“MOBILIZADOR 5G”

Components and services for 5G networks

Project nº 24539

Deliverable D2.2

Architecture for solutions targetting 5G network core

Relatório D2.2

Arquitetura de soluções para core de redes 5G

PPS	**Produtos e serviços para o core da rede**
Activity	**A2 – Especificações Técnicas**
Dissemination level	**Public**
Date	**December 2018**
Version	**1.0**

Project confinanced by:

Lisb@20 20 COMPETE 2020 PORTUGAL 2020 UNIÃO EUROPEIA Fundo Europeu de Desenvolvimento Regional

Project Leader:
Altice Labs, S.A.
Rua Eng. José Ferreira Pinto Basto
3810-106 Aveiro – Portugal
http://www.alticelabs.com
Tel: +351 234 403 200
Fax: +351 234 424 723

Sumário executivo

Esta entrega descreve a arquitetura de soluções para o core das redes 5G (5GC). As soluções propostas baseiam-se em estruturas de *Policy*, *Orchestration*, *Assurance*, *Support* e *Security*.

As soluções endereçadas abrangem, em certa medida, o chamado "*closed loop*" [1], que será o facilitador de um sistema autónomo que pode coletar dados, correlacionar eventos e transformar todos esses dados em inteligência para uma auto-correcção e auto-escalonamento do ecossistema 5G. O *Assurance* monitoriza os níveis de serviço acordados que, juntamente com o *Policy*, o ponto de decisão do ecossistema 5G, fornece à *Orchestration* os meios para atuar com precisão na rede, levando a ações de correção ou dimensionamento. Esses serviços são baseados em sistemas de suporte, como DNS, vitais para as operações de rede e com procedimentos de segurança que visam reduzir ou anular completamente as ameaças para as novas redes 5G.

Depois de revistos os objecticos da arquitetura 5G, em consonância com a arquitetura do sistema 3GPP 5G [2], esta entrega foca-se na visão funcional de todas as soluções ambicionadas, descrevendo o desenho e a implementação de elementos funcionais, suas responsabilidades, interfaces e interações principais das várias partes da arquitetura.

A definição da arquitetura foi validada contra um conjunto de requisitos identificados, provenientes dos casos de uso apresentados na entrega anterior D2.1 [3]:

- Orquestração de Serviço *vCDN* em 5G
- *PPDR* alavancando *IoT* em 5G

Por fim, é endereçado o modelo de colocação, sendo descrito o ambiente de execução lógico e físico disponível para posterior instalação dos produtos.

A arquitetura definida nesta entrega servirá de base para a implementação e validação das soluções dos vários protótipos a serem entregues, no futuro, pelo projeto.

Executive Summary

This Deliverable outlines the architecture for solutions targeting the 5G network core. The proposed solutions are based on *Policy*, *Orchestration*, *Assurance*, *Support* and *Security* frameworks.

The addressed solutions cover, to some extent, the so called "*closed loop*" [1], that will be the enabler of an autonomous system that can collect data, correlate events and turn all this underlying data into intelligence for a self-healing and self-scaling 5G ecosystem. *Assurance* monitors the agreed service levels that, together with *Policy* (the decision point of the 5G ecosystem), provides to the *Orchestration* the means to accurately actuate over the network, leading to correction or scaling actions. These services are based on *Support* services like DNS, vital for the network operations, and with *Security* procedures that aim to reduce or completely nullify the threat towards the upcoming 5G networks.

After going trough the 5G architecture goals, in line with the 3GPP 5G System Architecture [2], the Deliverable focus on the functional view of all the targeted solutions, describing the design and implementation of functional elements, their responsibilities, interfaces, and primary interactions of the several architecture parts.

The architecture definition was validated against a set of identified requirements, coming from the use cases presented in the previous deliverable D2.1 [3]:

- vCDN Service Orchestration in 5G
- PPDR leveraging IoT in 5G

At last, the planned deployment view is high level addressed, where the logical and physical run-time environments available to later deploy the products are described.

The architecture, defined in this Deliverable, will serve as basis for the implementation and validation of the solutions in the various prototypes, subject of future project deliveries.

Table of Contents

List of Figures

List of Tables

Glossary

3GPP	3rd Generation Partnership Project
5GC	5G Core
5QI	5G QoS Indicator
AAA	Authentication, Authorization and Accounting
ACP	Access Control Point
AF	Application Function
AI	Artificial Intelligence
AM	Autonomic Manager
AMF	Access and Mobility Function
ANN	Artificial Neural Network
API	Application Programming Interface
BSS	Business Support System
CAPEX	Capital Expenditure
CCC	Command Control Center
CERT	Computer Emergency Response Team
COTS	Commercial Off The Shelf
CPU	Central Processing Unit
CRUD	Create/Read/Update/Delete
D2D	Device to Device
DC	Data Center
DDoS	Distributed Denial of Service
DÉCOR	Dedicated Core Network
DevOps	Development and Operations
DNN	Deep Neural Networks
DNS	Domain Name Server
DNS	Domain Name System
DNSaaS	DNS as a Service
DPDK	Data Plane Development Kit
E2E	End-to-End
ECA	Event/Condition/Action
eMBB	Enhanced Mobile Broadband
EMS	Element Management System
EPA	Enhanced Platform Awareness
EPC	Evolved Packet Core
eV2X	Enhanced Vehicle-to-Everything
FCAPS	Fault, Configuration, Accounting, Performance, Security
FW	Firewall
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HPLMN	Home PLMN
IDPS	Intrusion Detection and Protection System
IDS	Intrusion Detection System

IM	Information Model
IoT	Internet of Things
IPS	Intrusion Prevention System
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LB	Load Balancer
LTE	Long Term Evolution
MANO	Management and Orchestration
MCPTT	Mission Critical Push to Talk
MEC	Multi-access Edge Computing
ML	Machine Learning
mMTC	Massive machine type communication
MQTT	Message Queuing Telemetry Transport
MVNO	Multi Virtual Network Operator
NBI	Northbound Interface
NEF	Network Exposure Function
NF	Network Functions
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NFVM	Network Function Virtualization Manager
NFVO	Network Function Virtualization Orchestrator
NGMN	Next Generation Mobile Networks
NG-RAN	Next Generation Radio Access Network
NRF	Network Repository Function
NS	Network Service
NSI	Network Slice Instance
NVT	Network Vulnerability Tests
NWDAF	Network Data Analytics Function
O&M	Operations and Management
OODA	Observe-Orient-Decide-Act
OPEX	Operational Expenditure
OSS	Operations Support System
OTT	Over-the-top
PAP	Policy Administration Point
PCC	Policy and Charging Control
PCF	Policy and Charging Function
PCM	Policy Context Manager
PCRF	Policy and Charging Rules Function
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PLMN	Public Land Mobile Network
PNF	Physical Network Function
PPDR	Public Protection and Disaster Relief
PPS	Product, Process or Service
ProSE	Proximity Services

QoS	Quality of Service
RAM	Random Access Memory
RBM	Restricted Boltzmann machines
RLP	Reasoning, Learning and Prediction
RNN	Recurrent Neural Networks
SA	System Architecture
SDN	Software Defined Networking
SDNC	Software Defined Network Controller
SFC	Service Function Chaining
SIEM	Security Information and Event Management
SLA	Service Level Agreement
SMF	Session Management Function
SON	Self-Organized Networks
SSH	Secure Shell
TETRA	Terrestrial Trunked Radio
TR	Technical Report
TS	Technical Specification
UDR	User Data Repository
UHD	Ultra-High Definition
URLCC	Ultra-Reliable and Low Latency Communications
VAS	Vulnerability Assessment System
vCDN	Virtualized Content Delivery Network
VDU	Virtual Deployment Unit
VIM	Virtualized Infrastructure Manager
VM	Virtual Machine
VNF	Virtualized Network Function
VNFC	Virtualized Network Function Component
VNFD	Virtualized Network Function Descriptor
VNFM	Virtualized Network Function Manager
VoD	Video on Demand
WLAN	Wireless Local Area Network

1 Introduction

The purpose of this document is to describe each of the solutions proposed for core 5G networks (i.e. Policy Framework, Orchestration Framework, Assurance Framework, Support Framework, Security Framework). As such, it provides a detailed description of the different solutions, their respective architectures (including internal modules and interfaces), as well as the external views regarding potential relationships between solutions.

This document has the following structure:

- Architectural Goals − Describe the implementation of the applicable architectural goals for the 5G core proposed solutions.
- Functional View − Describe the design and implementation of functional elements, their responsibilities, interfaces, and primary interactions of the several architecture parts.
- Concurrency View − Describe the design and implementation of use case flows to be carried out during the pilot phase.
- Deployment View − Describe the logical and physical run-time environment available to later deploy the products.

2 Architectural Goals

2.1 Orchestration in 5G

The products and services being defined in PPS2 (Product, Process or Service #2) can be organized under 6 main technical areas:

- Policy management and control
- Service definition and orchestration
- Service assurance
- Support mechanisms (e.g. DNS)
- Security
- Virtualization platforms

Such technical areas reflect the interest of each partner in PPS2, both in terms of business and research goals. The products will either 1) partially or fully implement specific 5G networking functions, potentially extending them with additional capabilities, or 2) interact with 5G standard functions, coexisting in a 5G ecosystem but whose operation goes beyond that of mobile domains. This means that a close alignment with 3GPP (3rd Generation Partnership Project) 5G system architecture must exist.

The 3GPP 5G System Architecture Phase 1, Stage 3 specification [1] was completed in June 2018 (3GPP Release 15). The completion of the Standalone specification, which complements the Non-Standalone specification released in December 2017, not only gives 5G New Radio the ability of independent deployment but also brings a brand new end-to-end network architecture. The specification defines the overall architecture model and principles, support of broadband data services, subscriber authentication and service usage authorization, application support in general but also specific support for applications closer to the radio using edge computing. Its support for 3GPP's IP Multimedia Subsystem includes also emergency and regulatory services specifics. Further, the 5G system architecture model uniformly enables user services with different access systems, like fixed network access or interworked WLAN (Wireless Local Area Network), from the onset. The system architecture provides interworking with and migration from 4G, network capability exposure and numerous other functionalities.

Service-Based Architecture Support

Compared to the previous generations, the 3GPP 5G system architecture, depicted in next figure, is service based (SBA - Service Based Architecture). That means wherever suitable the architectural elements are defined as network functions that offer their services via interfaces of a common framework to any network functions that are permitted to make use of these provided services. Network Repository Functions (NRF) allow every network function to discover the services offered by other network functions. This architectural model, which further adopts principles like modularity, reusability and self-containment of network functions, is chosen to enable deployments to take advantage of the latest virtualization and software technologies. The service-based architecture depicts those service-based principles by showing the network functions, primarily Core Network functions, with a single interconnect to the rest of the system, as specified TS 23.501 [1]. In the context of PPS2, the service-based architecture is at the core of product and service design.

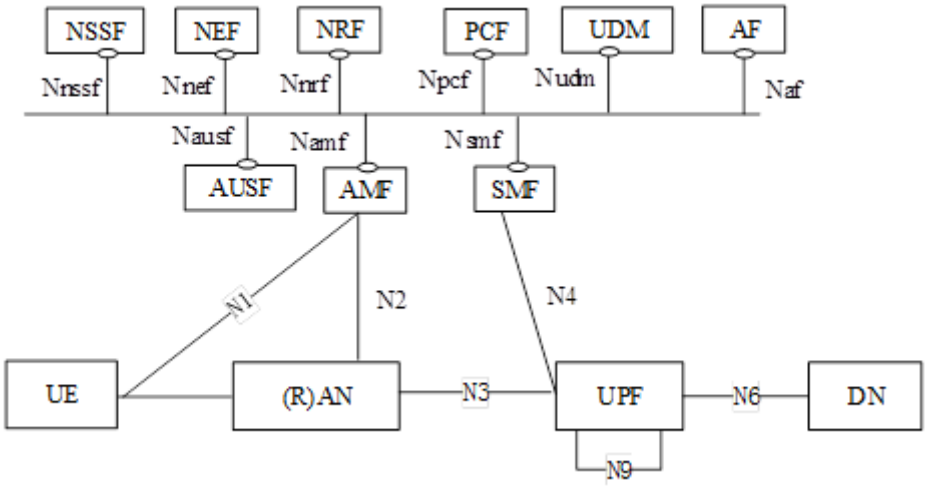

Figure 1 – 5G System architecture

2.2 Assurance impact on the infrastructure

With 5G promises of 100 times more area capacity, 10 to 100 times more connected objects, 10 times more responsive, assurance will need to operate at the speed necessary for whatever process it is serving. For most 5G use cases this means it must be real-time, and based on real-time reference data and correlation. Since virtualized networks are by definition dynamic, it is critical that topology and inventory are updated in real time, and that the correlation or association of VNFs with infrastructure, flows and other dynamic mappings are resolved accurately and quickly.

Service assurance systems will need to adapt to a more rapid pace of innovation, and to a hybrid environment of physical and virtual resources. Operators will need to continue to be disciplined in their testing, leveraging the increased visibility and analytics capabilities of next-generation service assurance and monitoring systems. Because physical and virtual resources will be highly distributed to provide the quality of experience required, it will be critical to have an end-to-end view.

Cisco's Global Cloud Index [2] estimates that global data center traffic was 6.8 zettabytes in 2016, and will grow to 20.6 zettabytes in 2021. This will force data center operators to not only dramatically scale their data center networking infrastructure but do so quickly and cost-effectively.

Beyond the physical layer, the high demands on data center networks also mean that sophisticated analytics are necessary for end-to-end verification and service assurance in a new environment of complex NFV and emerging 5G requirements.

2.3 Security impact on the infrastructure

The evolution from 4G to 5G brings several high profile changes to the way operator's core network is managed. The greatest impact from the security standpoint is the introduction of mechanisms capable of adapting to the inherent flexibility of a 5G network. The introduction of NFV technology as a driver for the deployment and management of network elements allows the operator to allocate its resources in a flexible manner, enabling network elements to be constantly adjusting to the provided service requirements. However, this increase in flexibility requires an increased connectivity between different segments of the operator's network, and some segments that were previously isolated from the Internet may now require connectivity to the Internet. That scenario, on a traditional network, would certainly be seen as a security issue. This extended connectivity of operator's network is nevertheless

a required driver for the implementation of NFV, especially in scenarios with multiple VIM or POP under the control of a single MANO (Management and Orchestration) system.

Common malware at the end client's network is usually of no concern to the operator's network, and its traffic can be regarded as usual traffic. However, if the amount or type of traffic impacts the operator's ability to supply its portfolio services, it will be mandatory to detect and disrupt malware generated traffic, for instance, by stopping a specific malware from communicating with its control system in order to stop a running DDoS (Distributed Denial of Service) attack. This kind of activities, performed at an operator level, have the primary goal of stopping the increase of traffic generated by this type of attack from impacting the quality of the operator's supplied services. In some cases, these protective activities may be performed in a cooperative way by multiple operators, usually coordinated by a CERT (Computer Emergency Response Team).

2.4 PPS2 general architecture

Figure below shows the general architecture for PPS2, where the various components are organized in three layers. The first layer represents applications deployed by a network operator to enable supporting services, such as security related functions or name resolution, while the third layer is where the 5G Core functions, specified by 3GPP, are located. Finally, the management layer, consists of operation support systems that provide all the required functionalities, excluding business related, for network operators to deliver services. These systems are also responsible for the management of the functions that are part of the two layers mentioned above.

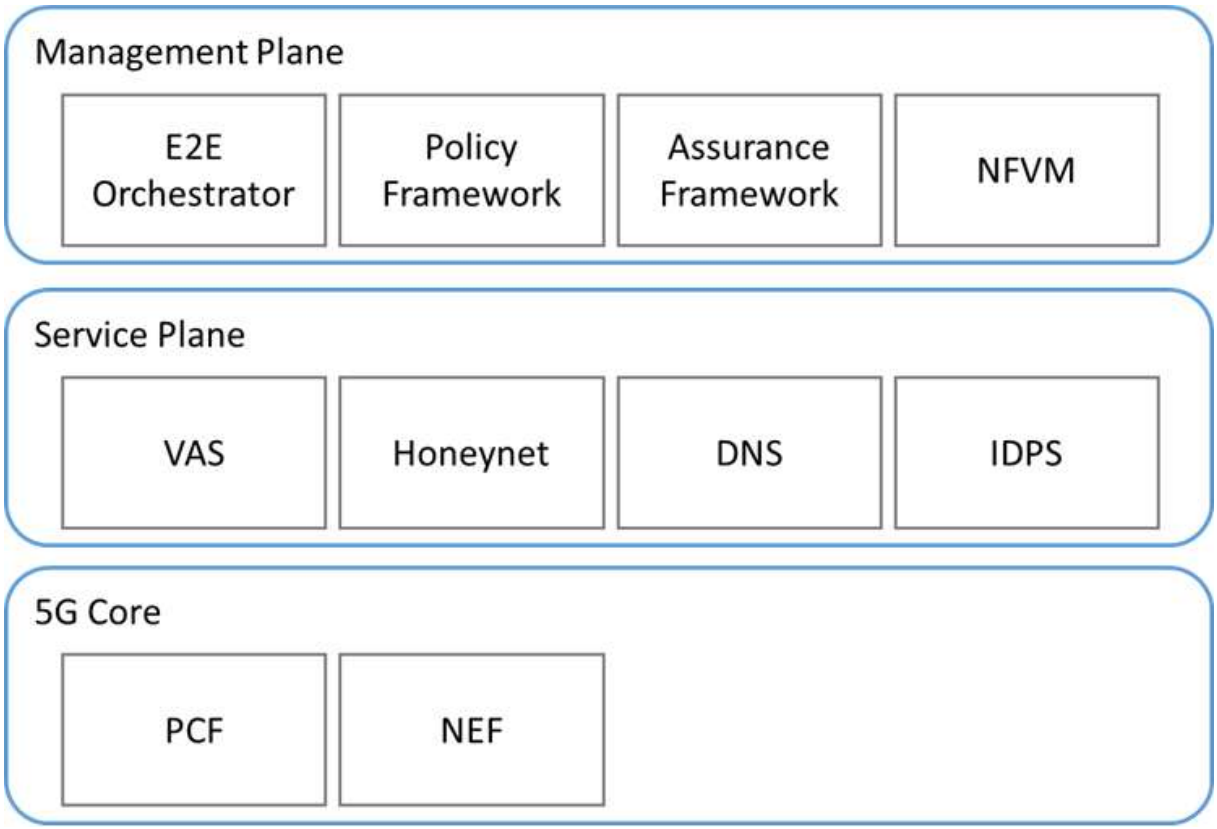

Figure 2 – PPS2 Architecture organized by functional planes

The following is a list of all the components shown above with a brief description of their functional role:

Service Plane

- **Vulnerability Assessment System** (VAS) − security related function that provides an identification of potential security threats.
- **Honeynet** − security related function to mitigate attacks based on lateral movement between virtual resources.
- **Domain Name System** (DNS) − supporting system that translates domain names into IP addresses.
- **Intrusion Detection and Prevention System** (IDPS) − security related function to monitor networks and systems for malicious activity.

Management Plane

- **E2E** (End-to-End) **Orchestrator** − management system that exposes automatized operations on top of services and resources.
- **Policy Framework** − management system that provides decision making capabilities at a service level.
- **Assurance Framework** − management system that enables monitoring and analytics on top of services and resources.
- **Network Function Virtualization Manager** (NFVM) − management system responsible for the lifecycle management of network functions.

