H2020 5GASP Project

Grant No. 101016448

D4.2 Final Methodology for Designing and Developing NetApps

Abstract [YoGoKo]

Alongside the 5GASP project Architecture and NetApp's definition, the consideration to provide a pre-defined design and development methodology was one of the main focuses in the WP4 to introduce a generic PoC NetApps for the three main verticals (Automotive, PPDR, and cross-vertical). As the approved methodology is the founding pillar of the envisioned 5GASP framework, the progress in both NetApps certifications and validation will depend on this deliverable outcome where all design and development of the NetApps are defined. The definition of a commonly agreed methodology was initially described in D4.1 [1] and consequently progressed to the final version in this deliverable as an answer to the involvement and feedback of different SMEs that correspond to the three main verticals.
This document details the final methodology and the onboarding procedure and provides a Proof-of-concept (PoC) NetApp example to facilitate the development and deployment at the 5GASP portal. Moreover, the document further summarises the up-to-date status of the evolved NetApps within the 5GASP project.

Document properties

Document number	1
Document title	D4.2 Final Methodology for Designing and Developing NetApps
Document responsible	
Document editor	Mohammad-Yakub Abualhoul
Editorial team	YoGoKo
Target dissemination level	
Status of the document	Internal Review ready
Version	6

Document history

Revision	Date	Issued by	Description
1.0	19/10/2021	YoGoKo	Initial draft
1.1	17/12/2021	YoGoKo	Review version
1.2	28/12/2021	Neobility, DriveU	Internal review
1.3	31/12/2021	YoGoKo	Submission version

List of Authors

Company	Name	Contribution
YoGoKo	Yakub Abualhoul, Emmanuel Thierry	Abstract, Introduction, Development and Onboarding, Generic PoC NetApp
OdinS	Jorge Gallego-Madrid Ana Hermosilla Antonio Skarmeta	NetApp Development, NetApp Management, NetApp Design, Development and Onboarding exemple, Security considerations

Lamda Networks	Leonidas Lymberopoulos Angeliki Kavvalou	Sections 3.1.1, 3.2, and 4.2.4 .
ORO	Elena-Madalina Oproiu Oana Badita Ioan Constantin Andreea Bonea Marius Iordache	Architecture, Integration in 5GASP Architecture
UoP	Christos Tranoris Kostis Trantzas Ioannis Chatzis	Onboarding configuration, Ecosystem Workflow, NetApp Management
BLB	Roman Odarchenko, Yevgeniya Sulema	NetApp Testing and Validation Lifecycle
ININ	Luka Koršič Janez Sterle	PPDR PoC NetApp design, development, onboarding

Disclaimer

This document has been produced in the context of the 5GASP Project. The research leading to these results has received funding from the European Community's H2020 Programme under grant agreement number 101016448.

Contents

List of Figures

List of Tables

List of Acronyms

5GASP	5G Application & Services experimentation and certification Platform
API	Application Programming Interface
BW	Bandwidth
CI/CD	Continuous Integration and Continuous Deployment
CNF	Cloud-native Network Function
DevOps	Development and Operations
DSP	Digital Services Provider
GST	Generic Network Slice Template
ITS-S	Intelligent Transport Systems Station
KPI	Key Performance Indicator
MNO	Mobile network operator
NEST	Network Slice Type
NetApps	Network Applications
NFV	Network Function Virtualization
NFVI	Network Functions Virtualization Infrastructure
NFVO	NFV Orchestrator
NS	Network Services
NSD	Network Service Descriptor
NSP	Network Service Provider
OSM	Open-source MANO
PLR	Packet Loss Ratio
PNFD	Physical Network Function Descriptor
RT	Real-Time
SMEs	Small and Medium-sized Enterprises
V&V	Validation and Verification
V2C/C2V	Vehicle-to-Cloud / Cloud-to-Vehicle
VM	Virtual Machine
VM	Virtual Machine
VNF	Virtual Network Function
VNFD	Virtual Network Function Descriptor
5GASP	5G Application & Services experimentation and certification Platform

Definitions

This document contains specific terms to identify elements and functions considered mandatory, strongly recommended or optional. These terms have been adopted for use similar to that in IETF RFC2119 and have the following definitions:

- **MUST** This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- **MUST NOT** This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
- **SHOULD** This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a specific item. Still, the full implications must be understood and carefully weighed before choosing a different course.
- **SHOULD NOT** This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the individual behaviour is acceptable or even beneficial. Still, the full implications should be understood, and the case carefully weighed before implementing any behavior described with this label.
- **MAY** This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product. In contrast, another vendor may omit the same item. An implementation that does not include a particular option MUST be prepared to interoperate with another implementation that includes the option, perhaps with reduced functionality. In the same vein, an implementation that includes a particular option MUST be prepared to interoperate with another implementation that does not include the option (except, of course, for the feature the option provides).

1. Introduction

The main objective of D4.2 is to utilize the SMEs feedback to improve the initially proposed methodology, which was defined respecting the requirements introduced in WP2 (the ETSI standard of OSM/SOL006 (YANG model), as well as ONAP/SOL001), and draw a final commonly agreed and unified methodology. This final methodology aims to provide ready-to-use PoC NetApps and define vertical-specific 5G facilities to expedite the creation of dedicated NetApp that are ready to be automated and integrated efficiently. Moreover, considering the proposed methodology and the provided PoC NetApps will also assure compatibility with the envisioned standardized certification and CI/CD processes and enable the optimal vertical-specific 5G facilities selection by following standard interfaces on NFV and 5G solutions.

