "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Network Service Templates Specification - ETSI GS NFV-IFA 014 V2.5.1", August 2018 [Online]. Available: https://www.etsi.org/deliver/etsi_gs/nfv-ifa/001_099/014/02.05.01_60/gs_nfv-ifa014v020501p.pdf

[11] ETSI. "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; VNF Descriptor and Packaging Specification - ETSI GS NFV-IFA 011 V2.5.1", August 2018 [Online]. Available: https://www.etsi.org/deliver/etsi_gs/nfv-ifa/001_099/011/02.05.01_60/gs_nfv-ifa011v020501p.pdf

[12] 5G EVE. "D4.3 Models for vertical descriptor adaptation" [Online]. Available: https://zenodo.org/record/4311316

[13] ETSI. "ETSI EG 202 009-3 Part 3: Template for Service Level Agreements (SLA)", April 2015 [Online]. Available: https://www.etsi.org/deliver/etsi_eg/202000_202099/20200903/01.03.00_50/eg_20200903v010300m.pdf

[14] GEYSERS. “SLA Management for Composite Infrastructure as a Service” [Online]. Available: http://lmcontreras.com/wp-content/uploads/2015/12/geysers-sla-whitepaper.pdf

[15] R. Rokui, S. Homma, K. Makhijani, L. M. Contreras, J. Tantsura. "Definition of IETF Network Slices", February 2021 [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-teas-ietf-network-slice-definition-01

[16] 5Growth. D2.2, “Initial implementation of 5G End-to-End Service Platform”. Available: https://5growth.eu/wp-content/uploads/2020/05/D2.2-Initial_implementation_of_5G_End-to-End_Service_Platform.pdf

[17] Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: “The pascal visual object classes (voc) challenge”. International journal of computer vision 88 (2), 2010

[18] 5G-DIVE. “D2.1: 5G-DIVE Innovations Specification.”, September 2020 [Online]. Available: https://5g-dive.eu/wp-content/uploads/2021/01/D2.1-5G-DIVE-innovations-specification_v1.0_compressed.pdf

[19] GSMA. “Operator Platform Telco Edge Proposal”, version 1.0, October 2020. [Online]: https://www.gsma.com/futurenetworks/wp-content/uploads/2020/10/GSMA-Operator-Platform-Telco-Edge-Proposal-2020-Final.pdf

[20] ETSI. Draft ETSI GR MEC 035, “Study on Inter-MEC systems and MEC-Cloud systems coordination”, V3.0.0, April 2021. [Online]: https://docbox.etsi.org/ISG/MEC/Open/gr%20mec-0035v300_final%20draft%20for%20review_clean.pdf

[21] ETSI. ETSI GR MEC 017, “Deployment of Mobile Edge Computing in an NFV environment”, V1.1.1, February 2018. [Online]: https://www.etsi.org/deliver/etsi_gr/mec/001_099/017/01.01.01_60/gr_mec017v010101p.pdf

[22] L.M. Contreras, C.J. Bernardos, “Overview of Architectural Alternatives for the Integration of ETSI MEC Environments from Different Administrative Domains”, Electronics, September 2020.

[23] 5G-CORAL. “D1.3–5G-CORAL refined system design and future directions”, May 2019 [Online]. Available: http://5g-coral.eu/wp-content/uploads/2019/06/D1.3_final.pdf

[24] 3GPP, “TS 29.520”, 3GPP, Online, V17.2.0, March 2021.

[25] Vivo, “3GPP TSG-WG SA2 Meeting #144E e-meeting,” 3GPP, S2-2103275, April 2021.

[26] 3GPP, “TS 23.288”, 3GPP, Online, V17.0.0, March 2021.