5G Core

- **Policy Control Function** (PCF) − 5G Core function specified by 3GPP realizing a policy framework to govern network behavior.
- **Network Exposure Function** (NEF) − 5G Core function specified by 3GPP that exposes capabilities, events and information to other functions.

Regarding the interfaces or the relationship between them, section 3 presents the external interfaces for each component and its respective characterization.

2.5 Requirements for the 5G Core

This section describes the compliance of the proposed solutions implementation against the service requirements for the 5G core system.

2.5.1 5G general requirements

5G has three main areas boundering its requirements:

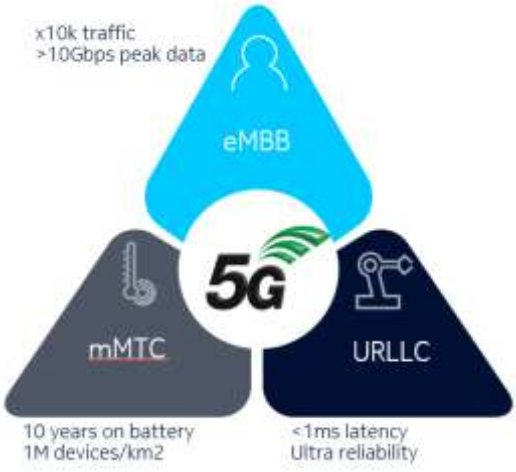

Figure 3 – 5G main areas of requirements

- **Enhanced Mobile Broad Band (eMBB)**
  - 5G is demanded to support higher bitrates and traffic volume.
  - 5G is required to provide wireless communications with more than 10Gbps peak datarate and 10 times current traffic volume.
- **Ultra-Reliable and Low Latency Communications (URLLC)**
  - 5G needs to communicate with very high resiliency.
  - 5G is required to provide less than 1ms latency.
- **Massive machine type communication (mMTC)**
  - 5G expects a massification of the connected devices and expects that they won't need much bandwidth and mobility.
  - 5G is required to provide the means by which devices may have a 10 year battery life expectation and a device density of 1M devices/Km2.

The extreme mobile broadband speed and traffic requirements are already possible to be met by using the new 5G NR connected to a 4G EPC (Evolved Packet Core) and EUTRAN (NSA - Non StandAlone - deployment).

The new 5G Core will pave the way for the other two demanding targets: the massive machine communication and the critical machine communication.

2.5.2 Project specific requirements

The 3GPP TS 22.261 [3] defines the service requirements for the 5G system.

Based on PPS2 scope, only a subset of this Technical Specifications are to be accomplished by this architecture solution.

This document has been revised. The result of this analysis can be found at the Annex 8.1. What the project, and PPS2 in particular, plans to accomplish has been highlighted in green, what is not expected to be fulfilled has been highlighted in red.

Assumptions for the TS analysis:

- This project is not supposed to support 5G migration scenarios.
- This project is a pure 5G demo. It is not supposed to support legacy service support or any transitory scenarios (e.g. NSA).
- There will be no shared 5G. This will be a 5G standalone project with limited inter access technologies (WLAN access, fixed broadband access).
- There will be 2 fixed network slices. There will be no functionality to perform any LCM (LifeCycle Management) on slices. There will be no cross-network slice coordination, apart from the one inherent to the technology.
- Satellite access will not be supported.
- Application server located outside the operator's network will not be supported. They will be considered as being at a 5G DN (Data Network), with no interaction with the 5GCN.
- Virtual networks and MVNOs (Multi Virtual Network Operators) will not be supported.
- Only HPLMN (Home Public Land Mobile Network) scenarios will be supported.
- No roaming scenarios are to be supported.
- Limited network capability exposure to 3rd party entities.
- Self back-haul, extreme long range coverage scenarios will not be considered in this 5G core document.
- Multi-network connectivity and service delivery across operators will not be considered.
- NG-RAN (Next Generation Radio Access Network) sharing scenarios will not be supported.
- The markets requiring minimal service levels and extreme long range coverage in low density areas scenarios are not going to be supported in this 1st approach.
- Multiple core networks sharing will not be supported.
- The Ethernet transport services support, the 5G LAN/5G LAN-VN, the higher-accuracy positioning were left for the 5G radio team to check.
- The audio-visual interaction like the VR is not considered in this project.
- 5G security scenarios were kept to its minimum. No change of ownership during time is yet to be considered (dynamically establish or refresh credentials and subscriptions).
- The charging aspects were considered to be out of scope of this project.

3 Functional View

This chapter describes the design and implementation of functional elements, their responsibilities, interfaces, and primary interactions of the several architecture parts.

3.1 Policy Framework

The Policy Framework (PF) is an environment where policies are managed and decisions based on those policies are made.

At the highest level, the PF comprises two basic components: The Policy Administration Point (PAP), which is responsible for the management of policies, and the Policy Decision Points (PDP), where these policies are active and actual policy-based decisions are made.

For this project, two different instances of PF will exist:

- A 5G Network Policer, which will be the framework in charge of managing and applying policies at the service/session level, which only PDP will be the **PCF** (Policy Control Function) as defined by 3GPP.
- A High-Level Policer, which will be the framework in charge of managing and applying policies at operations management level, which only PDP will be the **HLPolicer**, an engine accepting inputs from analytics systems and determine the actions to be performed by the solution orchestrator.

3.1.1 External View

The next diagram represents the external systems and how they interact with the Policy Framework when acting as PCF as described by 3GPP standards, on a general 5G environment.

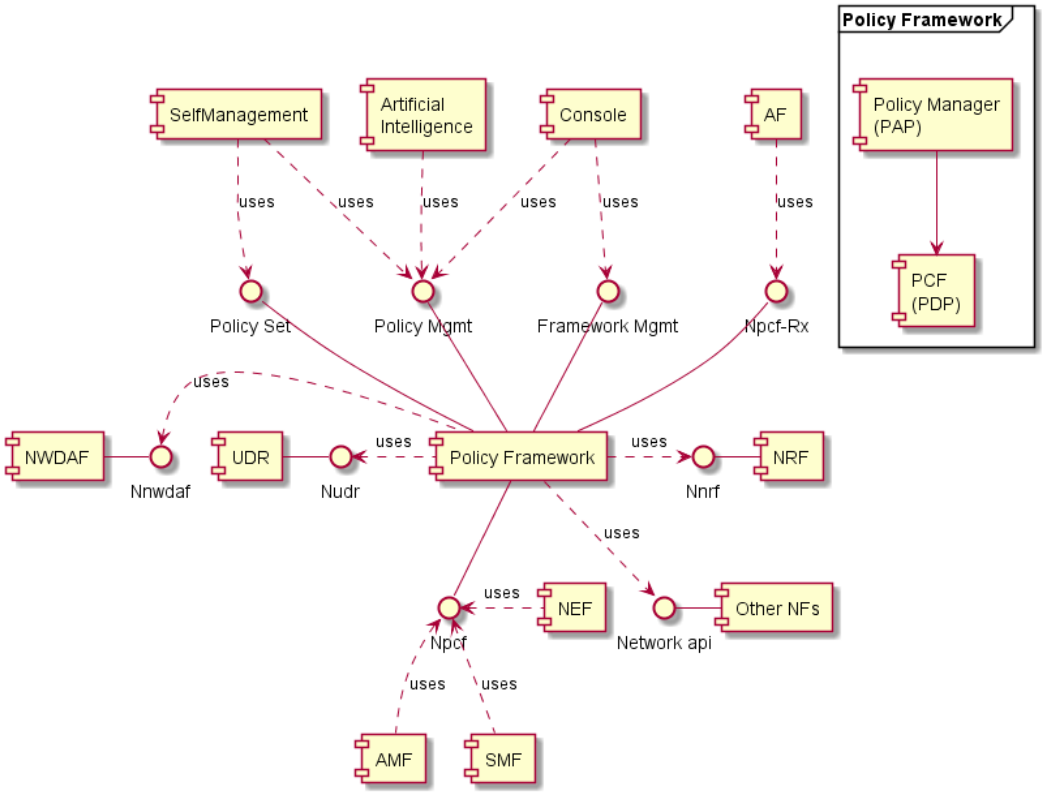

Figure 4 – Policy Framework external systems - PCF

The following diagram represents the interactions of the Policy Framework with the surrounding ecosystem when it is playing the particular role of HLPolicer, in the scope of this project.

In this case, the interactions are considered for a generic, simplified, policy framework meant to enforce simple rules, after the verification of a certain Event against a number of simple, static, conditions. This framework will be put in place to provide means to "translate" Events generated by various analysis processes into concrete actions, when out of the scope of Service Sessions.

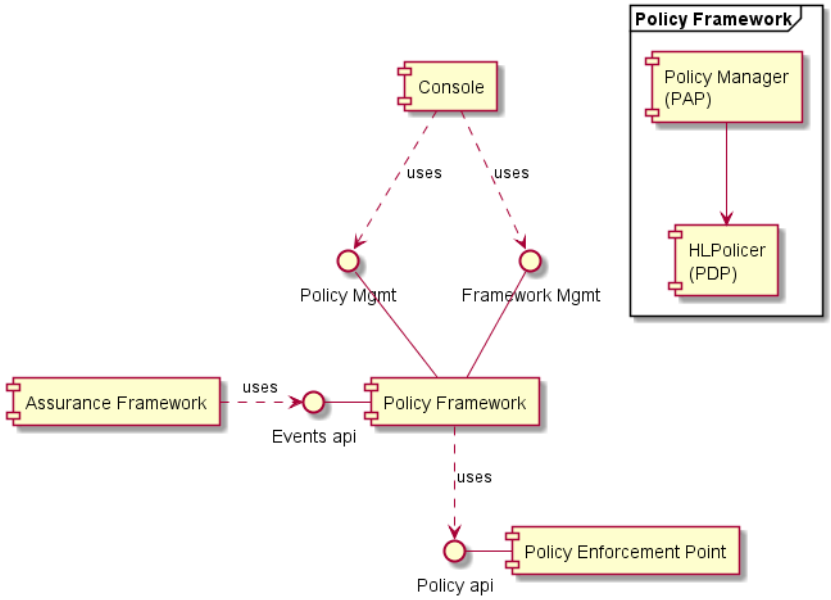

Figure 5 – Policy Framework external systems - HLPolicer

3.1.2 Internal View

Next is shown the high-level internal view of the policy framework.

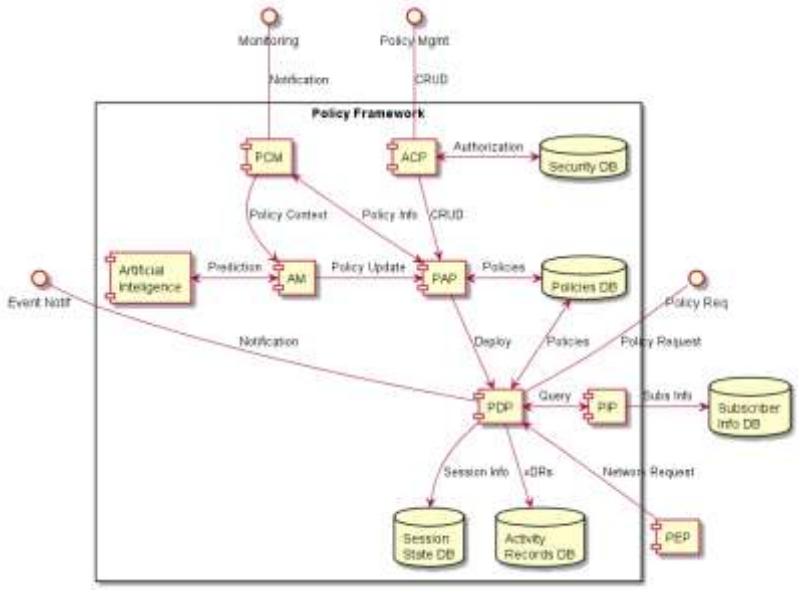

Figure 6 – Policy Framework internal view

In this internal view the following functional processes are considered as part of the Policy Framework:

Table 1 – Policy Framework components

Component	Component Description
PAP	**Policy Administration Point –** Central point for administration of policies. It is responsible for the life-cycle of the policies from the creation to the deletion including all modifications. Associated to the management of the life-cycle one very important component is the validation and verification of the created/updated policies. The PAP is the component responsible for inserting/updating policies (imperative policies) on the Policy DB (DataBase). Every time the PAP creates/updates policies on the Policy DB it also informs the several PDPs of the deployment of those changes. The PAP is also responsible for the translation of declarative policies that can be used by analytics or other external systems into the imperative policies that are used by the PDP on their decision process.
PDP	**Policy Decision Point –** Evaluates requests from the PEP (Policy Enforcement Point) and computes a decision based on the event information, the configured policies and any additional external info that may be used in the decision process. The PDP reads the configured policies from the Policies DB and uses PIP to retrieve information from external data sources (Info DB).
PIP	**Policy Information Point –** The system entity that acts as gateway to the several external data sources that might be used by the PDP on its decision process.
ACP	**Access Control Point –** Performs the management of policy management API (Application Programming Interface) clients' authorization and serves as bus to the PAP CRUD (Create/Read/Update/Delete) functions.
PCM	**Policy Context Manager –** Computes policy context retrieving from PAP the information related to active policies and delivers to AM (Autonomic Manager) that context and any changes that happen to the policy contexts.

Component	Component Description
AM	**Autonomic Manager −** AM uses a context-aware policy model [RFC 8328] [4]. When context changes, AM and associated Artificial intelligent component use machine reasoning to generate hypotheses and propose policies update. This way, AM adjusts the set of policies that are being used to control the service delivery.
AI (Artificial Intelligence)	• **IM −** Information Model: representation of the characteristics and behavior. Uses a simplified abstract view. • **Analytics −** Normalizes and evaluates data and events in search of new characteristics, behavior or distortions. • **RLP −** Reasoning, Learning and Prediction: uses machine reasoning to generate hypotheses when a problem occurs and machine learning to reinforce actions taken. These actions result in the identification of models to generate policies that can execute more generalized behavior.
Session State DB	This component is used to persist session related information during each session.
Activity Records DB	This component is used to store the records which detail the activity of the Policy Framework.
Security DB	This component is used to persist API client information for authentication purposes.

Other relevant functional processes that are outside the Policy Framework are summarized in next table.

Table 2 – Outside Policy Framework components

Component	Component Description
PEP	Policy Enforcement Point: Controls access to a given resource. When a request is detected, it queries the PDP to obtain the access requirements decision (i.e. access to the resource is approved or rejected and what are the access conditions), and acts upon the received decision parameters to enforce a required/specified service delivery ecosystem.
Info DBs	These are databases that are outside of the Policy Framework but contain information that is vital for the decision-making process of the PDP. This information can be of various types (e.g.: subscription info; network inventory; service inventory; service catalog, etc.) and is accessed via PIP.

3.1.3 Interfaces

The 3GPP defines two different approaches for 5G architecture in the TS 23.501 [5] specification: Service-Based representation and a Reference Point representation. The External view representation in section 3.1.1 provides an hybrid approach where we include a service view (what service is being delivered to whom) and still keep some of the functional approach that defines the reference points between functions.

Despite any representation differences, for standard interfaces the Standard determinations will always be honored.

To the South, the Policy Framework will connect to AMF (Access Mobility Function), SMF (Session Management Function) using the interfaces exposed by these entities and will expose the Npcf service itself (Figure 4).

To the North, the Policy Framework will expose its management functions, mostly related to the management of Policy entities (CRUD - Create/Read/Update/Delete) and to the management of the framework itself (Figure 4).

Also northbound are the interfaces that will allow external functions like AF (Application Function) to use.

From the side interfaces (East/West), the Policy Framework will receive from the NWDAF (Network Data Analytics Function) the Events that will trigger PCF Policies, will read and update User Data from/to the UDR (Unified Data Repository - Subscriber Info DB in the 5G policy control architecture [3GPP TS 23.503 [6]), and register its services to the NRF.

In the scenario of HLPolicer, the relevant interfaces are the east-bound event notification API to be used by the 5Go Assurance Framework and the south-bound generic PEP interface to be used by the Policy Framework to contact the e2e 5Go orchestrator.

The table below summarizes the interfaces used/exposed by the Policy Framework (PF):

Table 3 – Policy Framework interface components

IF ID	Interface Description	Components	Type of interface	Exposure interface (Internal/External; Mandatory/Optional)
#1	Activation of Policies exposed by the framework. CRUD (Create/Read/Update/Delete) of policy entities as well as the mechanisms that support them, like Policy verification and validation	Policy Framework ↔ SelfManagement	REST	External Mandatory
#2	CRUD of policy entities as well as the mechanisms that support them, like Policy verification and validation	Policy Framework ↔ Artificial Intelligence	REST	External Mandatory
#3	Activation of Policies exposed by the framework. CRUD of policy entities as well as the mechanisms that support them, like Policy verification and validation Platform Management. Namely FCAPS (Fault, Configuration, Accounting, Performance, Security).	Policy Framework ↔ Console	REST, Others for Management (SNMP, Netconf)	External Mandatory
#5	[3GPP TS 23.503] [6] Corresponds to Ref. point N5 and Rx	Policy Framework ↔ AF (Application Function)	REST, Diameter	External Mandatory
#6	[3GPP TS 23.503] [6], [3GPP TS 29.510] [7] Interface towards FRF in 5G core	Policy Framework ↔ NRF (Network Repository Function)	REST	External Mandatory
#7	[3GPP TS 23.503] [6] Corresponds to Ref. point N30	Policy Framework ↔ NEF (Network Exposure Function)	REST	External Mandatory
#8	[3GPP TS 23.503] [6] Corresponds to Ref. point N7	Policy Framework ↔ SMF (Session Management Function)	REST	External Mandatory
#9	[3GPP TS 23.503] [6] Corresponds to Ref. point N15	Policy Framework ↔ AMF (Access and Mobility Management Function)	REST	External Mandatory
#10	Interfacing with other service platforms in the scope of service sessions	Policy Framework ↔ Other Platforms featuring Service / Session Control	REST, Other	External Optional
#11	[3GPP TS 23.503] [6] Corresponds to Ref. point N25	Policy Framework ↔ UDR (Unified Data Repository)	REST	External Mandatory

IF ID	Interface Description	Components	Type of interface	Exposure interface (Internal/External; Mandatory/Optional)
#12	[3GPP TS 23.503] [6] Corresponds to Ref. point N23	Policy Framework ↔ NWDAF (Network Data Analytics Function)	REST	External Mandatory
#13	Interface for pushing policies to non-specific Policy Enforcement Points	Policy Framework ↔ Generic PEP (Policy Enforcement Point)	REST	External Optional
#14	Interface for conveying Events from Analytics Functions	Policy Framework ↔ Assurance Framework	REST	External Optional

3.2 Orchestration Framework

The role of Orchestration components in 5G network environments remains unchanged with respect to legacy network environments. This role is two-folded and comprises the following:

- **Service Order Management** – a service order is an operation that is executed during the fulfillment process and involves creating, modifying and terminating services.
- **Resource Order Management** – a resource order is an operation that is executed on top of resources, and involves the creation, retrieval, modification and termination of resources.