The unified final methodology provides guidelines to further assist NetApp's owners toward solutions such as the model-transformation service (detailed in WP3). It offers the feasibility to propose a customized assessment process and KPIs. This deliverable describes both generic and vertical-based high-level onboarding procedures, showing the workflow of this process from the design phase to the onboarding and testing of the NetApp.

The procedure deploys a NetApp lifecycle is detailed in this deliverable, where each phase of the process highlights the design and development phases and is supported by a PoC NetApp. Considering the feedback from some of NetApp's owners on the onboarding approach, the methodology will describe how the NetApps will be managed in the 5GASP platform and the categorization (either VNFs or CNFs). However, it is essential to mention that the associated tests, validations, and certifications procedures are out of this document's scope.

Finally, three NetApps PoC examples are presented in the methodology and consider the three main verticals and the associated facilities, detailing the used descriptors, design and development to the onboarding steps, followed by all contributing NetApps up-to-date status.

1.1. Document structure

This document is organised into six different parts. First, the introduction provides a brief insight into the 5GASP proposed methodology for interoperable NetApp's design and development. The last part concludes the methodology with an up-to-date status of all 11 developed Netapp within 5GASP. Consequently, section two proposes the overview of the recommendation to properly onboard a NetApp considering the ecosystem workflow and the architecture integration and description of the onboarding components. Next, more details on the NetApp design and development methodology are presented in section three, describing the NetApp onboarding life cycle and the design and development phases. Section 4 is dedicated to the NetApp onboarding approach, presenting the NetApp management procedure. The NetApp categories are differentiated in this section, including templates for the three descriptors that will define a NetApp. Finally, three PoC NetApp designs are reported in section 4, representing an adaptation for the proposed methodology to design, develop, and test NetApps for the considered three verticals.

2. NetApp onboarding overview

This section provides an overview of the NetApp onboarding procedure proposed by the 5GASP project, describing and indicating the location of the onboarding process in the 5GASP architecture and the proposed workflow.

2.1. Architecture

The 5G Network architecture related to the NetApp's onboarding procedure, described as a simplified architecture proposed in figure 1 of the deliverable number D4.1 [1] "Initial Methodology for Designing and Developing NetApps" of the current project, shows a specific layer that corresponds of a step where the procedure can be described in the entire format with respect to final methodology.

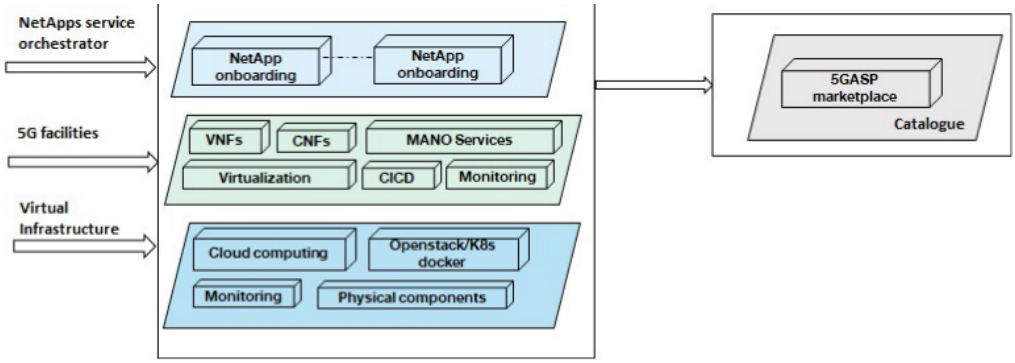

Figure 1. 5G Simplified Architecture for NetApps onboarding

To describe such a process, it is mandatory to have in place essential components from the ecosystem: hardware infrastructure, virtual infrastructure with all elements ready for use: resource allocation, security rules, resource monitoring, APIs, users' access, and a repository with all packages, accessible using a 5GASP marketplace that represents the catalogue exposed to the portal user.

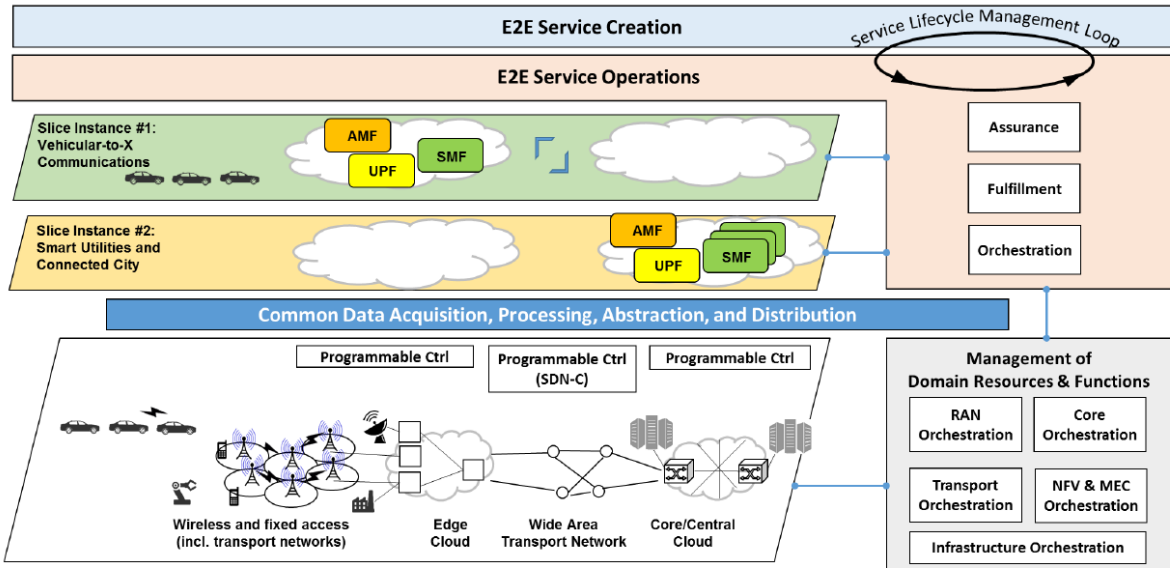

Figure 2. 5G-PPP Overall E2E 5G Architecture

The Architecture in 5GASP should support the NetApps DevOps functionality, which 5G architecture should integrate and which is also described in the D4.1 deliverable [1]. For each step of instantiation, it should contain in 3 DAYs operations, the entire network setup should comply with some onboarding requirements, as ETSI [2] describes:

- **Day 0 operation**: description of each VNF component, defining NFVI requirements (resources for every component, rules, particular parameters), topology definition, the location needed to be deployed and need or must run, images of the software and other files.
- **Day 1 operation**: specific parameters that need to be modified and required by a specific service or environment.
- **Day 2 operation**: defining remaining configurations for runtime operations and linking all components, KPIs and monitoring parameters, closing the loop.

The proposed 5GASP Architecture is designed to support the NetApps services operations, the Life Cycle Management and the service RUN time configuration that is supported by the architecture, focusing on:

- Management and orchestration, services NetApps orchestration
- Programmable network, SDN/NFV and automation service deployment
- 5G SA network
- 5G virtualised infrastructure (e,g OpenStack, K8s)

- E2E service slice deployment
- Services and Infrastructure monitoring
- Secured architecture

2.1.1. Integration in 5GASP architecture

The NetApp onboarding architecture is providing to the NetApp Developers the framework for the domain requirements VNFication, developments and adaptation of onboarding procedures of the Applications on the 5GASP platform and targeted facilities, within the integration for:

- NetApp's onboarding to the NetApps catalogues
- CI/CD framework testing pipeline for the target facility
- Target facility deployment (Service Orchestration, CI/CD pipelines)
- NFVO, ETSI MANO onboarding and services information
- NetApps validation process

From the architectural point of view, OSMv10 [3] is the leading actor that makes onboarding possible in the 5GASP framework, taking the role of orchestrator. From an architectural view, the OSM is the integration point between the upper 5GASP service layer and the project facilities. OSM is connected to VIM accounts/tenants such as Openstack, Kubernetes. Providing the VNF or CNF descriptors starts the configuring activities of the VNFs/CNFs in corresponding NFVI.

The onboarding workflow between the different 5GASP architectural components (Services and facility orchestrators, testbeds, NetApps) is presented in Figure 3, with respect to workflow implemented for the onboarding and connection to CI/CD services, a fully interactive process from the NetApp developer point of view, configuring and testing a NetApp step by step within the target testbed.

The relevant OSM ETSI SOL 005 interface is exposed to the service orchestrator on the server and further on the chain of architectural elements, making calls to VIM APIs of Openstack or Kubernetes (depending on the need: VNF or CNF).

Overall, the day two configurations and LCM are maintained by CI/CD framework, which also is fed with infrastructure and service information from monitoring APIs.