Nevertheless, dealing with multiple virtualization layers for a single or multiple services adds complexity to what was already a complex infrastructure composed by a myriad of technologies that must be integrated to realize end-to-end services. Thus, in the NFV era, its successful integration and validation in 5G networks is an essential building block towards realizing a cloud-native mobile network, and ultimately obtaining the much-desired reduction in CAPEX (Capital Expenditure) and OPEX (Operational Expenditure).

3.2.1 Milestone 2.2 - Selection of the Orchestration Platform

Open source networking projects are becoming increasingly relevant for the Telecom industry, which has led to the involvement of both Telecom Operators, traditional network vendors, software companies and integrators in foundations such as Linux or Apache, or dedicated initiatives such as OpenStack foundation. There are several projects addressing network service Orchestration. To choose the Orchestration platform to be used in this project, the following were considered:

- Open-Source MANO
- ManageIQ
- Aria / Cloudify
- Tacker
- ONAP – Master Service Orchestrator
- OpenBaton
- OpenCORD – XOS

The parameters listed below define some of the criteria to make the choice for Orchestration platform:

- **NBI: NS Lifecycle Management** – existence of NBI (Northbound Interface) for managing NSs
- **NBI: VNF Lifecycle Management** – existence of NBI for managing NFVs.

- **NBI: Configuration Management** – existence of NBI for managing application configuration processes
- **NBI: Catalog Management** – existence of NBI for managing the catalog and its content
- **NBI: Inventory** – existence of NBI for accessing inventory information
- **NBI: Monitoring** – existence of NBI for accessing NSs metrics
- **Openstack Version** – supported Openstack version(s)
- **Openstack Interface** – type of available interface for interacting with Openstack
- **EPA (Enhanced Platform Awareness)** – ability to characterize DCs (Data Centers) through specific features (e.g. DPDK [Data Plane Development Kit], SR-IOV)
- **SFC (Service Function Chaining)** – NFVO (Network Function Virtualization Orchestrator) support for managing and configuring NSs with SFC
- **Support for containers** – support for containers utilization
- **Multi-VIM** – support for NSs/VNFs instantiation in multiple DCs
- **NSs/VNFs Management Type** – whether the NSs/VNFs management is realized through automated or manual processes (or both)
- **VNF Integration Requirements** – specific requirements for VNF integration in NFVO (e.g. agents, OSs)
- **Multi-VDU VNFs** – support for instantiation of multi-VDU (irtual Deployment Unit) VNFs
- **Supported SDN Controllers** – identification of SDN (Software Defined Network) controllers supported by the NFVO
- **Support for WAN Management** – support for WAN network resource management
- **Integrated VNF Manager** – whether the NFVO embeds a VNFM
- **External VNF Manager Support** – NFVO support for using external VNFM (ex: due to NS which requires specific VNFM)
- **Integrated Monitoring** – whether the NFVO has its own monitoring system
- **Policy-based Management** – whether the NFVO supports the definition of NSs/VNFs management policies (e.g. for threshold-based scaling)
- **NSs/VNFs Catalog** – whether the NFVO has NSs/VNFs catalog
- **Descriptor Language** – languages supported for NSs/VNFs description
- **License** – NFVO license (e.g. Apache 2.0)
- **Release** – which NFVO version does is characterized by the table
- **Available Documentation** – amount and quality of online documentation of the solution
- **Association with other projects** – association and/or utilization of the NFVO by other projects / fora / initiatives of interest

It should be noted that the previous list does not reflect the weight of each parameter when making the final choice. So, based on this criterion, ONAP was the chosen platform to be used on the project.

The following sections already reflect the choice made.

3.2.2 External View

The figure below shows the external view of the 5G Service Orchestrator component, highlighting the external systems it directly interacts with. Each external system may interact with the 5G Service Orchestrator by more than one interface.

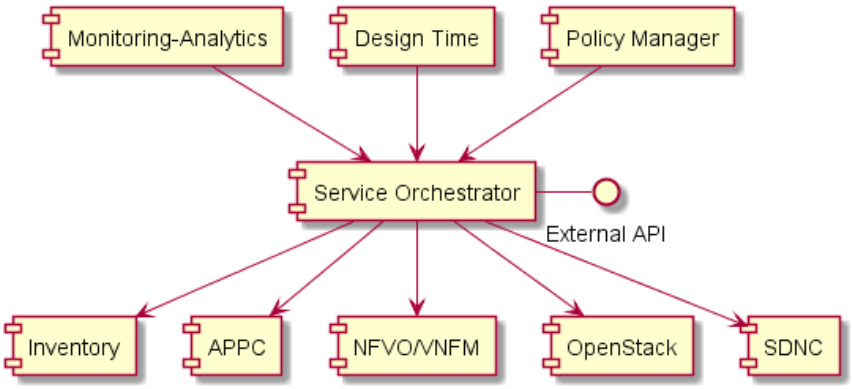

Figure 7 – External view of 5G Service Orchestrator component

It should be noted that the Service Orchestrator shown above and the predicted interactions with external systems are aligned with the functional role of the Service Orchestrator present in ONAP.

From a system-to-system perspective, the foreseen interactions are listed below.

- **Monitoring-Analytics** – interactions to exchange service and resource metrics or to request an operation on a specific service as a result of an analytics process;
- **Design Time** – the Design Time notation represents systems that participate on the design of services and thus the interface with the Service Orchestrator is used to on board services, resources and associated operations;
- **Policy Manager** – during run-time this interface is used by the Policy Manager to request an operation (or more) on specific services, thus enabling the Service Orchestrator as a Policy Enforcement Point (PEP) from the Policy Manager perspective;
- **External API** – the External API represents all systems that may use the Service Orchestrator Order Management interface to request an operation on a Service or an infrastructure resource
- **Inventory** – the Inventory represents systems that persist instantiation and runtime information regarding services or resources, thus this connection will be used by the Service Orchestrator to store new data as a result of an operation on a service or resource;
- **Application Controller (APPC)** – although the APPC is seen as an external system to the Service Orchestrator, it is a component that is part of it from a functional perspective. The APPC provides the Service Orchestrator with the ability to perform configuration operations (integrating configuration management tools such as Ansible or Chef);
- **Network Functions Virtualization Orchestrator (NFVO) & Virtual Network Function Manager (VNFM)** – although the NFVO and VNFM are seen as external systems to the Service Orchestrator, they are components which are part of it from a functional perspective. NFVO and VNFM provide the Orchestration capabilities of the ETSI NFV specification, which include the lifecycle management of network services (through NFVO), and lifecycle management of VNFs and the integration of a vendor specific VNF Manager (through a generic VNF Manager);
- **OpenStack** – the Service Orchestrator interacts with OpenStack to perform operations on cloud-based resources such as those that support VNFs;
- **Software Defined Network Controller (SDNC)** – although the SDNC is seen as an external system to the Service Orchestrator, it is a component that is part of it from a functional perspective. The SDNC provides the Service Orchestrator the ability to interact with SDN Controllers or Element Management systems to perform configuration on network elements.

3.2.3 Internal View

The next figure shows the internal view (i.e. architecture) of the 5G Service Orchestrator.

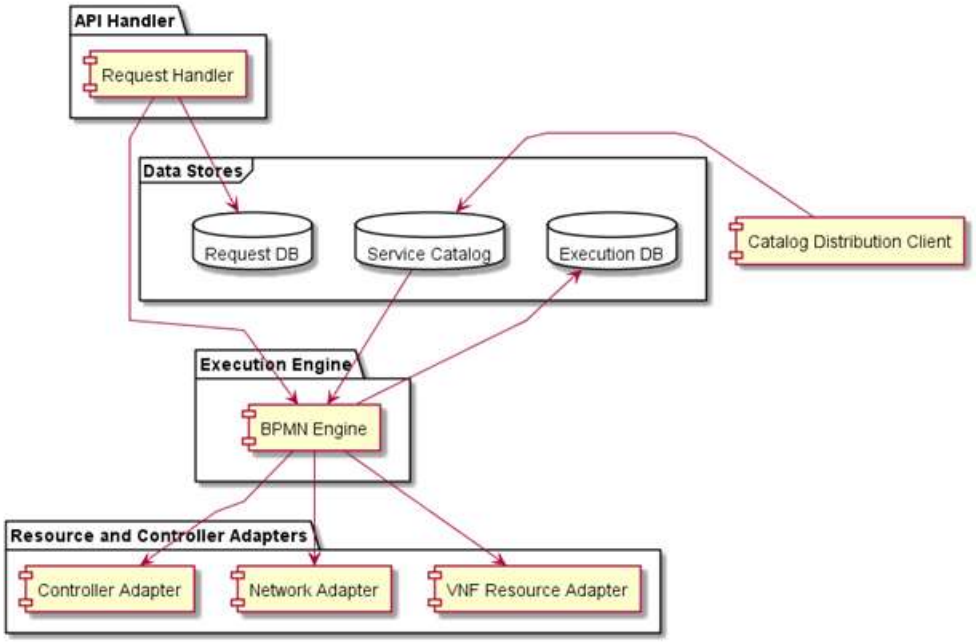

Figure 8 – Internal architecture of 5G Service Orchestrator component

The role of each component comprising its architecture is summarized in next table.

Table 4 – Service Orchestrator components

Component	Component Description
Request Handler	The role of this component is to receive and process request messages on the external API.
Catalog Distribution Client	This component intermediates the connection between Service Orchestrator and a logically centralized catalog, enabling the on boarding of orchestration artifacts on the Service Catalog.
Data Stores	**Request DB** − The role of this component is to persist information about received request messages and their respective status. **Execution DB** − This database holds the state information about workflows being executed on the BPMN Engine. **Service Catalog** − This component's role is to persist service and resource definition, including other relevant and necessary information, for orchestration components to execute operations on services and resources.
Execution Engine	**Business Process Modeling Notation (BPMN) Engine** − This component is the core of the Service Orchestrator and is responsible for putting into action all the necessary individual operations that constitute a complete workflow for an operation performed on a service or resource.
Southbound Adapter Layer	**Virtual Network Function (VNF) Resources Adapter** − This component is part of the southbound adapter layer, which in this case refers to the adapters for the virtualization infrastructure of compute-based components **Controller Adapter** − This component is part of the southbound adapter layer, which in this case refers to the adapters to the various controllers for configuration and life cycle management of components that are used to build services **Network Adapter** − This component is part of the southbound adapter layer, which in this case refers to the adapters for the virtualization infrastructure of network-based components.

To onboard services and respective operations on the 5G Service Orchestrator, it is necessary to use the Catalog Distribution Client, which after receiving a service template will store it (i.e. populate) on the Service Catalog. After this process, it is possible to deploy, terminate or request any other operation on top of a service or resource. The request for a service operation - made through the API - will be received by the request handler and stored on the Request DB, while also invoking the BMPN Engine to execute the operation. The BPMN Engine fetches the workflow that constitutes the service operation from the Service Catalog and during the workflow execution will use the Execution DB to store the necessary information. When the workflow defines an operation on an external system, the BPMN Engine invokes one of the adapters, where the specific adapter is chosen based on the type of resource. The purpose of the Resource and Controller Adapters is to decouple orchestration from implementation and thus, allow the development of both in distinct timeframes while providing an abstraction to the infrastructure implementation, i.e. a service with a firewall does not change if a provider wants to use a firewall from another vendor.

3.2.4 Interfaces

The figure below shows the various interfaces to external systems that are implemented on the 5G Service Orchestrator. Additionally, the technology used to implement the aforementioned interfaces is also shown.

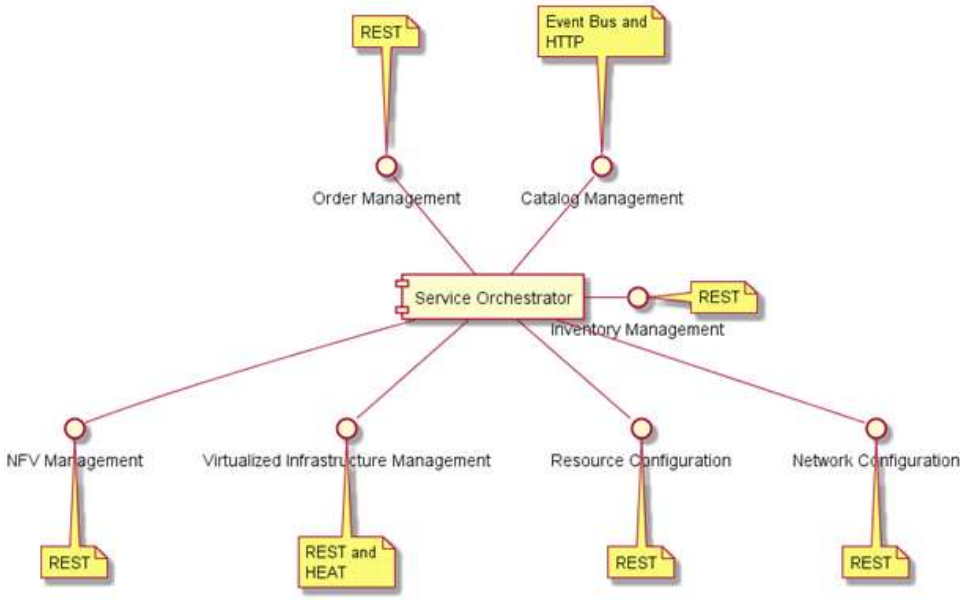

Figure 9 – External interfaces of 5G Service Orchestrator

The following table presents the various interfaces shown on the figure above.

Table 5 – Service Orchestrator interface components

IF ID	Interface Description	Components	Type of Interface	Exposure interface (Internal/External; Mandatory/Optional)

IF ID	Interface Description	Components	Type of Interface	Exposure interface (Internal/External; Mandatory/Optional)
#1	**Order Management** This interface is used to request operations on services or resources, typical operations are: create, update or terminate service. Reference specifications - TMF641, TMF640, MEF Legato, MEF Interlude	Monitoring/Analytics → Service Orchestrator Policy Manager → Service Orchestrator	REST	External Mandatory
#2	**Catalog Management** This interface is used to on board or to update entities stored on the Service Orchestrator internal catalog. Typical operations include: upload new service, update VNF descriptor. Reference specifications - TMF TMF633	Design Time → Service Orchestrator	Kafka, REST	External Optional
#3	**Inventory Management** This interface is used to retrieve or modify entities stored on a centralized inventory system. Typical operations include: update resource, create new resource, get VIM (Virtualized Infrastructure Manager) endpoint. Reference specifications - TMF638, TMF639	Service Orchestrator → Inventory	REST	External Optional
#4	**NFV Management** This interface is used to perform lifecycle management of network services (as defined in ETSI NFV), typical operations are: deploy network service, scale network service, heal VNF. Reference specifications - TMF652, ETSI NFV IFA007, ETSI NFV IFA012, ETSI NFV IFA013	Service Orchestrator → NFVO Service Orchestrator → VNFM	REST	External Mandatory
#5	**Virtualized Infrastructure Management** This interface is used to perform life cycle management of cloud resources. Typical operations include: deploy VM (Virtual Machine), migrate VM. Reference specifications - ETSI NFV IFA005, ETSI NFV IFA006	Service Orchestrator → OpenStack	REST, HEAT	External Mandatory

IF ID	Interface Description	Components	Type of Interface	Exposure interface (Internal/External; Mandatory/Optional)
#6	**Resource Configuration** This interface is used to perform changes on resource configurations. Typical resources include: VMs, Cloud Applications, VNFs (Virtualized Network Functions). Reference specifications - TMF652, ETSI NFV IFA008	Service Orchestrator → APPC	REST	External Optional
#7	**Network Configuration** This interface is used to perform changes on network elements configurations, typical operations include: update access list configuration, add network route. Reference Specifications - ETSI NFV IFA005, ETSI NFV IFA006	Service Orchestrator → SDNC	REST	External Optional

3.3 Assurance Framework

The Assurance Framework , will guarantee that services offered over the 5G network meet the service quality level for an optimal subscriber experience. This chapter describes the functional view of *the Assurance Framework* design and implementation, external view, internal view and interfaces.

3.3.1 External View

This section describes the external view of the Assurance Framework design and implementation; implemented interfaces and dependencies to other components. VNFs component is representing all project VNFs, like NEF, VAS, DNS etc.

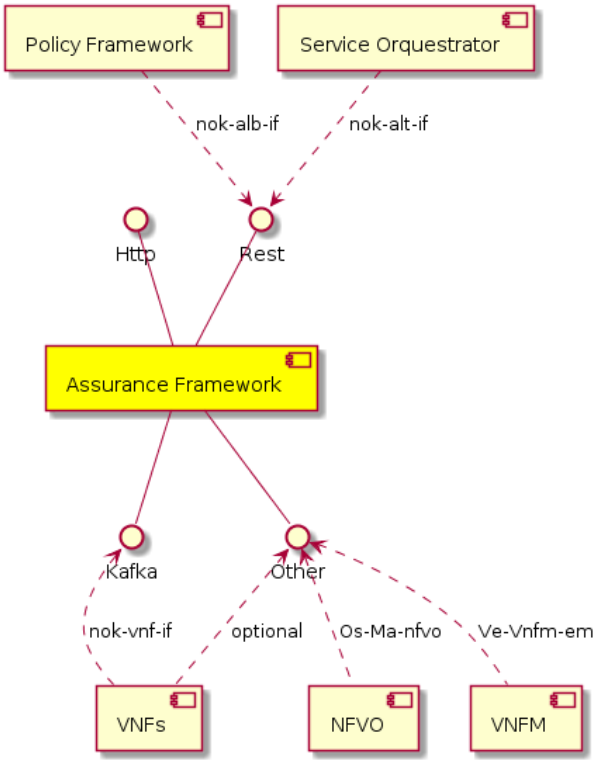

Figure 10 – Assurance Framework - External view

3.3.2 Internal View

This section describes the internal view of *the Assurance Framework* implementation breakdown into components and relationships between them.