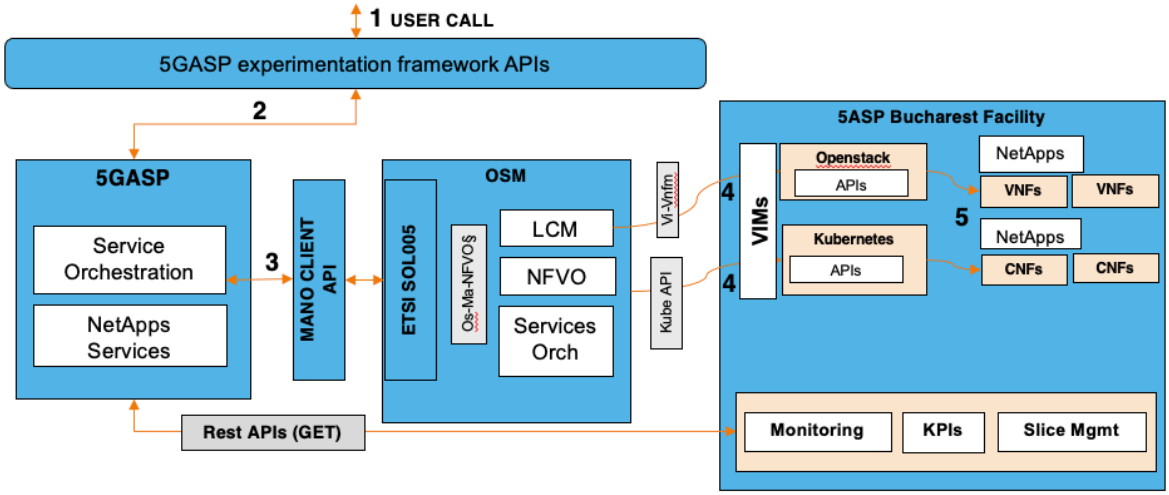

Figure 3. 5G-ASP NetApps integration architecture and workflows

2.1.2. Onboarding configuration

In the context of the 5GASP project, NetApps will be described as an aggregation of Virtual Network Function Descriptors (VNFDs) and eventually forming one or more Network Services Descriptors (NSDs). The YANG [4] model will be initially supported as OSM-native, and by the later stages of the project, TOSCA [5] model is also expected to be supported. The resource aspects of the NetApps, along with their respective network requirements, are modeled as TMForum's Resource Facing Service Specifications (RFSSs) [6]. As NFV architecture proceeds towards a more "Cloud-native" approach, 5GASP ultimately aims to support the definition of NetApps as CNFs and simultaneously provide effortless transformation for already containerized applications to VNFs via Kubernetes Helm Charts [7]. An extensive description of the NetApps' onboarding configuration along with the rest of the unified standards-based model introduced in 5GASP can be found in D3.1 [8].

2.2. Security considerations

Security aspects during the onboarding procedure need to be considered, although it is not covered specifically by 5GASP objectives. A state-of-the-art analysis will be performed targeting this VNF onboarding security concerns.

ETSI has examined initial considerations about this aspect in [9], in which they have considered the problem from the VNFD integrity and confidentiality point of view. ETSI considers that the VNF Package security validation check during the onboarding is crucial for the successful deployment of VNFs. In this way, ETSI proposes a mechanism to improve security during this process. Each item included in the VNF Package (VNFD, images, scripts, etc.) shall have a cryptographic signature when stored in order to assure its integrity. Thus, the NFV-MANO will verify the signatures (they will be signed by the VNF provider, as their creator) and then sign the whole VNF Package again when storing it. In this way, the VNF Package authenticity and integrity during VNF instantiation is granted, as the Package integrity was ensured during the onboarding due to the signature validation process.

The ETSI specification also includes confidentiality considerations for the storage of the VNFs in the corresponding catalogues. Once the VNF provider's signature verification of the VNF package has been performed, the NFVO shall encrypt the VNF package artefacts using the appropriate encryption keys that the service provider may provide. Then these encrypted packages shall be stored in the catalogue. The encryption mechanisms are not described in the document. The main objective of this procedure is to provide confidentiality protection to the VNF packages during the instantiation. Before this step, if the service provider policy considers confidentiality, then the VNF artefacts shall be encrypted in the storage and decrypted before their instantiation. After the decryption, the signatures shall be verified.

The implementation of the ETSI proposal is currently being studied and analyzed to check if it is feasible to use it or if it is only in its conception stages, as it is only described from a high-level perspective without giving technical details. For example, at the moment, current versions of OSM does not include this functionality. Also, we are analyzing whether changes or additions could be made to them to support this new specification.

In summary, the 5GASP consortium is exploring security considerations in the onboarding of NetApps, analyzing if it is possible to handle the NetApp onboarding integration and confidentiality; if that is the case, how could it be done following the existing standards.

2.3. Ecosystem workflow

The onboarding procedure in the 5GASP system involves the uploading to the 5GASP portal of the NetApp by means of the triplet of descriptors that defines it. The NetApp developer performs the onboarding itself (or by a NetApp experimenter), and the interaction is done against the unique 5GASP portal.

The process will be totally interactive with the user, requiring more actions in the first stages and evolving towards a more automatic procedure later in the project. This is because the automated functions will be integrated gradually.

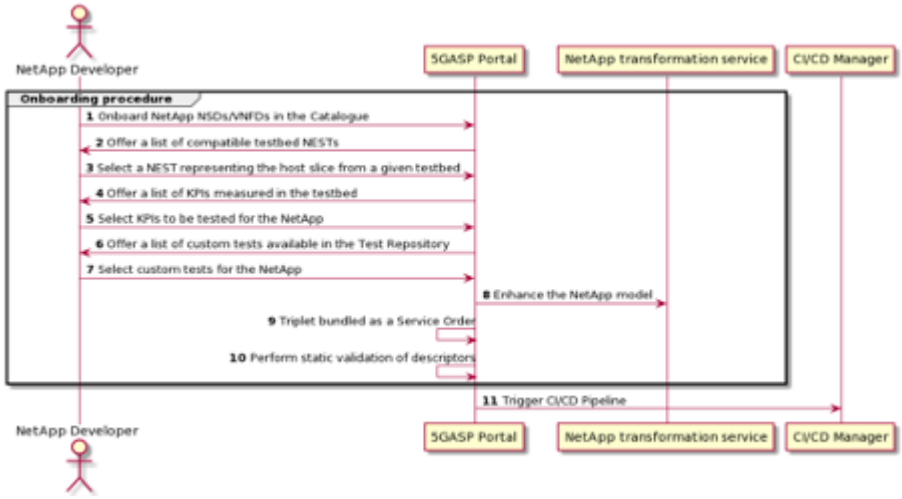

Figure 4. Onboarding and connection to CI/CD service

The onboarding procedure is depicted in Figure 4. This figure shows the different steps that conform the onboarding process in the initial stages of the 5GASP project, in which the NetApp descriptors will be uploaded individually. By doing so, the developer will be able to interact with the 5GASP project to configure different NetApp and testing parameters in each step.

It all starts with uploading the NetApp NSDs/VNFDs, which will be stored in a catalogue. Then, based on the submitted NSDs and their respective network requirements, the developer should select the test site that fulfills the latter. In advanced project phases, the portal could compare the network requirements with the capabilities offered by the multiple facility test sites, filtering out the unfitting ones or even automatically selecting the ones that match the criteria of the NetApp. The list of test sites is shown to the developer as a list of NESTs representing each facility. Each one defines the host slice that will be reserved and eventually deployed if the site is finally selected. The developer chooses one of them, and this information is stored together with the NetApp NSDs/VNFDs. Alternatively, the developer can choose among a list of pre-defined NESTs offered by the 5GASP portal, the one best suited to the NetApp. Based on this selection, the 5GASP portal provides the developer with a list of KPIs that can be measured in the NetApps deployed in the corresponding facility test site. Then, the portal offers a list of custom tests available in the Test Repository, which the developer may select to execute against the NetApp. These custom tests can be pre-defined and available in each facility. In later project phases, the developer of the NetApp, apart from the default testbed tests, should be able to upload its own test suites (in the form of custom test scripts or test VNFs) that could extend the available test cases to be run against.

At this point, the NetApp is sufficiently defined in the 5GASP portal, and it is composed of the three descriptors: **(i)** the NSDs/VNFDs, **(ii)** the NEST, and **(iii)** the tests descriptors. Now, the descriptors will be sent to the NetApp transformation service, which will automatically enhance the descriptors, looking for bad practices, e.g. syntax errors, simultaneously trying to correct them. Once this is done, the triplet of descriptors can be bundled as a TMF Service Order [10] and pass a static validation of the descriptors. If succeeded, the onboarding procedure is complete.

Finally, the 5GASP portal can trigger the CI/CD pipeline interacting with the CI/CD Service Manager and send the required information for the deployment and the testing. The details about these subsequent testing phases of the 5GASP workflow are covered by WP5 and will be initially examined in D5.1. [11]

3. NetApp design and development methodology

This section details the updates in the first version of the NetApp design and development methodology [12], which the 5GASP project partners have already followed. The objective of the improvements is motivated by the need to bring the NetApp developers, verticals, customers, etc., closer and to resolve some gaps identified during the implementation of the pre-defined methodology. In this way, the new updated version of the methodology will further reduce the time and cost for making new 5G NetApps or adapt existing ones to these new technologies.

3.1. Lifecycle

The main stages of the overall 5GASP lifecycle remain the same as in the previous version. The lifecycle was identified as a process used by the NetApp developers to design, develop, and test high-quality NetApps. 5GASP lifecycle contains four main phases: design, development, testing, and validation (see Figure 5).

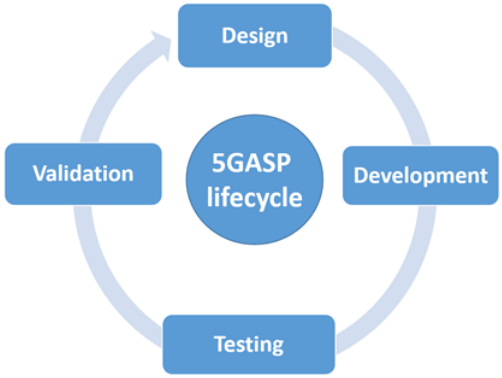

Figure 5. 5GASP lifecycle

A high-level description of these phases is presented in the sections below. And, in more detail, some of these phases were already described in other deliverables, and some of them only will be defined and described in the appropriate WPs and deliverables accordingly. For example, design and development are described in this document, testing is defined in WP5 [13], and validation will be explained later.

3.1.1. Design

The design of a NetApp consists of three main steps, as they are depicted in the following figure. The programmer must complete these steps so that their existing code can be onboarded as a NetApp through the 5GASP platform and tested through 5GASP's testing framework.

Figure 6. Phases of the design of a NetApp in the context of 5GASP

Note that the Network Services' definitions and the Virtual Network Functions' definitions are different in the case of VM-based NetApps than in the case of Container-based NetApps. To that end, NetApp developers will be provided with templates for each type of NetApp. More specifically, solid documentation shall be compiled by the Consortium that will include templates as well as examples of template instantiations for both VM-based and Container-based NetApps.

It is important at this point to note that once the NS and VNF have been designed by the NetApp developers using either VM-based or Container-based packaging for their software, the 5GASP platform shall seamlessly treat the NetApp descriptors. Specifically, a NetApp, either VM-based or Container-based, shall be onboarded through the same 5GASP portal and shall make use of 5GASP's CI/CD and testing tools in a standard way.

3.1.2. Development

Following the design phase, the developer can start developing the NetApp. The development consists of materializing the design considerations studied in the previous step, resulting in, in the end, a NetApp ready to be onboarded to the 5GASP portal. Consequently, the development of the NetApp is also divided into three steps (see Figure 7), one for each component of the needed triplet: the VNFDs/NSDs, the network slice and the test descriptors.

Figure 7. Phases of the development of a NetApp in the context of 5GASP

The development phase is further discussed in Sections 3.3 and 4.2.

3.1.3. Testing [BLB]

The current version of the Testing stage aims to check whether the developed NetApp matches expected requirements and ensure that NetApp is defect-free. This process assures quality and optimized efficiency.

5GASP platform has to run tests on every form, every script, run spell-checking software to remove every possible typo. The testing is the development process where bugs, errors, and risks are checked to be removed to ensure a robust continuous delivery, strong infrastructure, and the final successful execution of the NetApps.

The 5GASP automated testing will help to have meticulous continuous testing and check NetApp performance to ensure the final market-ready product fulfils all requirements.