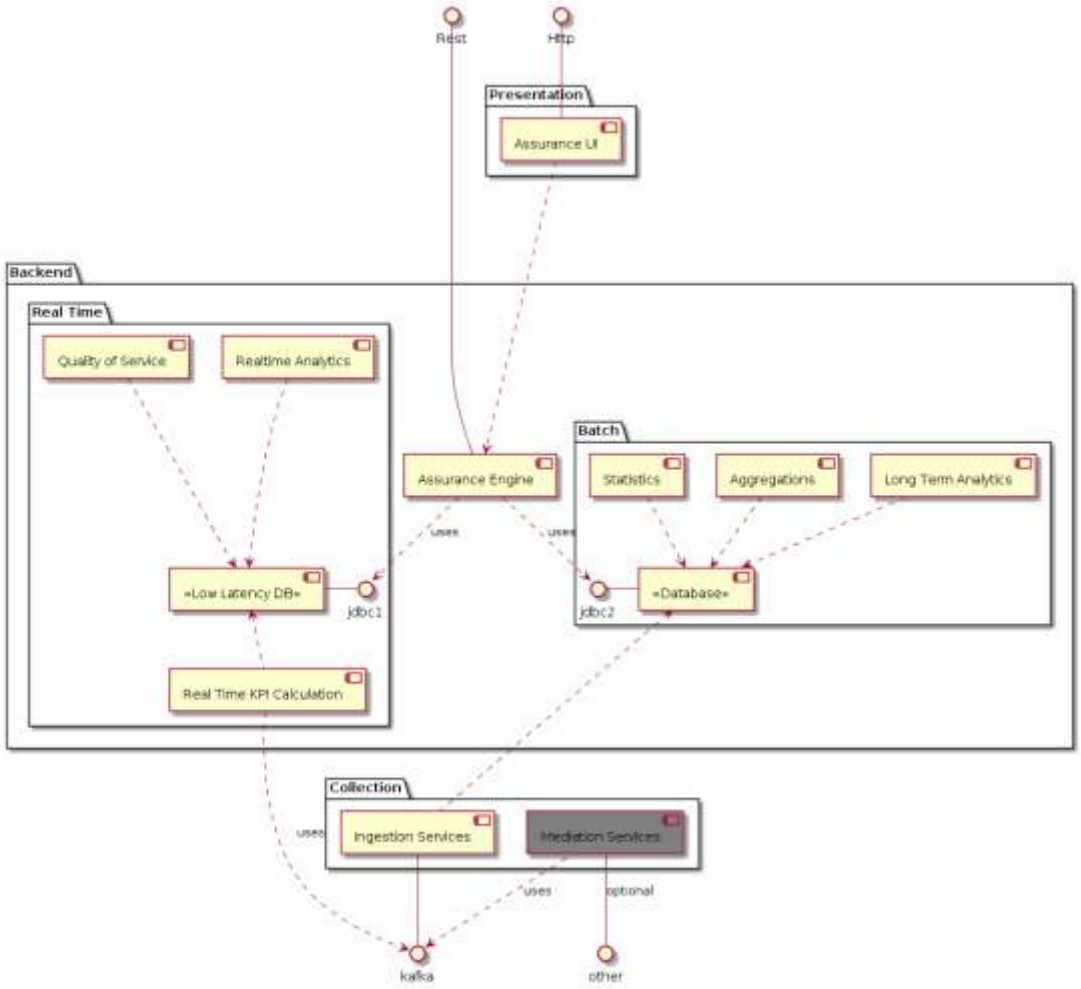

Figure 11 – Assurance Framework - Internal view

The role of each component comprising its architecture is summarized in next table.

Table 6 – Assurance Framework components

Component	Component Description
Assurance UI	Report visualization UI, including report CRUD (Create/Read/Update/Delete); view on PM data in real time; Schedule reports and other tasks; Search KPIs (Key Performance Indicators), reports, counters, measurements, etc. Export / Import report & KPI definitions.
Quality of Service	This service will calculate KQI based on all KPI collected.
Real Time Analytics	This service is responsible for performing analytics in real time.
Real Time KPI Calculation	This service is responsible for calculating KPI in real time through stream-based processing.
Assurance Engine	This service is responsible for running reports and calculating associated KPI.
Statistics	This service will gather statistics on all the collected data.
Aggregations	This service orchestrates summaries calculation, and stores aggregates in database. Handles raw data aggregation in time/object dimension, according to the hierarchies, BH definitions and aggregation rules defined in each adaptation.
Long Term Analytics	This service is responsible for performing analytics in real time in predefined intervals bigger then 1 minute.

Component	Component Description
Ingestion Services	Ingest analytics data from Kafka towards database.
Mediation Services	If some element can't publish directly data into Kafka, it will be possible to integrate with different interfaces and then create adaptations to write data to the Kafka bus. This component is highlighted in gray due to be optional.

3.3.3 Interfaces

This section describes the Assurance Framework used and provided interfaces as summarized in next table.

Table 7 – Assurance Framework interface components

IF ID	Interface Description	Components	Type of Interface	Exposure interface (Internal/External; Mandatory/Optional)
#1	**nok-vnf-if:** Primary interface for VNFs to publish analytic data.	Assurance Framework ↔ VNFs	Kafka	External Mandatory
#2	**nok-vnf-if:** If VNFs can't connect with primary interface using Kafka, other interfaces might be customized for the very specific needs.	Assurance Framework ↔ VNFs	Other	External Optional
#3	**nok-alb-if:** Provides KPI and KQI to help assess if any existing policies are being violated.	Assurance Framework ↔ Policy Framework	Rest	External Mandatory
#4	**nok-alt-if:** Interface for exchanging service and resource metrics or to send information on QoS (Quality of Service) degradation or crossed KPI/KQI.	Assurance Framework ↔ Service Orchestrator	Rest	External Mandatory
#5	**os-Ma-nfvo** - Open Source NFV Management and Orchestration (MANO) reference point is used for exchanges between the Assurance Framework and the NFV Orchestrator (NFVO). Information flowing in this interface will be related to Network Service performance management, fault management and life cycle.	Assurance Framework ↔ NFVO	Rest	External Mandatory
#6	**ve-Vnfm-em** - MANO reference point is used for exchanges between Assurance Framework and VNF Manager. Information flowing in this interface will be related to VNF lifecycle, performance fault and configuration management.	Assurance Framework ↔ NFVM	Rest	External Mandatory

3.4 Support Framework

The Support Framework includes the services/products that are required by other services to perform their function. In this sense, the Support Framework includes services like the DNS, which is a required service for services operating over 5G and the Network Exposure Function (NEF), which is a service in the 5G core that is relevant to allow the communication of QoS requirements.

3.4.1 External Views

DNS

The DNS service includes APIs for the regular DNS clients (e.g. AFs and NFs) to gather information regarding the endpoints of other services (i.e. the IP addresses). From the operator's perspective, the management of the DNS information is crucial to allow dynamic configuration of DNS records. Having this in mind, the DNS Manager (or DNSaaS [DNS as a Service] Management API) allows to configure the DNS records which are used by the DNS (DNS block) to provide the required information to the DNS clients. The interface with the AppIDPS enables the analysis of the DNS requests/replies and the DNS Manager requests to assure that the DNS records are not tampered.

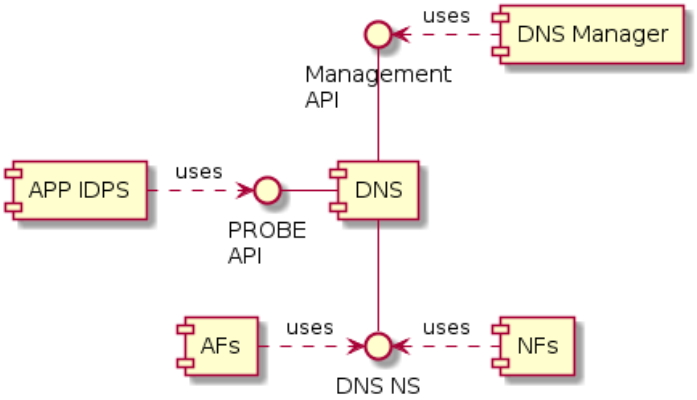

Figure 12 – DNS - External view

NEF

NEF is a service that is part of the 5G core. The diagram illustrated below highlights the connections with other services. For instance, products/services like the Bodykit (e.g. acting as a Application Function) interact with NEF to request the configuration of the 5G network for flows with specific QoS requirements. Consequently, NEF interacts with other 5G core services like PCF, NRF to assure the configuration of the 5G network.

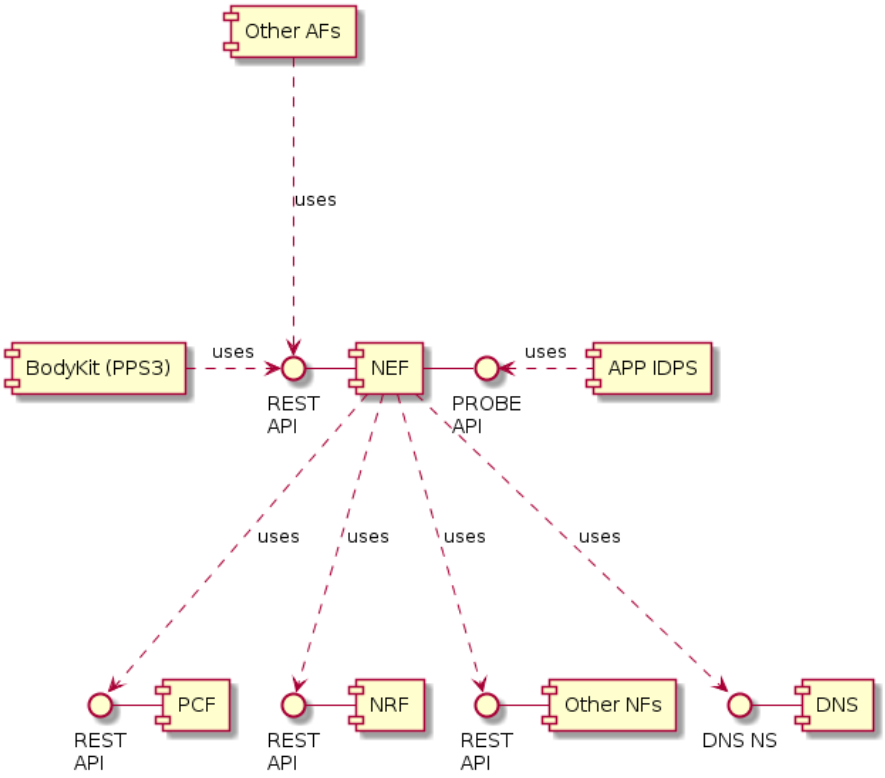

Figure 13 – NEF - External view

3.4.2 Internal Views

DNS

This subsection presents the internal view of the DNS component, as pictured in next figure. The yellow boxes correspond to the internal components of DNS whereas the grey boxes are external to the DNS but are required to assure its functions. The DNS manager (pictured in the external view) corresponds to the DNSaaS component, which includes an API and a management component. The management information is kept on the DNSaaS database.

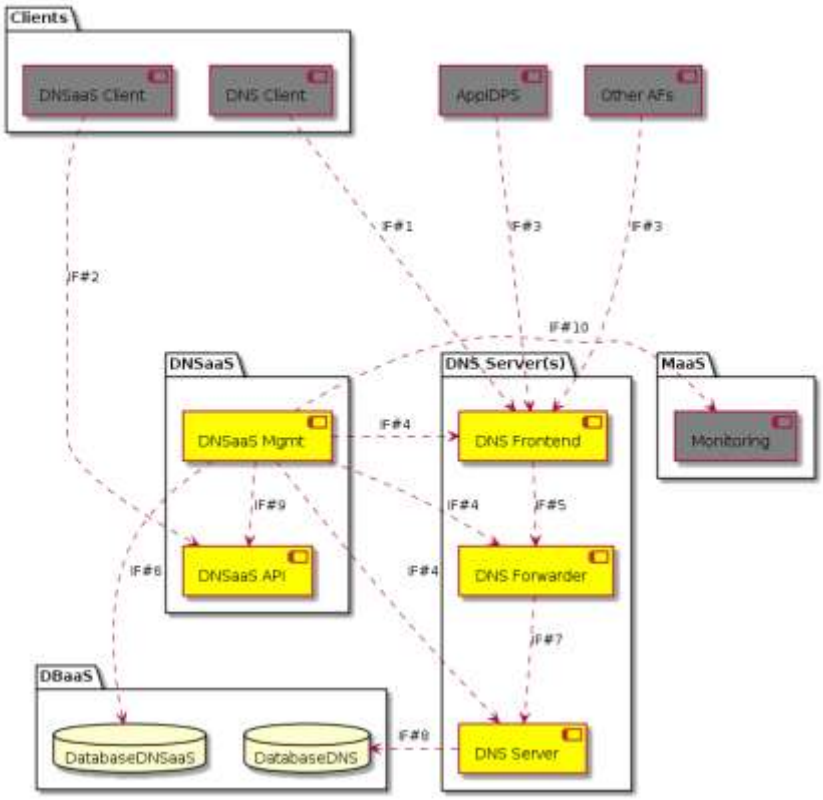

Figure 14 – DNS - Internal view

The components of the DNS are described in the table below.

Table 8 – DNS components

Component	Component Description
DNS Frontend	Receives the DNS requests from DNS Clients, for instance, who has the IP address for the name www.google.pt ?
DNS Forwarder	Is a component of the DNS to assure Load Balancing and high availability. This component receives the requests from the DNS Frontend and forwards them to the pool of DNS servers.
DNS Server(s)	Component responsible for performing the name resolution. This component has all the information regarding the mapping of names to IP addresses. The DNS record information is kept on the DNS databases.
DNSaaS Mgmt	Component responsible for the management of DNS product.
DNSaaS API	Component that allows applications to manage the DNS information, for instance to perform CRUD operations regarding DNS records.

NEF

This subsection presents the interval view of the NEF component. The components are represented in the figure below, where yellow components correspond to the internal components of NEF, while the grey components are external to NEF.

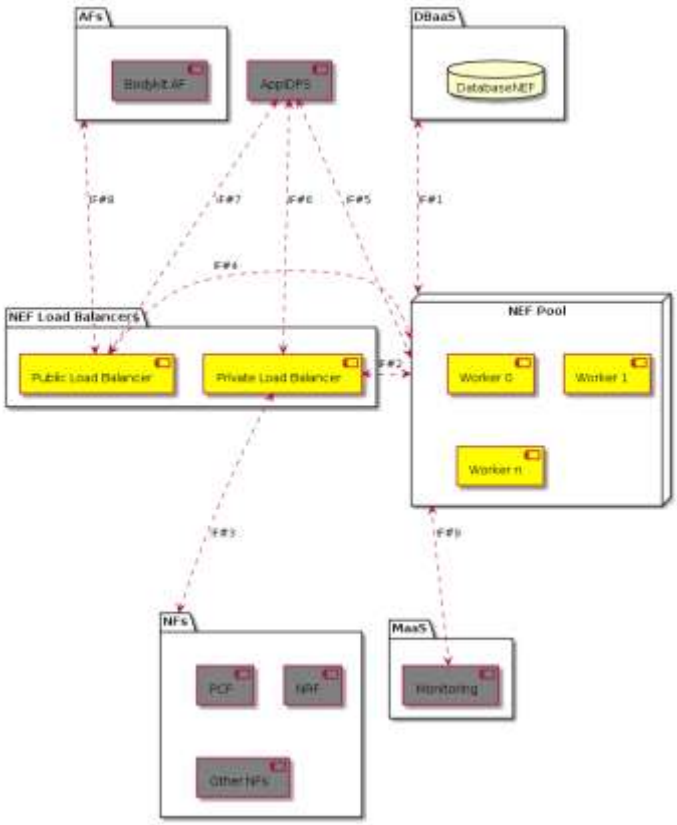

Figure 15 – NEF - Internal view

The components of NEF are described in the table below.

Table 9 – NEF components

Component	Component Description
NEF Public LB (Load Balancer) or NEF External LB	Component responsible to receive the requests from AFs. This component exposes the public API of NEF to other services.
NEF Private LB or NEF Internal LB	Component responsible to receive the requests from NFs, like PCF, NRF.
NEF Pool Worker	Components that perform the required work, this component interacts with the database and processes all the requests that are received by the Load Balancer components.

3.4.3 Interfaces

DNS

The interfaces of the DNS component are summarized in the table below.

Table 10 – DNS interface components

IF ID	Interface Description	Components	Type of Interface	Exposure interface (Internal/External; Mandatory/Optional)

IF ID	Interface Description	Components	Type of Interface	Exposure interface (Internal/External; Mandatory/Optional)
#1	Interface to receive DNS queries and to provide DNS replies.	DNS Frontend ↔ regular DNS client(s)	DNS or DoH (DNS over HTTPS)	External Mandatory
#2	REST Interface to receive information to manage DNS records.		REST + SSL/TLS	External Mandatory
#3	Interface to send probes to the IDPS solution to enhance security of DNS.	DNS Frontend ↔ AppIDPS		External Optional (if security is not required)
#4	Interface to manage and monitor the DNS internal components.	DNSaaS API → DNS Frontend DNS Forwarder DNS Server	TCP socket to collect metrics and check status of components.	Internal Mandatory
#5	Interface to forward initial DNS requests to the DNS Forwarders.	DNS Frontend ↔ DNS Forwarder	DNS	Internal Optional (DNS Frontend might be merged with DNS Forwarder)
#6	Interface to store and retrieve data in the database service.	DNSaaS API ↔ DBaaS	REST (if considering Database as a Service) SQL considering regular RDBMS.	Internal Mandatory
#7	Interface to receive DNS queries and to provide DNS replies.	DNS Forwarder ↔ DNS Server	DNS	Internal Mandatory
#8	Interface CRUD (Create/Read/Update/Delete) operations in DNS data.	DNS Server ↔ DBaaS	SQL	Internal
#9	Interface between DNSaaS API and DNSaaS Management.	DNSaaS API ↔ DNSaaS Managerment		Internal
#10	Interface to the monitoring system.	DNSaaS Management ↔ Monitoring		External

NEF

The components of NEF use different interfaces to communicate, and these are described in the table below.

Table 11 – NEF interface components

IF ID	Interface Description	Components	Type of Interface	Exposure interface (Internal/External; Mandatory/Optional)

IF ID	Interface Description	Components	Type of Interface	Exposure interface (Internal/External; Mandatory/Optional)
#1	Interface for the NEF Workers to interact with the NEF databases.	NEF Workers ↔ DB Cluster	TCP + SSL/TLS + SQL	Internal Mandatory
#2	Interface for the Private Load Balancer to distribute load towards the NEF Workers.	Private Load Balancer ↔ NEF Workers	REST + SSL/TLS	Internal Mandatory
#3	Interface for the NFs to access NEF and its API.	NFs ↔ Private Load Balancer	REST + SSL/TLS	External Mandatory
#4	Interface for the Public Load Balancer to distribute load towards the NEF Workers.	Public Load Balancer ↔ NEF Workers	REST + SSL/TLS	Internal Mandatory
#5	Interface for the NEF Workers to provide metrics to the APP IDPS (Intrusion Detection and Protection System) probes.	NEF Workers ↔ APP IDPS Probes	REST + SSL/TLS	Internal Optional (if APP IDPS is used)
#6	Interface for the APP IDPS probes to apply policies in the Private Load Balancer.	APP IDPS Probes ↔ Private Load Balancer	REST + SSL/TLS	Internal Optional (if APP IDPS is used)
#7	Interface for the APP IDPS probes to apply policies in the Public Load Balancer.	APP IDPS Probes ↔ Public Load Balancer	REST + SSL/TLS	Internal Optional (if APP IDPS is used)
#8	Interface for the AFs to access NEF and its API.	AFs ↔ Public Load Balancer	REST + SSL/TLS	External Mandatory
#9	Interface to the monitoring system.	NEF Pool ↔ Monitoring		External

The interface #8 has been specified in the ETSI TS 23.501, as the Common API Framework for 3GPP Northbound APIs (CAPIF). The initial version of NEF does not include support for CAPIF.

3.5 Security Framework

The security of a complex system like an operator's network, especially one that introduces a disruptive technology such as the upcoming 5G, is a very complex subject and will always be a work in progress due to the nature of the subject. Each discovered threat requires the evaluation of its impact on the network considering the security mechanisms already in place, auditing if the current security measures are enough to thwart the threat. Upon this analysis, if needed, new security elements or procedures that aim to reduce or completely nullify the threat must be designed and implemented.

The security framework aims to secure the services supporting the 5G core. Different security products/services exist, but they are integrated in the event collector (kind of dashboard) which allows the visualization of the events related with security and the action taken or suggested by the security components. This section, after providing the integrated view of the security framework, details each of the security products/services, that include:

- **AppIDPS –** Developed by OneSource. Aims to secure services/products in a service perspective. Each service that requires protection (like DNS, NEF) sends information to the AppIDPS for analysis.