- Usability testing
- Compatibility testing
- Interface testing
- Services testing
- Low-level resource testing
- Performance testing
- Operational testing
- Security testing
- Network KPIs fulfilment testing

Therefore, two main types of tests are involved in the 5GASP:

- Platform-specific tests. Platform-specific tests might access the connectivity and the performance of the communication between VNFs or CNFs of a NetApp. An example tests to assess the bandwidth between the OBU and vOBU VNFs is provided in Section 4.2 of D.5.1.

- NetApp-specific tests. NetApp-specific tests shall be uploaded to the test repository by the developers of each NetApp. These tests intend to examine the behavior and/or performance of specific functionalities of the NetApp as well as to evaluate NetApp-specific KPIs. Outcomes

of this stage will be used for the Validation (described in Subsection 3.1.4). All the testing procedures have been already defined in detail in WP5 and have been already reflected in D.5.1 [11].

3.1.4. Validation [BLB]

After the testing phase Validation process takes place, the validation can be defined as the process of evaluating developed NetApp during or at the end of the development process to determine whether it satisfies specified requirements (defined KPIs) [14].

Several KPI could be extracted from 3GPP/ETSI/others covering:

- Functional Suitability (Functional Completeness, Functional Correctness, Functional Appropriateness);
- Performance efficiency (Time-behavior, Resource utilization, Capacity);
- Compatibility (Co-existence, interoperability);
- Reliability (Maturity, availability, Fault tolerance, recoverability);
- Security (Confidentiality, integrity, non-repudiation, accountability, authenticity);
- Maintainability (reusability, analyzability, Modifiability, testability);
- Portability (adaptability, instability, replaceability).

3.2. Design

For a NetApp developer, for her software to be deployed and tested in the 5G environment offered by 5GASP, several 5G-specific designs must be undertaken by the programmer so that the 5GASP environment is firstly able to 'understand' the NetApp design as we briefed in section 3 so that the 5GASP platform later can deploy and test the NetApp.

The first design consideration towards producing a NetApp based on a custom code is to define the main components of the NetApp in terms of services (described with Network Service Descriptors-NSDs) and interworking service components, that is Virtual Network Functions (VNFs) in the 5G terminology. This practice should be familiar to engineers of cloud-bound applications, as they use the notion of services for designing and building their applications. Therefore, the programmer must list the services that are part of their overall application and then describe these services with NSDs. This means that each service must be described by an NSD as there exists a 1-1 relationship between a service and an NSD.

Overall, in this first NetApp design step, the programmer must clearly define the list of NSDs and VNFs and their respective cardinality since the latter is essential for a testbed to be able to accommodate them. As an example, the programmer shall communicate to the 5GASP platform that their application consists of N NSDs and M VNFs where an NSD can consist of more than one VNF since there is a 1-many relationships between an NSD and a VNF as is the case of programmer's services: one service is composed by one or more service components/functions. As we discussed in section 3.1.1, the 5GASP platform shall treat seamlessly NSDs and VNFs irrespectively of the virtualization technology used by the NetApp developer, whether the technology used is VM-based or Container-based.

During this step of the design of the NetApp, the goal is that the programmer actually defines the NSDs for all services of the NetApp. The NSDs shall include at least the following important information that shall be taken into account by 5GASP's control plane when the NetApp will be onboarded:

- Number of VNFs that are part of the NS.
- Type of packaging, i.e., if the service shall be packed in a VM-based manner or in a Container-based manner (VNF/CNF, as we shall elaborate later in this document).
- Maximum required latency value in milliseconds and minimum slice BW required.
- The need for Internet connectivity.
- Hardware resource requirement is needed to deploy the NetApp in terms of CPU cores, RAM and hard-disk requirements.
- Ingress and egress bandwidth requirements of the NetApp's services.

This first design phase of the NetApp also includes the definition of dependencies from other NetApps. This is especially important in implementing the 5GASP use cases, which consist of more than one inter-working NetApps, that work together to meet the technical objectives of the use case.
This second design phase encompasses defining the 5G network requirements of the NetApps' slices associated with the NetApp to be instantiated on the 5GASP platform. As briefed earlier in this document, the slice(s) definition shall adhere to the NEST template, which is explained in the document.
The third design phase of the NetApp, before it can be submitted to 5GASP for onboarding, is to define the sets of tests that the 5GASP platform must undertake to ensure that the NetApp does not exhibit any errors during instantiation and/or operation over the relevant 5GASP testbed(s) where the NetApp shall be onboarded. Tests shall be defined according to Test Descriptors as elaborated in Sections 3.1.3 and 4.1.2.

3.3. Development

Once the design of the NetApp is performed, the developer can start developing the different descriptors that compose the NetApp as a whole. To do this, the methodology considers a division of the development phase into three steps: (i) the VNFDs/NSDs, (ii) the GST/NEST, and (iii) the test scripts and test VNFs descriptors. The development insights of the NetApp functionality itself, i.e., the application code, are out of the scope of this methodology definition.

The first descriptor to consider is the VNFDs/NSDs that compose the NetApp. These descriptors must follow the standards supported by the NFVOs offered by each of the test sites (OSM8, 9, 10, TOSCA). This information will be available to the experimenters to avoid descriptor structure and keywords problems.