- **VAS (Vulnerability Assessment System) −** Developed by Ubiwhere. Has the purpose of scanning a set of network functions for multiple vulnerabilities by means of Network Vulnerability Tests (NVTs) regularly updated by an open source community.
- **HoneyNet −** Developed by Ubiwhere. It aims to avert illicit activity inside provisioned network services or even inside the 5G core network, by deploying diversion and detection mechanisms in order to prevent unauthorized penetration by external entities, i.e. attackers.
- **DNS IDPS −** Developed by PDMFC. Aims to be a protection layer to provide better security for the network DNS Service, by intercepting malicious or malformed DNS requests before they arrive at the DNS Server itself.

In this section, a brief description of the Security Framework is provided in subsection 3.5.1 (External View) and each individual component is described and detailed in subsection 3.5.2 (Internal View).

3.5.1 External View

A general overview of the Security Framework is depicted in the following diagram. As briefly described, each module has its own distinct purpose, tackling different security vulnerabilities that are prone to occur in the network. Nonetheless, each main module, namely the App IDPS, the VAS, the Honeynet and the DNS IDPS, reports to the Event Collector which gathers all the occurred security events and the subsequent performed measures by the security components as pictured in the figure below.

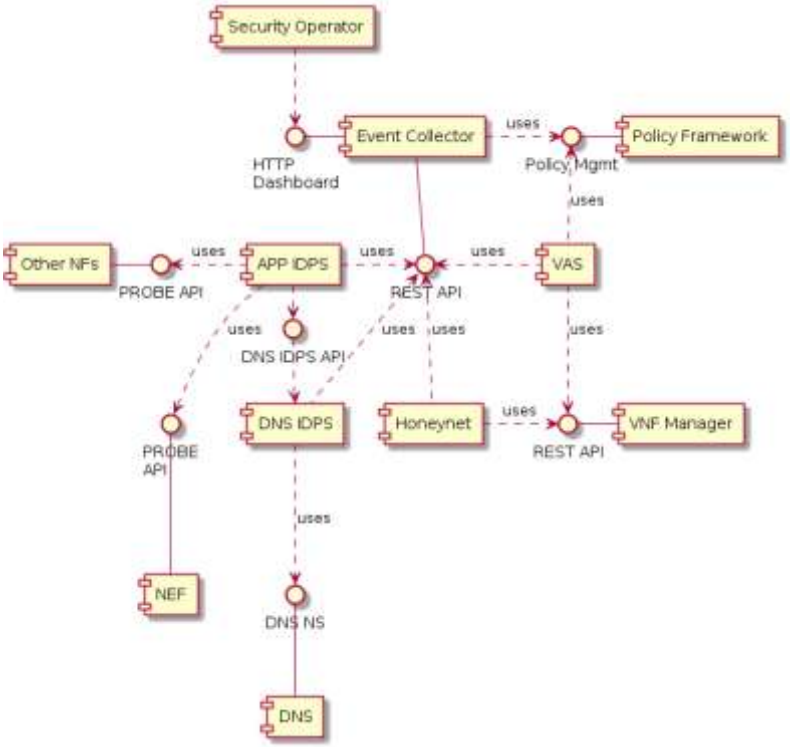

Figure 16 – Security Framework - External view

The architecture and internal interfaces of the mentioned modules are described in more detail in the following sections.

3.5.2 Internal View

APP IDPS

The App IDPS gathers information from other services, through specific APIs, to perform a security analysis.

A general perspective of the App IDPS is depicted in the figure below.

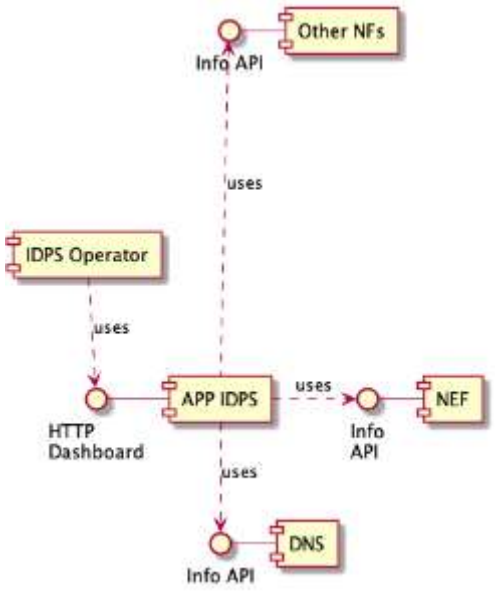

Figure 17 – APP IDPS – General perspective

The internal architecture of App IDPS, includes diverse elements:

- Scalable component to gather information from Network Functions (NFs) and Applications Functions (AFs), which aim to be secured.
- Correlation component to perform the security analysis based on a set of rules and security algorithms to detect security flaws, attacks in the provided information.
- Security server component which assures that the signaling between the different components is secured, and performs the integration with external dashboards.
- Real-time database components, which are responsible to provide the information in real-time to support the security analysis of the correlation components.

The internal view is next pictured.

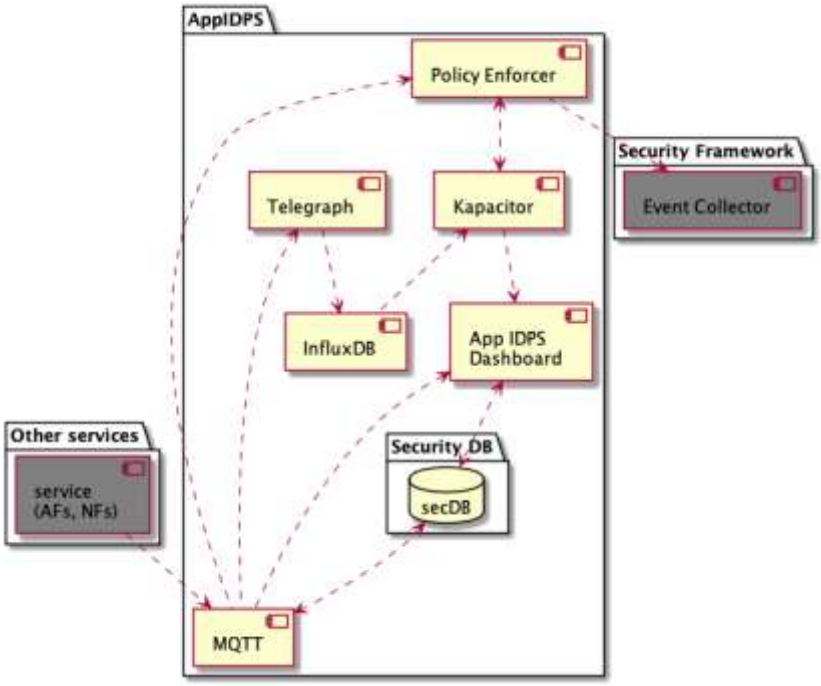

Figure 18 – APP IDPS - Internal view

The components are described in the following table.

Table 12 – App IDPS components

Component	Component Description
Policy Enforcer	Component to apply policies in response to certain events. Policies are placed in a specific MQTT topic, which is subscribed by the other components that are responsible for applying them. It also keeps record of applied policies in its internal database.
InfluxDB	Time series database optimized for fast, high-availability storage and retrieval of time series data. It stores large quantities of monitoring data from Telegraph.
Kapacitor	Data processing engine to process data from InfluxDB with custom logic or user-defined functions to process alerts with dynamic thresholds, match metrics for patterns, compute statistical anomalies, and perform specific actions based on these alerts like dynamic load rebalancing.
Dashboard	Component to allow human interaction with the system. It allows the definition of rules and policies, and also to handle situations that the automatic system cannot handle by itself.
Telegraph	An agent to collect and report both metrics and data. Its plugins allow the interaction with multiple data sources. In this case, it receives data from MQTT and feeds the InfluxDB with the collected data.
SecurityDB	Database with ACLs to provide access control to MQTT server and its topics. It also includes AAA (Authentication, Authorization and Accounting) for the MQTT server and the Dashboard.
MQTT	MQTT is an ISO standard publish-subscribe-based messaging protocol. This component is the broker for such messages and allows receiving and sending data from/to external components.

VAS

The present section highlights the VAS high level architecture. The goal is to have two main modules:

- **VAS Master**: (i) collect and control the scanning tests and results of each slave and (ii) compiles the NS vulnerabilities to be reported to the common dashboard.

- **VAS Slave**: related to one or several Network Services (NS) it's responsible to (i) find vulnerabilities within each NS Virtual Deployment Units (VDU).

Since it's expected that this solution will leverage on opensource works, such as OpenVAS, in more detail both VAS Master and VAS Slave are composed by OpenVAS components. The following paragraph describes the internal architecture of both VAS Master and VAS Slave:

VAS Master:

Table 13 – VAS Master components

Component	Description
openvasmd	Responsible to coordinate all the agents within the whole network and gather all the results. Provides visual representation of the events to human operators.
Manager Client	Allows the openvasmd configuration, capable of understanding requests related to M5G VAS specific operation and translate them into openvasmd commands.

VAS Slave:

Table 14 – VAS Slave components

Component	Description
openvassd	Performs the desired vulnerability scans, finding the targeted vulnerabilities.
Manager Client	Controls the openvassd process.
Register	Performs the registration on the VAS Master, providing contextual information.
NVT (Network Vulnerability Test) Sync	Component that allows the Network Vulnerability Test synchronization between the Slave and the Master.

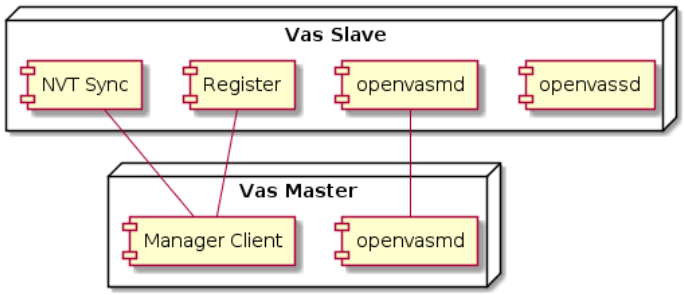

Figure 19 – VAS - Internal view

The following figure illustrates the interfaces of the two main components of VAS, namely the VAS Slave and the VAS Master, for a better understanding of their internal interactions. The details of each interface are further described in Section 3.5.3.

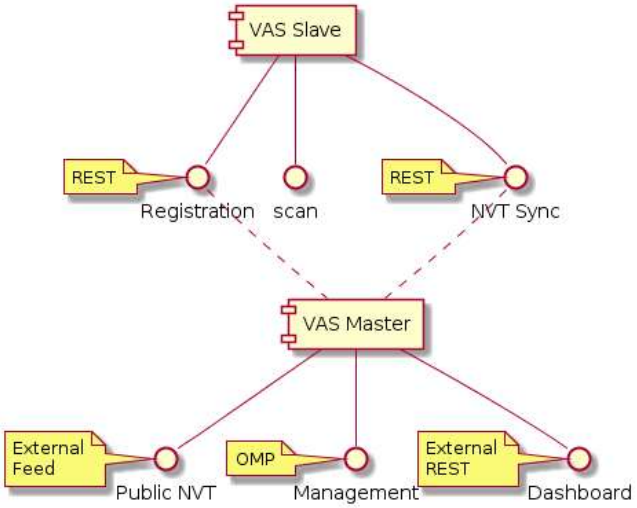

Figure 20 – VAS – VAS Slave/Master interfaces

Honeynet

The honeynet is composed by multiple Honeynet Agents and at least one Honeynet Server. The Honeynet Agents are meant to be lightweight servers, deployed in multiple locations (typically inside network services), that forward all incoming traffic to the Honeynet Server. All traffic will preserve the original remote IP address and will be sent to the Honeynet Server through an encrypted tunnel.

The "real" honeypots, i.e. the services, simulators, proxies and containers are running in the Honeynet Server which accepts the incoming connections from its agents. The Honeynet server can either provide simulated services (such as an SSH [Secure Shell Simulator) or real services deployed on containers or VMs through the usage of proxies. The former is more lightweight but is usually prone to be identified as honeypots, the latter on the other hand, take more resources but the honeypot fingerprinting is rather difficult.
Despite the used service paradigm, the Honeynet Server is capable to monitor and log all activity performed by the attacker, which will be later processed and reported to the Event Collector module (as described in Section 3.5.1).

The general responsibilities of the two main modules are described as follows:

- **Honeynet Agent** - (i) register itself to a Honeynet Server and (ii) forward traffic to the Listener module in the Honeynet Server
- **Honeynet Server** - (i) register agents upon successful authentication, (ii) provide simulated services and/or proxies to Linux containers or remote servers and (iii) raise containers, based on templates, upon request

The following diagram shows the basic architecture of Honeynet along with examples of services, proxies and containers supported by the Honeynet Server.

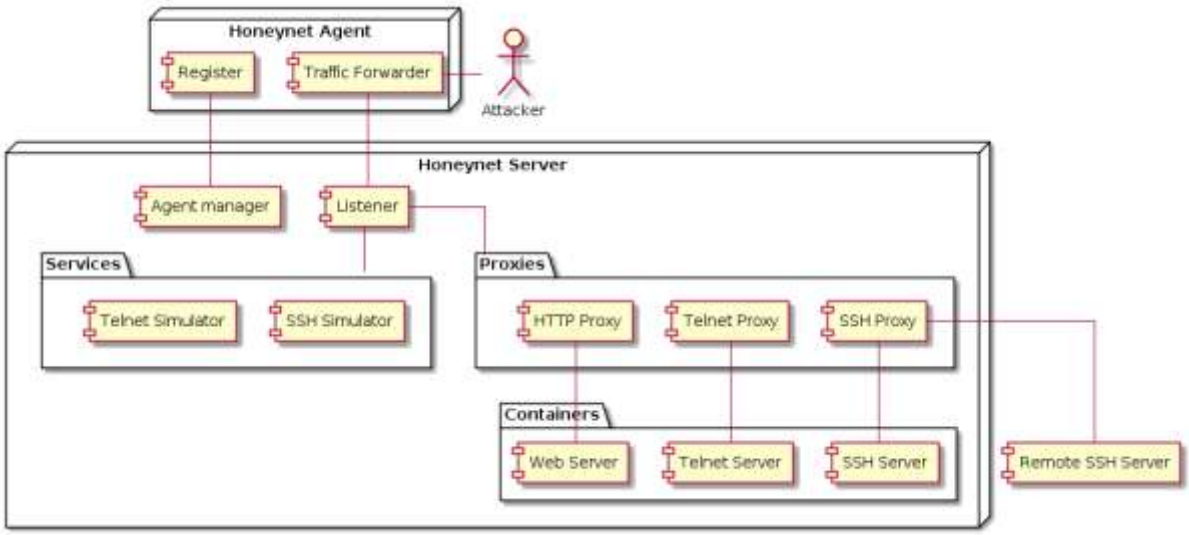

Figure 21 – Honeynet – Architecture

The following figure illustrates the interfaces of the two main components of the Honeynet solution, namely the Honeynet Agent and the Honeynet Server, for a better understanding of their internal interactions. The details of each interface are further described in Section 3.5.3.3.

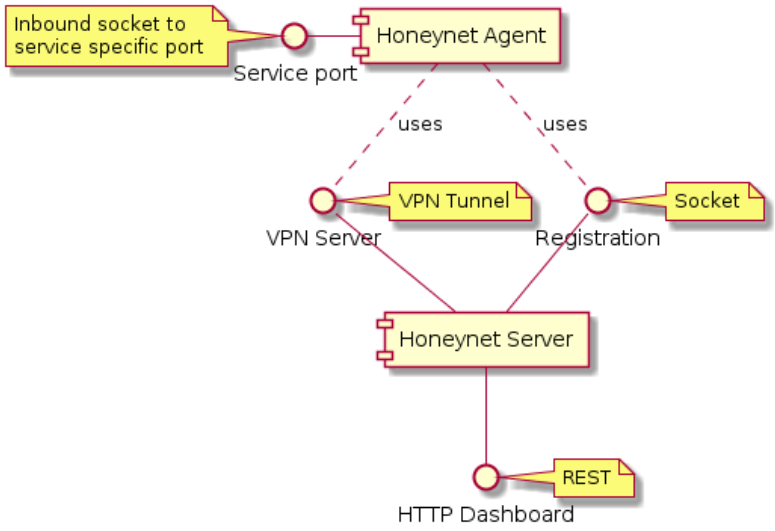

Figure 22 – Honeynet – Honeynet Agent/Server

DNS IDPS

The DNS-IDPS was designed specifically as a security add-on for DNS services.

It includes both an Intrusion Detection System (IDS) and an Intrusion Prevention System (IPS), and incorporates several features, as for example:

- Detecting DNS attacks and exploits
- Stopping attacks from reaching the DNS servers
- Mitigating some attacks that can't be blocked
- Generating security reports and alerts
- Monitoring and logging the DNS server usage

The DNS-IDPS High-Level architecture is presented in the following diagram:

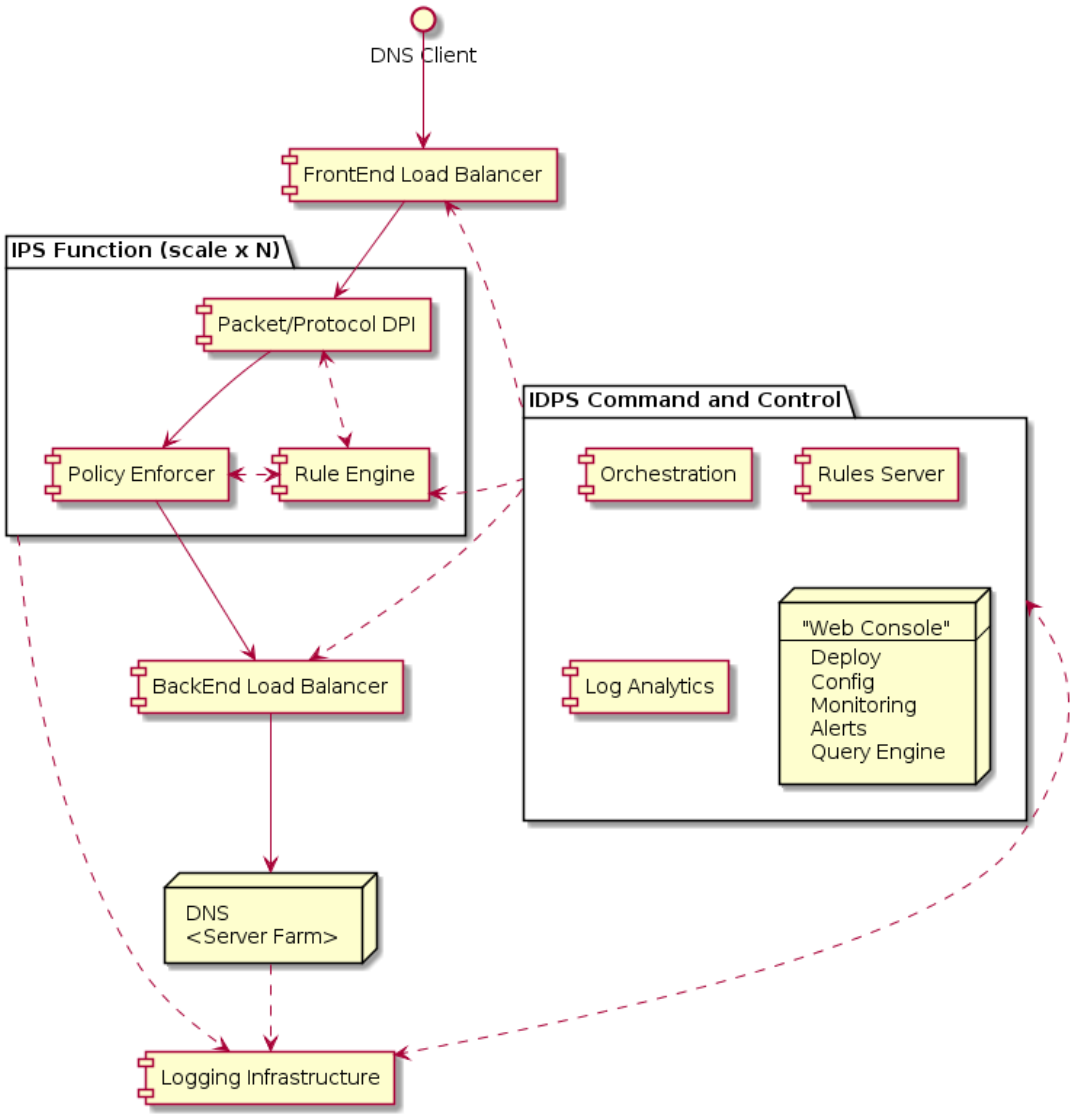

Figure 23 – DNS-IDPS - High-Level architecture

The IDPS includes the following components described in the next table.

Table 15 – DNS IDPS components

Component	Component Description
Frontend Load Balancer	This component is a specialized Load Balancer that distributes the DNS requests across the IPS Function components. The main characteristics of this Load Balancer are: • Supports both TCP and UDP • It has an orchestration agent, so that more IPS components can be added or removed and the Load Balancer will adapt accordingly. • It monitors the availability of the IPS components and adjusts itself to maintain the peak performance.
Backend Load Balancer	The Backend Load Balancer is like the Frontend Load Balancer but distributes the requests to the DNS Servers.
DNS Server Farm	This is just one or more regular DNS Servers, not a part of the IDPS.
Logging Infrastructure	The main function of this component is: • Collect and process the logs from the other components.
IPS Function	This component can be deployed and scaled as a Virtual Machine (VM) or as a physical server. It directs the DNS requests through a pipeline with the following workflow: 1. It opens, examines and analyzes the request (Packet/Protocol DPI). 2. Logs request. 3. Uses rule engine to decide which action to take (Policy Enforcer): a) Forward Request b) Drop Request c) Introduce a delay in request 4. Retrieve DNS response and reply to original client.

Component	Component Description
IDPS Command and Control (C&C)	This component is the Master component of the IDPS and contains several modules. The IDPS Command and Control consists of: Orchestration module This module manages the infrastructure, by: • Monitoring the other components for availability and performance • Starting up the components • Dynamically re-configuring the Load-Balancers • Pushing changes to the Rule Engine of the IPS Rules Server This module manages all the IPDS Rules, namely: • Real Time Rules (running on the IPS VNF) • These Rules are applied in real-time to the incoming DNS requests and responses • Offline rules • These rules are applied through the Log Analytics module, and can generate rule changes for the IPS Log Analytics This module runs analysis on the log events received by the Logging Infrastructure, and will communicate changes to the Rule Engine based on several heuristics and pattern recognition algorithms Web Console This module provides a Management Console (Both GUI and API based), through which an administrator can: • Perform change management • Deploy changes and add new features • Access monitoring dashboard and reports • Manage alerts • Perform queries to the system's information

Event Collector

Being part of the Security framework, the Event Collector gathers information provided by the security framework components (APP IDPS, VAS and Honeynet).

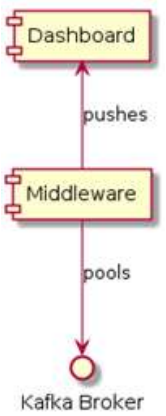

Figure 24 – Event collector - Components

The Event Collector is composed internally by the Dashboard to display a visualization of the information to an operator; a middleware to gather, collect and validate the events; and receives the messages by a Kafka Broker (refer to Annex 8.2). The components are listed in the following table.

Table 16 – Event Collector components

Component	Description
Dashboard	Provides visual representation of the events to human operators.
Middleware	Collects, validates, and processes the events.
Kafka Broker	Messaging broker.

The messages received by the Event Collector will follow a Data Model currently under specification

3.5.3 Interfaces

App IDPS

The components of the App IDPS use different interfaces to communicate, and these are described in the table below.

Table 17 – App IDPS interfaces

IF ID	Interface Description	Components	Type of Interface	Exposure interface (Internal/External; Mandatory/Optional)
#1	Interface to publish policies to be implemented by external components.	Policy Enforcer ↔ MQTT	MQTT over TCP/IP with SSL/TLS	Internal Mandatory
#2	Interface to apply policies based on human decisions.	Dashboard ↔ Policy Enforcer	REST + SSL/TLS	Internal Mandatory
#3	Interface to apply policies based on rules.	Kapacitor ↔ Policy Enforcer	REST + SSL/TLS	Internal Mandatory
#4	Interface to apply rules based on human decisions.	Dashboard ↔ Kapacitor	REST + SSL/TLS	Internal Mandatory

IF ID	Interface Description	Components	Type of Interface	Exposure interface (Internal/External; Mandatory/Optional)
#5	Interface for Kapacitor to receive data for processing from InfluxDB.	InfluxDB ↔ Kapacitor	HTTP, TCP, and UDP with line protocol (graphite compatible)	Internal Mandatory
#6	Interface for Telegraph to place data into InfluxDB.	Telegraph ↔ InfluxDB	HTTP, TCP, and UDP with line protocol (graphite compatible)	Internal Mandatory
#7	Interface for Telegraph to get data from MQTT topics.	MQTT ↔ Telegraph	MQTT over TCP/IP with SSL/TLS	Internal Mandatory
#8	Interface for Dashboard to subscribe to MQTT topics on policies.	Dashboard ↔ MQTT	MQTT over TCP/IP with SSL/TLS	Internal Optional
#9	Interface for MQTT to get AAA information and ACLs.	MQTT ↔ SecurityDB	TCP + SSL/TLS + SQL	Internal Mandatory
#10	Interface for Dashboard to get AAA information.	Dashboard ↔ SecurityDB	TCP + SSL/TLS + SQL	Internal Mandatory
#11	Interface for probes to publish metrics and receive policies.	Probes ↔ MQTT	MQTT over TCP/IP with SSL/TLS	External Mandatory
#12	Interface between Policy Enforcer and Event Collector.	Policy Enfocer ↔ Event Collector	REST + SSL/TLS	External Optional

Probes send metrics to APP IDPS and receive policies. The information that is sent for analysis includes diverse items, and contains both a common header and a body with app-specific information. This is illustrated in the figure below.

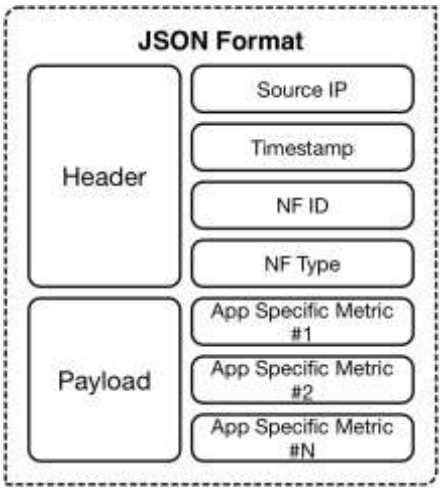

Figure 25 – APP IDPS – Metrics body format

The fields in the common header are the source IP of the application client, timestamp of access, ID of the network function and type of the network function (according to 3GPP 5G Core).

As for policies, the body format is also defined by each app. The common format is illustrated below.

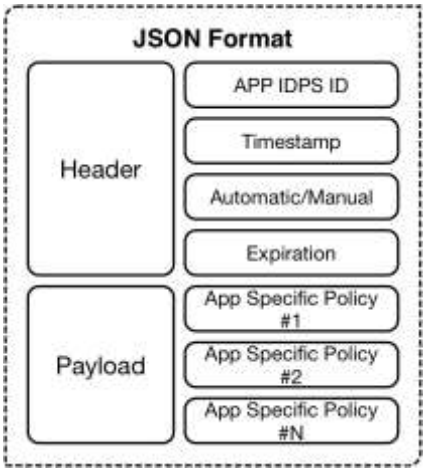

Figure 26 – APP IDPS – Policies body format

The fields in the common header are the ID of the APP IDPS decision, timestamp of decision, information if the policy was automatically defined or manually set and expiration timestamp. The expiration timestamp may be set with zeros or nines to define infinite expiration.

VAS

It's expected that the VAS solution will use both internal interfaces and external interfaces.

The Internal interfaces are interfaces that are only present within a specific component, e.g., VAS Master or VAS Slave, while External interfaces are interfaces that connect these components or allow the VAS Master to collect information on the Internet.

Table 18 – VAS interfaces

IF ID	Description	Components	Type of interface	Exposure of interface (Internal/External)
#1	Interface to manage the openvasmd.	Manager Client ↔ openvasmd	OMP	Internal
#2	Interface to register slaves.	Manager Client ↔ VAS Slave	REST	External
#3	Interface to update the NVTs.	openvasmd → Public RSS Feed	Feed	External
#4	Interface to update Slave NVTs.	Manager Client ↔ VAS Slave	REST	External
#5	Interface to submit security data to the dashboard.	Manager Client ↔ Dashboard	REST	External
#6	Interface to manage VAS slave.	openvasmd ↔ openvassd	OTP	Internal
#7	Interface between Slaves and Master.	openvasmd ↔ openvasmd	TCP	External

Honeynet

The Honeynet interfaces are summarized in the following table.

Table 19 – Honeynet interfaces

IF ID	Description	Components	Type of interface	Exposure of interface (Internal/External)
#1	Interface to analyze Honeynet activity.	Honeynet Operator ↔ HTTP Dashboard	REST	Internal
#2	Interface to register Honeynet Agents.	Register ↔ Agent manager	Socket	External
#3	Interface to VPN tunnel.	Traffic Forwarder ↔ Listener	OpenVPN Tunnel	External
#4	Interface accept incoming service connections.	Attacker ↔Traffic Forwarder	Socket	External

DNS IDPS

The DNS IDPS interfaces are summarized in the following table.

Table 20 – DNS IDPS interfaces

IF ID	Description	Components	Type of interface	Exposure of interface (Internal/External)
#1	DNS request from outside.	Frontend Load Balancer.	DNS	External
#2	Frontend DNS request distribution.	Frontend Load Balancer, IPS Function.	DNS	Internal
#3	Backend DNS request distribution.	IPS Function, Backend Load Balancer.	DNS	Internal
#4	Internal DNS request.	Backend Load Balancer, DNS server farm.	DNS	External
#5	IPS Configuration push.	IDPS Command and Control, IPS Function.	REST + SSL/TLS	Internal
#6	LB configuration push.	IDPS Command and Control, Backend Load Balancer, Frontend Load Balancer.	REST + SSL/TLS	Internal
#7	Logging event.	All elements, Logging Infrastructure.	REST + SSL/TLS	Internal
#8	Send event to Event Collector.	IDPS Command and Control, Event Collector.	REST + SSL/TLS	External

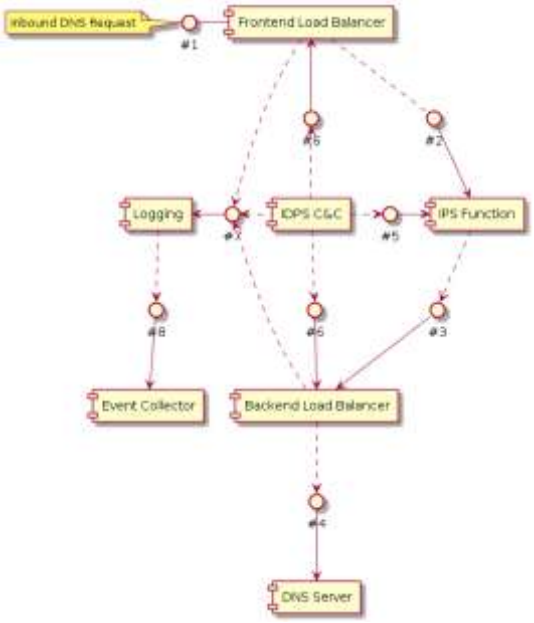

Figure 27 – DNS IDPS interfaces

Event Collector

The Event Collector interfaces are listed in the following table.

Table 21 – Event Collector interfaces

IF ID	Description	Interaction	Protocol	Exposure of interface (Internal/External)
#1	Interface to send events to the dashboard after being validated.	Middleware - Dashboard	WebSockets	Internal
#2	Interface used by the Middleware to consume.	Middleware - Kafka Broker	Kafka Protocol	Internal

4 Concurrency View

This chapter describes the design and implementation of use case flows to be exercised during pilot phase.

4.1 Use case 1 – vCDN Service Orchestration in 5G

4.1.1 Overview

The previous deliverable D2.1 [7] presented several sub-cases within the vCDN (virtual Content Delivery Network) use case. Such sub-cases highlighted several processes during the complete life cycle of the vCDN service with the purpose of showing the impact and novelties brought by 5G Networks. On this deliverable, the vCDN service will once again be used as an example of a service over a 5G environment.

It should be noted that it is not the purpose of this deliverable to extend or evolve the previous sub-cases but rather to validate the architecture presented in the previous sections. The selected sub-cases are intended to promote the interactions between various components while highlighting the role of component's interfaces during their resolution.

Having that in mind, three sub-cases were selected:

- QoS configuration
- Congestion Management
- Intrusion Detection and Mitigation

The first sub-case shows the simple but mandatory process of QoS configuration employing 5G Core, a process which in 5G is significantly evolved but, enabling greater customization granularity as well as a more dynamic 5G QoS (QoS Flow Identity) and 5QI (5G QoS Indicator) setting. The second and third-sub case demonstrate the PPS2 components role in managing 2 different types of events: on the sub-case 2, a scenario where a RAN congestion is caused by a vCDN service is presented; on the sub-case 3, an attack over the functions that realize the vCDN infrastructure is performed by using a lateral movement. The congestion event is resolved at two different levels, the first one being the 5G Core and its mechanisms for session control, the second one through the combined actions of Operation Support Systems (OSS). The attack sub-case highlights the role of the decision-making components, i.e. Policy Framework, to mitigate an attack on a service with the use of security functions and service level orchestration capabilities.

4.1.2 Message signaling charts

This is the simplest and most straightforward use case for policy control mechanisms. In this use case, a certain QoS profile has been provisioned for traffic associated to a vCDN service and its enforcement will be decided by the PCF, which evaluates the policies to be applied when the service is initiated.

Service initiation is hereby considered to be triggered in the session context by the UE itself when it tries to access a given vCDN content but, there are other possibilities, as services can be triggered otherwise.

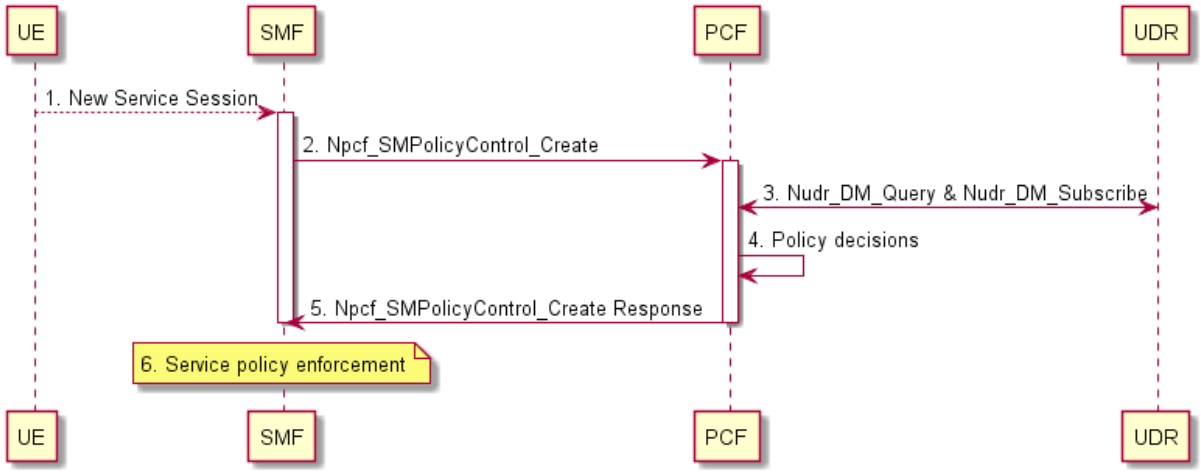

Figure 28 – Sub-case 1 Sequence Diagram

The following table describes the different steps depicted in the diagram.

Table 22 – Sub-case 1 - Sequence Diagram Steps

Step Id	Step's Description	Mandatory/ Optional
1	A new service session is triggered and is signaled to the SMF.	Mandatory
2	SMF uses Npcf Service Npcf_SMPolicyControl_Create to setup a policy control session for the service on the PCF.	Mandatory
3	PCF queries UDR (User Data Repository) for a service profile for the given service and subscribes to any further modifications to it (Nudr_DM_Query & Nudr_DM_Subscribe).	Mandatory
4	PCF makes its policy decisions considering the delivered Service Profile and all policies that apply to that particular service session.	Mandatory
5	PCF answers to the SMF (Npcf_SMPolicyControl_Create Response) with the actions that are to be enforced.	Mandatory
6	SMF interacts with UPF (or other enforcement points) to guarantee the application of PCF decisions.	Mandatory

4.1.2.1 Sub-case 2 - Congestion management

For this sub-case it should be noted that some steps are omitted such as the communications between the Honeynet Agent and the Honeynet Server or the content population within the vCDN Surrogate. This is due to the fact that these steps are out of the scope of the congestion response mechanisms highlighted here and would only make the subcase more difficult to understand.