Depending on the type of NFVO to be used, the descriptors' format will vary, but the developer has to take into account the version of the NFVO, as there may be differences between them. For example, the descriptor format for OSM Release NINE is not the same for OSM Release EIGHT (or previous releases), as they implement a new northbound API to be upgraded to the new ETSI SOL006. To prevent the developer from having to be aware of these details, 5GASP will offer the specific type and version of the NFVO available on testbeds, as well as examples for them. Furthermore, some examples of descriptors will be available to

simplify the developer's job, easing the task avoiding their creation from scratch. It could also be possible to offer some "ready-to-fulfill" descriptors, which developers could customize (for example, with the number of hardware resources or network interfaces). Also, a list of network and computing resources available on each facility will be offered to developers to easily select the resources they want to use in their NetApps.

When preparing the GST/NEST descriptor for the NetApp, it is important to consider the testbeds' capabilities. In the first stages of the project, a list of pre-defined NEST for each testbed will be offered to the developers when onboarding the NetApp. Therefore, the network slice assigned to the NetApp in the deployment will be one of the default networks slices. Later on, during the project, the experimenter will select the desired capabilities for the network slice. Then, a series of available NESTs from the different testbeds will be offered, deciding which the developer will use. Finally, in the final stages of the project, the portal will also accept the NESTs defined by the experimenter, choosing a best-effort option if any testbed cannot fulfil the requirements demanded by the NetApp.

Consequently, the developer needs to know the network resources that the NetApp will demand once deployed to achieve an optimal operation, pass all the test and validation procedures, and obtain the certification by the 5GASP platform. Therefore, in the following, we enumerate some example aspects to consider when fulfilling the NEST template:

- Area of Service and Region Specification: the preferred location to deploy the NetApp; this has to be considered if the test facility site of the chosen area has the necessary resources to host the NetApp properly, e.g., if the test site has edge capabilities.
- Throughput: necessary to estimate the bandwidth use that will require the uplink and downlink channels of the NetApp.
- Isolation level: It is essential to consider if the NetApp can share physical resources with other applications related to the slicing resource isolation.
- Mission-critical support: whether the network slice host critical services.
- Slice QoS parameters indicate the kind of traffic the 3GPP 5QI standard will host.
- Maximum PLR: The NetApp could handle the maximum Packet Loss Ratio without disrupting the service.
- Supported device mobility: it is important to consider the effects of speed on wireless communications in the automotive vertical, as the radio link technology and capabilities will have to support it.

Finally, regarding the development of the testing descriptors, the approach may change depending on the type of tests. Initially, multiple infrastructure-related tests will be pre-provided by the different test facility sites. Thus, the developer can avoid implementing these kinds of tests, as they will be available to select during the onboarding process. They will have the corresponding KPIs included in the NetApp descriptors.

Moving on to the custom test scripts, the code itself is under the developer's responsibility, and it will need to define an output to establish the success or failure of the test. Moreover, it is still to be discussed and decided in the project the details about the test repositories where these kinds of tests will be located.

Regarding the custom test VNFs, the development considerations are similar to the ones presented and discussed above in relation to the NetApp VNFDs/NSDs. It is important to mention that from the point of view of these test VNFs, the NetApp must be considered as a BlackBox with some inputs and some outputs, which will be the ones used to validate the test. In this way, these VNFs will commonly embed specific applications with a certain configuration prepared to validate a concrete aspect of the deployed NetApp. For example, a testing VNF could be a traffic generator together with a particular data packet trace in order to evaluate the behavior and response of the NetApp when that specific traffic is received and processed.

4. Onboarding approach

This section provides some more insights about the onboarding, concretely, regarding the NetApp management, the types of NetApps, and the descriptors that will define them. Besides providing a Generic development guide to implement a generic NetApp, we also provide a complete technical PoC example of the workflow from design and development to the onboarding and testing of three of the NetApps presented in the project representing the main three verticals in this project, the virtual On-Board Unit (vOBU), the PDR IOPS, and the PrivacyAnalyzer cross-vertical-PoC NetApps.

4.1. NetApp Management

The project goal is to express the NetApps as a set of virtual functions to facilitate an automated, repeatable deployment and testing cycle. To that extent, NetApps should be transformed into widely utilized models derived from major standards bodies. Therefore, the modeling and packaging of the ETSI's SOL006/OSM (YANG model) and/or SOL001/ONAP (TOSCA model) will be followed. Consequently, adaptions and enhancements could be made on those models to enable particular deployment and testing scenarios. Furthermore, given the VNFication phase of the NetApps, implementation can support both ordinary VM-based approaches and the more contemporary container-based ones. In the latest years, the global tendency aims towards the latter approach and 5GASP envisages going along and contributing to the NetApp community's development actively. Thus, developers would be expected and strongly encouraged to conform to that approach.

Furthermore, 5GASP envisions providing developers with a single entry-point using a portal to enable a straightforward procedure towards the onboarding process. This portal will allow any developer to onboard its NetApp, specify the accommodating testbed to host it and describe the tests that should be triggered once the NetApp is deployed. This "triplet" is bundled together in a single entity, creating a unified abstraction for all sites, thus assuring that the onboarding, activation and testing can be appropriately performed on any 5GASP facility. Also, aiming towards interoperability with any NFV/3GPP compliant 5G System, the project's approach for each entity of the "triplet" is to be defined under widely embraced models in the industry, i.e., GSMA's GST/NEST, TMF's ServiceSpecification and ServiceOrder, as further presented in the subsequent sections.