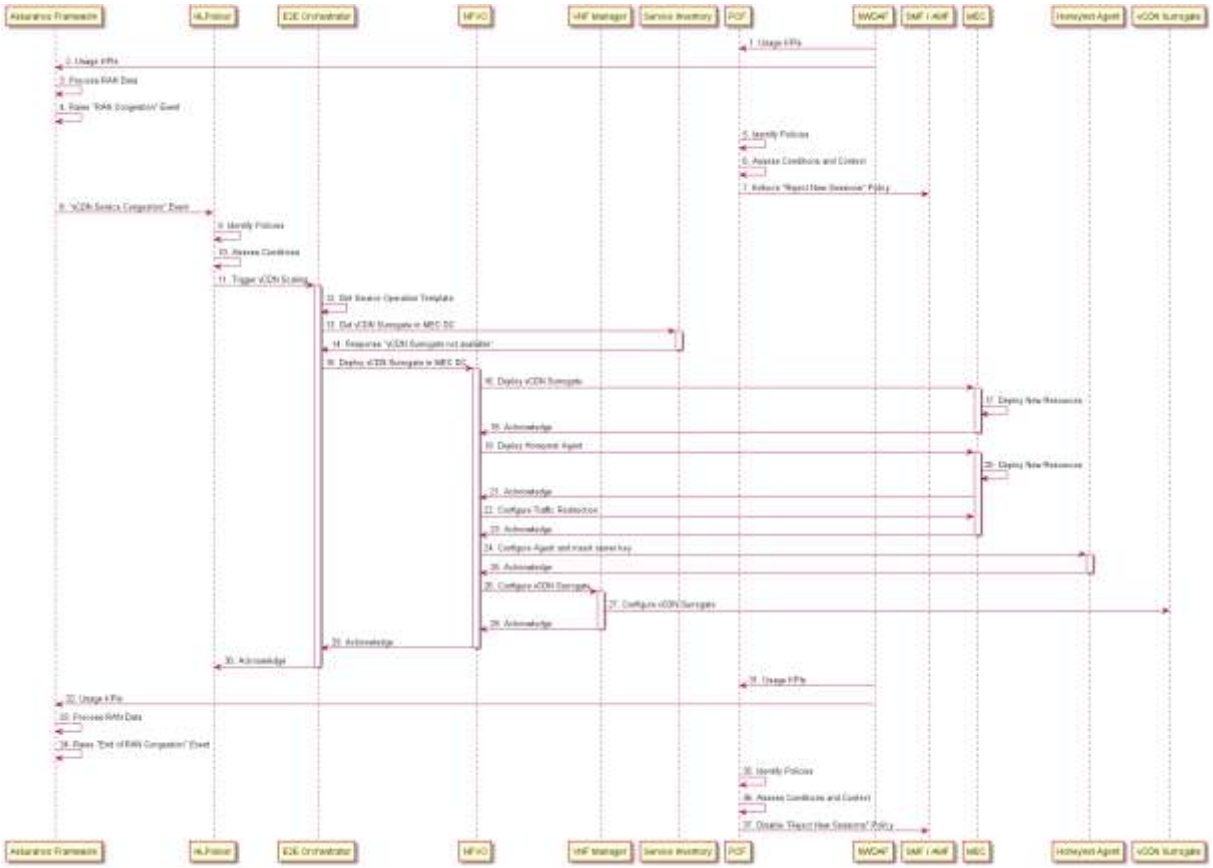

Figure 29 – Sub-case 2 Sequence Diagram

The following table describes the different steps depicted in the diagram.

Table 23 – Sub-case 2 - Sequence Diagram Steps

Step Id	Step's Description	Mandatory/ Optional
1	NWDAF exposes RAN KPI's to PCF.	Mandatory
2	NWDAF exposes RAN KPI's to external systems, i.e. Assurance Framework	Mandatory
3	Assurance Framework processes RAN KPI's.	Mandatory
4	Based on the KPI's received from NWDAF, the Assurance Framework raises a RAN Congestion Event.	Mandatory
5	The PCF after receiving KPI's that are above a certain threshold will try to identify related Policies.	Mandatory
6	After the impacted Policies have been identified, the PCF evaluates each policy conditions.	Mandatory
7	The PCF triggers the policy action whose conditions have been validated as true, therefore the SMF is notified to reject new sessions on the geographical location where the RAN congestion is happening.	Mandatory
8	The Assurance Framework, by monitoring all services, identifies a congestion on the vCDN service and sends an event to the Policy Framework (HLPolicer).	Mandatory
9	The HLPolicer starts identifying policies related to the vCDN congestion event.	Mandatory
10	After identifying the impacted policies, the HLPolicer starts evaluating each policy conditions.	Mandatory

Step Id	Step's Description	Mandatory/ Optional
11	After identifying the policy whose conditions have been validated as true, the HLPolicer responds with the respective action by interfacing the E2E Orchestrator, which is a scaling operation on the vCDN service.	Mandatory
12	The E2E Orchestrator fetches the operation template from its internal catalogue.	Mandatory
13	The E2E Orchestrator requests the vCDN Surrogate instance information (located on the MEC (Multi-access Edge Computing) DC where vCDN congestion is happening) from the Inventory.	Mandatory
14	The inventory replies to the E2E Orchestrator that no vCDN Surrogate instance is available at the specified location.	Mandatory
15	The E2E Orchestrator triggers the NFVO to deploy a new vCDN Surrogate instance on the specified MEC DC.	Mandatory
16	The NFVO interfaces the MEC DC to deploy a new vCDN Surrogate instance.	Mandatory
17	The MEC DC instantiates the necessary resources for the vCDN Surrogate.	Mandatory
18	The MEC DC acknowledges the NFVO that the resources have been deployed.	Mandatory
19	The NFVO interfaces the MEC to deploy a new Honeynet Agent instance for the new vCDN Surrogate instance.	Optional
20	The MEC DC instantiates the necessary resources for the Honeynet Agent.	Optional
21	The MEC DC acknowledges the NFVO that the resources have been deployed.	Optional
22	The NFVO interfaces the MEC DC to configure a traffic redirection rule between the vCDN Surrogate external connection point and the Honeynet Agent.	Optional
23	The MEC DC acknowledges the NFVO that the traffic redirection rule has been enforced.	Optional
24	The NFVO configures the Honeynet Agent and uploads the Honeynet Server key.	Optional
25	The Honeynet Agent acknowledges the NFVO the configurations and the correct Honeynet Server key.	Optional
26	The NFVO commands the VNFM to configure the vDCN Surrogate.	Mandatory
27	The VNFM configures the vCDN Surrogate.	Mandatory
28	The VNFM acknowledges the NFVO the configurations.	Mandatory
29	The NFVO acknowledges the E2E Orchestrator the completion of the deployment of the new vCDN Surrogate.	Mandatory
30	The E2E Orchestrator acknowledges the HLPolicer of the completion of the scaling operation on the vCDN Service.	Mandatory
31	NWDAF exposes RAN KPI's to PCF.	Mandatory
32	NWDAF exposes RAN KPI's to external systems, i.e. Assurance Framework.	Mandatory
33	Assurance Framework processes RAN KPI's.	Mandatory
34	Basing on the KPI's received from NWDAF, the Assurance Framework signals the end of the RAN Congestion Event.	Mandatory
35	The PCF after receiving KPI's that are above a certain threshold will try to identify related Policies.	Mandatory
36	After the impacted Policies have been identified, the PCF evaluates each policy conditions.	Mandatory
37	The PCF triggers the policy action whose conditions have been validated as true, therefore the SMF is notified to accept new sessions on the geographical location where the RAN congestion was previously reported.	Mandatory

4.1.2.2 Sub-case 3 - Intrusion detection and mitigation

When a network function is compromised, i.e. an attacker has gained access to a virtual machine, there must be a method of detecting the attacker's illicit activity. In order to detect penetration attacks inside a network service, a Honeynet Agent function is provisioned in the same network of the remaining functions with the prospect of being accessed by an attacker. Concerning that during normal operations the functions of the network service never contact the Honeynet Agent, when the latter is accessed in some way an attacker is detected. Following this detection the network service can be isolated or terminated, depending on the operator's policies.

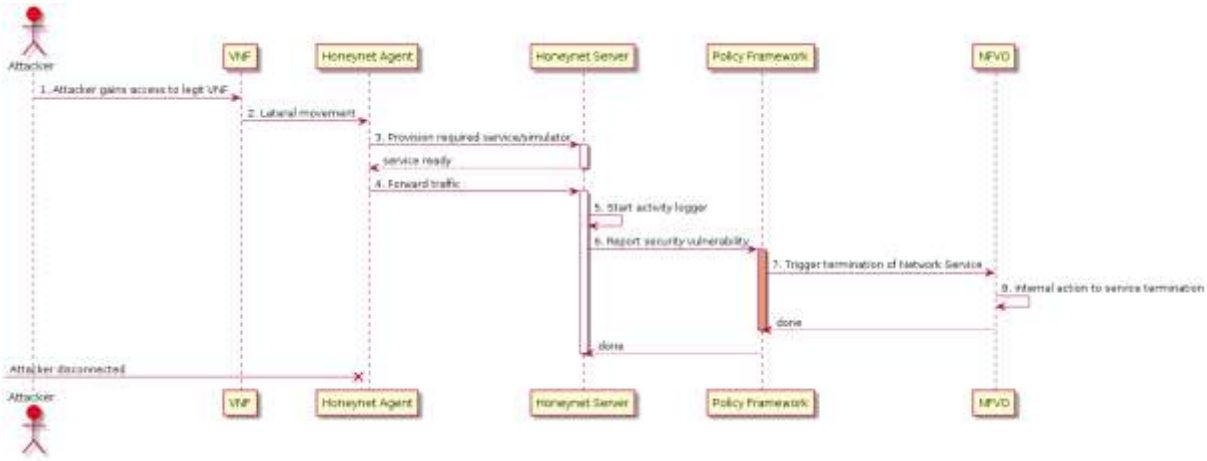

Figure 30 – Sub-case 3 Sequence Diagram

The following table describes the different steps depicted in the diagram.

Table 24 – Sub-case 3 - Sequence Diagram Steps

Step Id	Step's Description	Mandatory/ Optional
1	The attacker somehow gained access to a network function (VNF)	Mandatory
2	The attacker attempts the exploitation of other network functions provisioned in the same network (lateral movement). (i) Attempt to widen his footprint in order to gain access to as much information as possible. (ii) The exploited network function is in fact a Honeynet Agent deployed in the same service network. (iii) The Honeynet Agent function is never supposed to be contacted following the normal operation of the service.	Mandatory
3	The Honeynet Agent contacts the Honeynet Server, which provisions an isolated honeypot machine.	Mandatory
4	From the perspective of the attacker, a success penetration has occurred and he has now access to a real machine.	Mandatory
5	The Honeynet Server logs all the activity of the attacker with the purpose of discovering new attack vectors and methodologies.	Mandatory
6	The Honeynet Server reports the security vulnerability to the Policy Framework (HLPolicer).	Mandatory
7	The HLPolicer acquires a suitable policy in order to terminate/isolate the network service.	Mandatory
8	Eventually this information reaches the NFVO which endures internal actions in order to terminate/isolate the network service.	Mandatory

4.2 Use case 2 – PPDR leveraging IoT in 5G

4.2.1 Overview

This use case is associated with IoT usage in 5G in the context of mission-critical communications in PPDR (Public Protection and Disaster Relief). The goals of this use case have been documented in deliverable D2.1 [7] and are briefly discussed here to highlight the contributions of the architecture and diverse components as the enablers of such use case:

1. Optimization of resources for critical missions. The NEF component, acting as the entry point of the network and exposing functionalities of network, allows that the configuration of conditions required by PPDR missions are properly configured in the 5G network. Such configuration comprises other components in the network, like the Network Repository Function (NRF), the Policy Control Function (PCF), the Access and Mobility Management Function (AMF) for the enforcement of network policies in the 5G network. As an example, the components of the architecture of PPS2 allow the configuration of priority flows that can be used by PPDR devices (e.g. sensors) to transmit information related with safety of PPDR agents (firefighters, police).
2. Multi-vendor support and interoperability between different products. The compliance with 5G standards (TS 23.501 [5], TS23.502 [9])avoids the vendor lock-in. Platforms of different PPDR organizations can be easily integrated with 5G, considering the exposure of functions to configure the network. Whether this configuration is performed if a 5G core provided by Nokia or other vendor will not constitute an issue due to the interoperability and conformance with 5G standards.
3. Security of critical components. The “openness” of the 5G core through the NEF element, poses by itself additional security risks. The security framework, part of the architecture of PPS2, aims to mitigate such risks, by detecting possible attacks of Application Functions and applying policies to reduce the effects of such attacks and to prevent the same.

4.2.2 Message signaling charts

This subsection presents the message flows associated with the PPDR use case, all the signaling flows are summarized in the figure below.

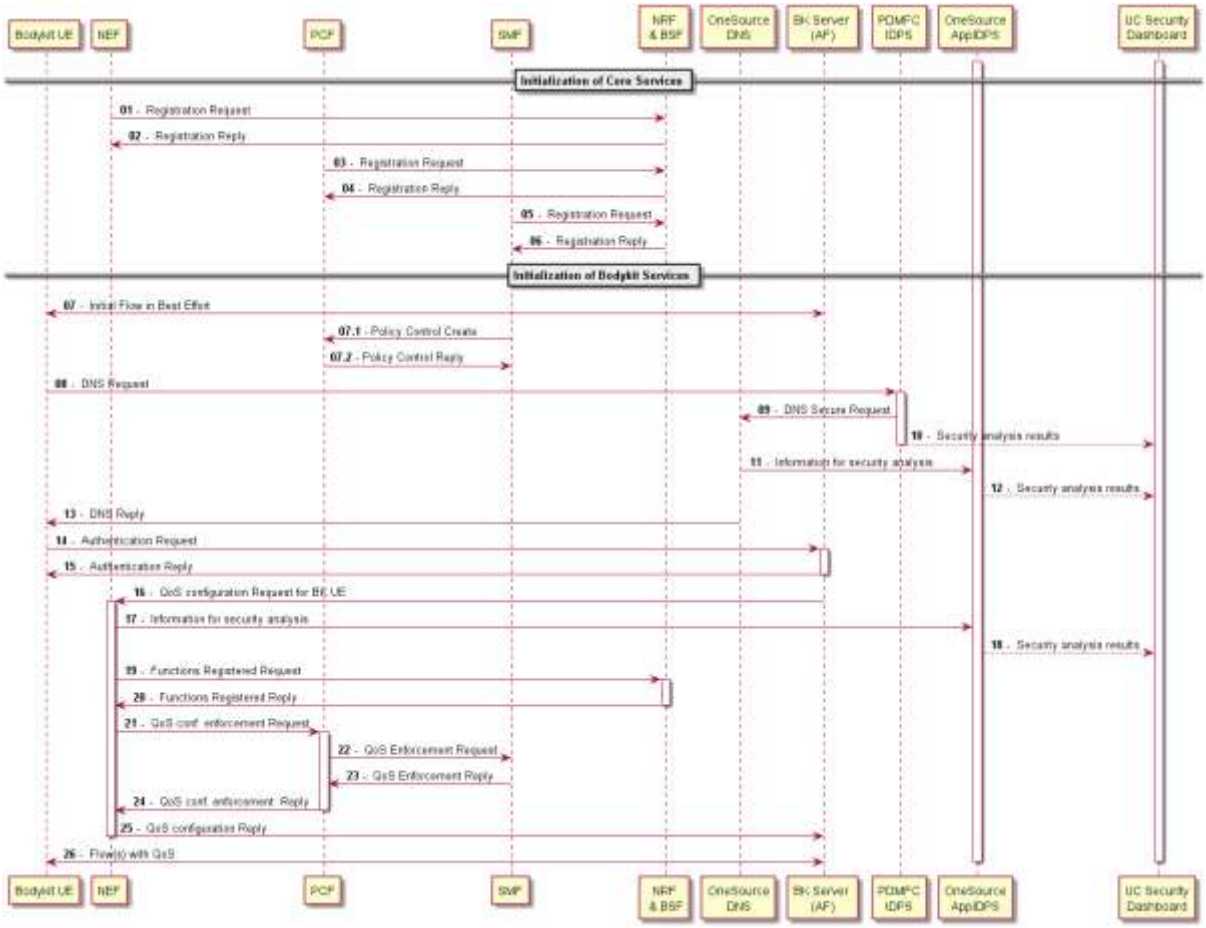

Figure 31 – Use Case 2 Sequence Diagram

As depicted in the figure, two main phases are included in the diagram.

1. Initialization of core services, in this phase the 5G core services are properly configured. For instance, they include the registration of PCF, NEF, in the NRF & BSF components;
2. Initialization of Bodykit Services, in this phase the services related with PPDR IoT (Public Protection and Disaster Relief Internet of Things) are properly configured.

The steps involved in the signaling are described in the table below.

Table 25 – Use Case 2 - Sequence Diagram Steps

Step Id	Step's Description	Mandatory/ Optional
1	Registration request of NEF in NRF (as in TS 23.501) [5]	Mandatory
2	Registration reply from NRF (as in TS 23.501) [5]	Mandatory
3	Registration request of PCF in NRF (as in TS 23.501) [5]	Mandatory
4	Registration reply from NRF (as in TS 23.501) [5]	Mandatory
5	Registration request of SMF in NRF (as in TS 23.501) [5]	Mandatory
6	Registration reply from NRF (as in TS 23.501) [5]	Mandatory

Step Id	Step's Description	Mandatory/ Optional
7	This step assumes that the Operator has a configuration in Best Effort for initial message flows.	Mandatory
7.1	With the BodyKitUE attach to the network the SMF request the creation of a PolicyControl session with the PCF.	Mandatory
7.2	The PCF replies to the SMF with the QoS parameters to be associated to that user (Best Effort for default traffic).	Mandatory
8	The BK devices requests IP from DNS Server (Through the IDPS).	Mandatory
9	The IDPS protecting DNS receives the request and starts the security analysis, then forwards the request to DNS server.	Optional
10	The IDPS sends the analysis results to the Security Dashboard.	Optional
11	The DNS Server sends information to the AppIDPS for security analysis.	Optional
12	The AppIDPS sends information to the Security Dashboard regarding the security analysis performed.	Optional
13	DNS reply to the original DNS request (information of BK server).	Mandatory
14	With the address of the BK server, the BK devices initiates the authentication process.	Mandatory
15	The BK server replies to the BK device, regarding the authentication request.	Mandatory
16	The BK server triggers the request towards NEF to perform the configuration of the flows of the BK device, previously registered.	Mandatory
17	NEF receives a request for QoS configuration and sends information to the AppIDPS to detect malicious AFs.	Mandatory
18	AppIDPS sends information regarding the analysis of NEF information.	Optional
19	NEF requests information to NRF, regarding the registered AFs and Network Functions.	Mandatory
20	The NRF provides the requested information. For instance, the PCF to contact in order to enforce the QoS settings in the network.	Mandatory
21	NEF contacts the appropriate PCF.	Mandatory
22	The PCF sends to SMF an update to the PolicyControl context.	Mandatory
23	The SMF responds to the PCF with the result of the PolicyControl update requested.	Mandatory
24	PCF informs NEF regarding the QoS configuration.	Mandatory
25	NEF informs the BK server (i.e. the AF that requested the QoS configuration).	Mandatory
26	The necessary flows for the BK device are now configured with the appropriate QoS settings.	Mandatory

5 Deployment View

This section describes the logical and physical run-time environment available to later deploy the products.

Because 5G is connected with network virtualization, virtual functions, automation and management, the choice of Openstack for the infrastructure management layer, was identified as the best solution. Openstack platform brings the standard and API to the platform management. The main features of Openstack are as follows:

- Isolation and multi-tenancy
- Management APIs
- Cloud native, elasticity, flexibility
- Standard API for VIM
- Manages compute, storage and network
- Multiple SDN integrations
- Edge computing
- Multi-cloud management

5.1 Virtual Infrastructure Manager

The role of the virtual infrastructure manager is to configure the compute, hypervisor and infrastructure network domains. For that different projects are used:

- **Nova −** provides a way to provision compute instances
- **Ceilometer −** collect, normalise and transform data produced by OpenStack services
- **Neutron −** SDN networking project focused on delivering networking-as-a-service (NaaS) in virtual compute environments
- **Ironic −** implement services and associated libraries to provide massively scalable, on demand, self service access to compute resources, including bare metal, virtual machines, and containers
- **Glance −** image services include discovering, registering, and retrieving virtual machine images
- **Cinder −** Block Storage service for OpenStack
- **Trove −** database-as-a-service provisioning relational and non-relational database engines.
- **Keystone −** OpenStack service that provides API client authentication, service discovery, and distributed multi-tenant authorization by implementing OpenStack's Identity API
- **Heat −** orchestrates the infrastructure resources for a cloud application based on templates in the form of text files that can be treated like code
- **Horizon −** implementation of OpenStack's dashboard, which is extensible and provides a web based user interface to OpenStack services

The use of this projects are represented in the following image:

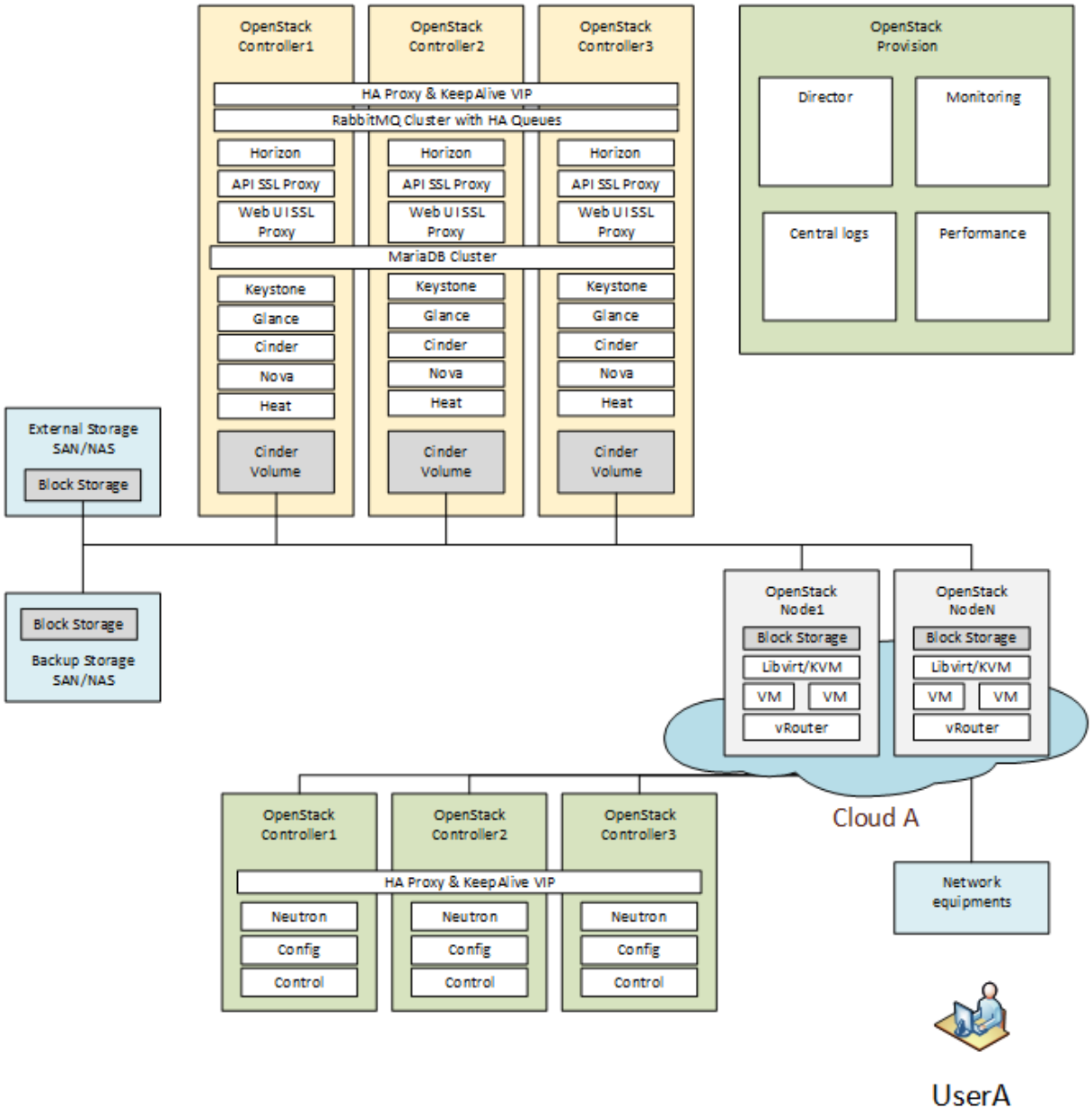

Figure 32 –Open Stack Architecture

There are three main interfaces for VIM - Nf-Vi, Vi-Vnfm and Or-Vi:

- Nf-Vi: this is the reference point between the management and orchestration agents in compute domain and the management and orchestration functions in the virtual infrastructure management (VIM). It is the part of the Nf-Vi interface relevant to the compute domain.
- Vi-Vnfm: this is the reference point used for exchange of information elements between the Virtualized Infrastructure Manager (VIM) and VNF Manager (VNFM)
- Or-Vi: this is the reference point is used for exchanges between NFV Orchestrator and VIM

5.2 VNF Manager

The VNFM is the responsible for the life-cycle management of VNFs over the VIM. VNFM operations include:

- VNFs deployment
- VNFs scaling
- VNFs updating and upgrading
- VNFs termination

There are four main interfaces fo VNFM - Vi-Vnfm, Or-Vnfm, Ve-Vnfm-em and Ve-Vnfm-vnf:

- Vi-Vnfm: this is the reference point used for exchange of information elements between the Virtualized Infrastructure Manager (VIM) and VNF Manager (VNFM)
- Or-Vnfm: this is the reference point used for exchanges between Network Functions Virtualization Orchestrator (NFVO) and Virtualised Network Function Manager (VNFM)
- Ve-Vnfm-em: this is the reference point used for exchanges between EM and VNF Manager
- Ve-Vnfm-vnf: this is the reference point used for exchanges between VNF and VNF Manager

6 Conclusions

This document presents the definition of the architecture of products and services for the 5G Core network, and serves as the basis for the elaboration of the implemented solutions (future D2.4) in the various prototypes (future D2.3).

7 Bibliography

[1] Grant Lenahan, Appledore Research Group, “Closed Loop Automation and the New Role of Assurance,” [Online]. Available: http://appledoreresearch.com/product/closed-loop-automation-new-role-assurance-research-note/.

[2] 3GPP, “3GPP TS23.501, "System Architecture for the 5G System",” [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144.

[3] “Deliverable 2.1 - Use cases and requirements for solutions targetting 5G network core,” 2018. [Online]. Available: https://wikis.ptinovacao.pt/download/attachments/44648131/Deliverable_21_final.pdf?api=v2.

[4] “3GPP 5G System Architecture,” [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144.

[5] “Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper,” [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html#_Toc503317524.

[6] “3GPP TS 22.261, “Service requirements for next generation new services and markets",” [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3107.

[7] RFC Series, “RFC 8328”, “Policy-Based Management Framework for the Simplified Use of Policy Abstraction (SUPA),” March 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8328.txt.

[8] 3GPP, “3GPP TS 23.503, "Policy and Charging Control Framework for the 5G System Stage 2 (Release 15) V15.3.0 (2018-09)",” [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3334.

[9] 3GPP, “3GPP TS 29.510, “Network Function Repository Services Stage 3 (Release 15) V15.1.0 (2018-09)”,” [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3345.

[10] 3GPP, “3GPP TS23.502, "Procedures for the 5G System",” [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3145.

[11] “Kafka, Apache Kafka documentation,” [Online]. Available: http://kafka.apache.org/intro.

[12] “StackShare. "RabbitMQ vs. Kafka vs. NSQ". Messaging systems comparison,” [Online]. Available:

https://stackshare.io/stackups/kafka-vs-nsq-vs-rabbitmq.

[13] “Google Trends, "RabbitMQ, Kafka, NSQ, ZeroMQ, and ActiveMQ Messaging Systems Trends", 2018.,” [Online]. Available: https://trends.google.com/trends/explore?date=2017-07-16%202018-07-16&geo=US&q=RabbitMQ,Kafka,NSQ,ZeroMQ,ActiveMQ.

[14] “Present Pivotal Software. RabbitMQ. Features.,” [Online]. Available: http://www.rabbitmq.com/features.html.

[15] “Present Pivotal Software. RabbitMQ Protocols.,” [Online]. Available: https://www.rabbitmq.com/protocols.html.

[16] “Present Pivotal Software. RabbitMQ Exchanges.,” [Online]. Available: https://www.rabbitmq.com/tutorials/amqp-concepts.html.

[17] N. Nannoni, Message-oriented middleware for scalable data analysis architectures, 2015.

[18] A. Richardson, “Confluent community,” [Online]. Available: https://slackpass.io/confluentcommunity.

[19] L. Toledo, *A Distributed Platform for Security Event Handling in Industrial Control Networks,* MSc thesis, 2018.

8 Annexes

8.1 Revised 3GPP TS 22.261

What is supposed to be accomplished by this project has been highlighted in green, what is not expected to be fulfilled has been highlighted in red.

Due to the huge volume information, this 3GPP TS 22.261 revised version document is going to be provided separately as an external annex (22261-g50 - 5G service req - Rev1.docx).

8.2 Kafka analysis

Since Apache Kafka was the selected bus to provide centralized communication between all proposed solutions in this architecture, the next chapters provide a analysis of this tool and all it's potential to be used on the 5G environments.

The Apache Kafka is a distributed streaming platform, which includes a dedicated message queueing, a publisher-subscriber pattern, and consumer groups. Generally, Kafka is used in two main types of scenarios: to build real-time streaming data pipelines that reliably move data between systems or applications, and to build real-time streaming applications that transform or react to the streams of data [10]. The main goal is to send data from producers to consumers. The former send data to topics and the latter consume data from it. These can be grouped into consumer groups to distribute the consumer load as a message is delivered to one of the consumers within a group, providing scalability and fault tolerance.

As a distributed platform it runs on a cluster, which can be hosted on one or more servers and span over multiple geographically dispersed datacentres. The stream of records stored in a Kafka cluster can be divided into multiple categories (the previously mentioned topics). Moreover, each of those records consist of a key, a value and a timestamp. The log of the topic can be divided into several partitions. These partitions provide several functions. For instance, they allow a log to scale beyond the size that would fit a single server. Each partition must fit in one server, but a log can be composed by several partitions, allowing it to scale greatly in size. Partitions also play a role in the distributed and fault tolerance characteristics, which will be further detailed below on this section.

Kafka topics have a stream of records that are multi-subscriber, which means that can have zero, one or many consumers subscribing the topic. The Kafka cluster maintains a partitioned log that is sequentially ordered, which is immutable. Each record is assigned a unique sequence id number (called offset). The records are then stored persistently during the retention period either if they have been consumed or not. Other than the records' keys, values and timestamps, the only information stored is the offset (or position) of each topic consumer. This offset represents the position of the consumer in that log and is controlled by the consumer itself. Moreover, the consumer as total control of this offset, meaning that it can jump freely between positions in the log.

When used as a messaging system, it can be configured to act in different ways, depending on the scenario needs. In a message queueing system, it can be configured for the messages to be deleted after being consumed. However, if used as a streaming platform, messages can be retained using policies based on time or available storage space.

Next figure illustrates the main components of the Kafka architecture. On the top the Kafka Cluster is composed by two Servers, each with two Partitions. The bottom row is composed by two Consumer Groups, each with several Consumers ranging from one to six.

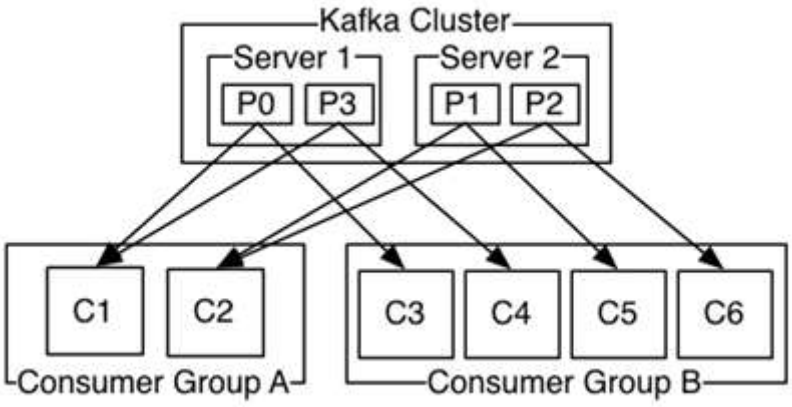

Figure 33 – Kafka main concept and components [10]

The distributed characteristics of Kafka are quite important to its usage during this project. Being a distributed platform, it can support the intrinsic distributed nature of the components that are in a 5G architecture. Kafka distributes the partitions of the log over the servers in the Kafka cluster, with each server handling part of the partitions. On the other hand, each partition can be replicated over several servers for fault tolerance purposes. The partition follows a hierarchy in which one server is a leader, and zero or more servers act as followers. The leader is responsible for handling all reads and writes. If it fails or crashes, then one of the followers will be assign the leader role in the cluster. A server can be leader and follower of different partitions simultaneously, distributing the leadership load across a cluster of servers. Kafka also supports geo-replication through MirrorMaker, replicating messages across datacentres or cloud domains.

Comparison with related messaging architectures

There are several messaging systems available, each with its own purpose and specific application scenarios. A quick look at the interest shown in some of the available options namely Kafka, RabbitMQ, NSQ, ZeroMQ, and ActiveMQ, as can be seen in next Figure, shows Kafka as the most popular messaging system, followed by RabbitMQ. The remaining ones have residual interest when comparing with Kafka and RabbitMQ.

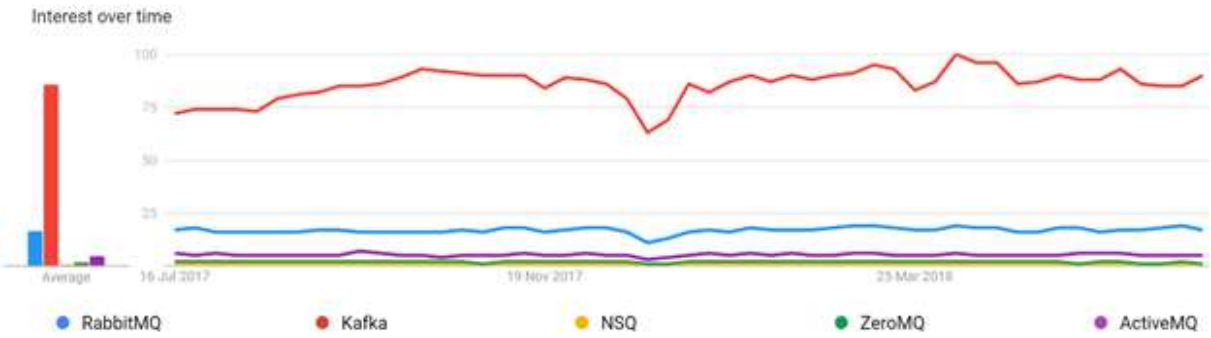

Figure 34 – Interest shown in messaging systems [11], [12]

Due to the difference of interest between the two main options and the remaining ones, only RabbitMQ will be compared with Kafka.

RabbitMQ [13] is a popular message broker that allows the exchange of information between applications. It implements the well-known Advanced Message Queueing Protocol (AMQP). Although

the initial protocol used by RabbitMQ was AMQP, it evolved to support several other messaging protocols. However, for the general use of RabbitMQ the AMQP is their recommended protocol. The full list of protocols supported by RabbitMQ follows [14]:

- **AMQP 0-9-1** and **0-8 −** Original protocol that served as base for RabbitMQ
- **AMQP 1.0 −** New standard for AMQP, radically different from previous AMQP 0-9-1 and 0-8
- **STOMP −** Text-based messaging protocol focused on simplicity
- **MQTT (Message Queuing Telemetry Transport) −** Binary protocol for lightweight publish/subscribe messaging, focused for use on constrained devices
- **HTTP −** RabbitMQ transmits messages over HTTP using plugins in three ways: using the management plugin; using the Web-STOMP plugin to send STOMP messages using WebSockets; using the JSON-RPC (JavaScript Object Notation-RPC) channel plugin to send AMQP 0-9-1 messages over JSON-RPC.

RabbitMQ allows the connection of different applications in different organizations and geographic locations. Also, it works as an asynchronous messaging protocol, meaning that the sending of the message is separated from its consumption. There are two main types of communication:

- **Queues −** It's the simpler type as it allows communication between a producer and a consumer asynchronously, providing a buffer for the messages. The producer sends messages directly to a queue.
- **Exchanges −** It is a more complex type and representing the full messaging model of RabbitMQ. The exchange is an endpoint where producers send messages. The exchange will then forward those messages to one or more queues (or it can also discard it). Exchanges can be of four types, which will control the message routing: direct, topic, fanout, and headers. More detailed information about the difference between the exchange types can be found on [15].

As a comparison, the main characteristics of Kafka and RabbitMQ are listed in Table 1. Nannoni [16] separates RabbitMQ and Kafka in their base concept, with RabbitMQ being queue-based and Kafka being a message streaming system. Based on Table 1, the two systems differ in some characteristics such as the main medium for main storage with Kafka using Disk-based storage and RabbitMQ storing on RAM. Also, in a high-rate message scenario Kafka provides better availability and integrity than RabbitMQ [16].

Table 26 – Apache Kafka and RabbitMQ feature comparison [14]

	Apache Kafka	**RabbitMQ**
Creation Year	2011	2007
Licence	Apache (open-source)	Mozilla Public Licence (open-source)
Documentation with examples and community	Quality documentation and a useful Slack community [17]	It has mature documentation and examples
Main Storage	Disk	RAM
Order Storage	At partition level	N/A
Queue data persistent	Mandatory	Temporary and optional
Message deletion triggering	After exceed size or time limit	Immediately when a consumption is confirmed
Queueing data compression operation	Allowed to use	Not allowed to use

	Apache Kafka	RabbitMQ
Authentication using SSL	Allowed to use	Allowed to use
Asynchronous publishing	Allowed to use	Not allowed to use
Batch fetch	Allowed to use	Not allowed to use
Delivery Guarantees	Allowed to use	Allowed to use
Message reply capability by consumer	Allowed to use	Not allowed to use
Consumer confirms with acknowledge	Using offset commits	Allowed to use
Message Rejected or Requeuing by consumer	Not allowed to use	Allowed to use

Authors list

Partner	Author
Altice Labs, S.A.	Rui Calé, Miguel Santos
Altran	Sérgio Figueiredo, Bruno Parreira
IT	Carlos Senna, Susana Sargento
IT Center	Rui Teixeira
Nokia	Hugo Vieira, João Fragoso Rodrigues, José Albino
Onesource	André Gomes, Bruno Sousa, Luís Cordeiro, Pedro Silva, Vitor Fonseca
PDMFC	Carlos Carvalho
Ubiwhere	Luís Conceição, Ricardo Preto, Tiago Teixeira
Universidade de Coimbra	Tiago Cruz, Jorge Proença, Vasco Pereira

Versions history

Version	Date	Description
0.1	30-10-2018	First version including all partners' contributions.
0.2	12-11-2018	Addition of 2.3, replacement of DNS IPDS picture, Glossary revision
0.3	15-11-2018	Addittion of 3.2.1 General revision based on initial the feedback received from partners
1.0	13-12-2018	Final version General revision based on the final feedback received from partners