4.1.1. NetApp's categories

Although NFV is designed to be completely agnostic about how VNFs are instantiated, in practice, the most common method is deploying them as virtual machines (VMs). Nevertheless, the instantiation of VNFs as Linux (Docker) containers has become more prevalent in recent years, since its deployment, scaling, etc., is considerably faster and requires lower resources. However, for certain purposes, it may be necessary to use one or the other depending on the characteristics and needs of the NetApp (or even depending on the facility where it is going to be deployed), so the idea is to offer both possibilities transparently.

4.1.1.1. NetApps as regular VNFs

Deploying VNFs as VMs is the most widely used and common way to instantiate them. A VM-based VNF is a virtual machine with hardware resources, network interfaces, and essentially, an image (usually qemu-based) that includes/implements the desired functionality. Therefore, one of the most important points when deploying a VM-based VNF is to have the image that conforms to the VM since it is the one that will contain the functionality that we want to offer in that VNF.

To do this, usually, a previously configured image is available with basic functionality (for example, the necessary packages installed), ready to receive the specific configuration requested (IPs, targets, configuration files, etc.) after instantiation. Thus, we have a generic image to adapt to the characteristics of the scenario, but specific enough for not having to do all the required configuration after instantiation (which reduces the time needed until the VNF is ready).

To automate the image generation, 5GTANGO [15] presents a method to create VM-based images, which consists of selecting a pre-existing container (or uploading one) that implements the required functionality. Once identified, it generates a VM with a vanilla OS (e.g., a freshly installed Ubuntu distribution) and installs the container on it. Then, a new VM image is generated based on that VM, so an image with the required functionality is ready to be instantiated.

Another typical option is to use a base image and configure it when deployed using different methods (for example, Ansible). In this case, a YAML file specifies the packages and functions to be installed, defined with a template and scripting system for the configuration files to be deployed on the VNF (a VM in this case). Another possibility to inject configuration is using Day-0 and Day-1 configuration (from OSM) that uses Juju Charms to configure instances. As above-mentioned, Day-0 is the configuration that applies while instantiation (cloud-init files, RSA public key, etc.), and Day-1 the configuration to be injected and applied once the VM is deployed (install Apache, clone a repository, etc.).

4.1.1.2. NetApps as CNFs

Containerized Network Functions (CNFs) is another approach in building cloud-native 5G NetApps with expected low-latency and ultra-reliability guarantees, amongst other guarantees as deriving from the business specifications for NetApps. The goal in using the CNF approach is to move from the 'heavy' VM-based approach discussed in the previous section to the cloud-native CNF approach, where CNFs are composed of microservices, often as open-source and which are hosted in Containers, such as Docker containers.

The motivation behind the CNF approach is that monolithic Network Functions implemented in VMs take a lot of time to deploy and to upgrade on one hand, and on the other hand, their performance might be as much as the performance of the lightweight CNF approach in an operational 5G setting given the ability of CNFs to scale easily and fast, leveraging Container as a Service tools such as Kubernetes (K8s) which have many capabilities in detecting failures of microservices and performance degradations and automatically deploys a new microservice in a container. This allows the capability to have a new microservice that replaces the old one with the required capacity.

Overall, the small footprint of microservices (note that a container may include one or more microservices) and their fast start times provides a high performance and scalable approach to deal with the 5G requirements in terms of latency, reliability, etc. in many use-cases. However, engineering of microservices/CNFs for 5G use cases is not a tedious task. For this reason, designed principles such as Istio [https://www.istio.io] have emerged to help developers solve the problems that arise while building CNF-based NetApps. The cloud-native industry is quickly evolving, and developers must choose the proper tools and update their tools to derive CNF-based NetApps that meet their requirements in an actual 5G deployment. A good overview of state-of-the-art tools for building cloud-native NetApps can be found at [16].

Regarding standardisation of the CNF approach for building 5G networks and services, the 3GPP Technical Specification Group on Service and System aspects have proposed architecture in 3GPP TS 23.501 [17], which is based on services. Specifically, the architecture defines CNFs and reference points used to interconnect CNFs. As shown in Figure 8. Proposed 3GPP architecture for cloud-ready Network Functions, with this service-based approach, the 5G network employs a service-based architecture that uses services that communicate with each other through a message bus where services are registered and subscribed to each other.

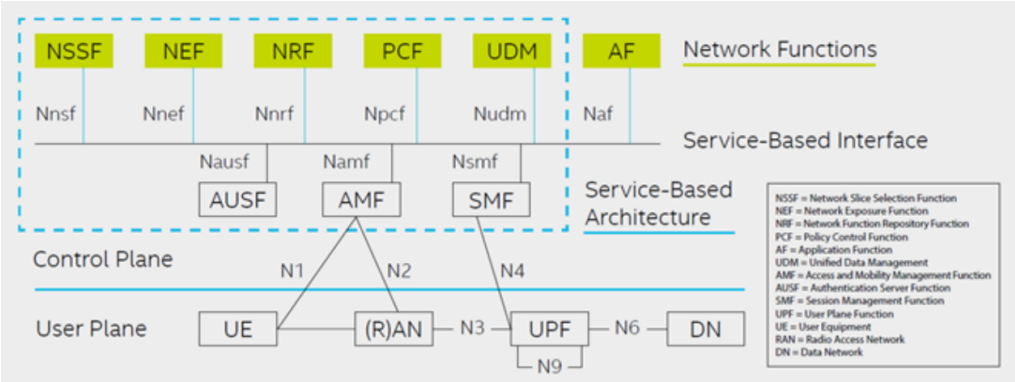

Figure 8. Proposed 3GPP architecture for cloud-ready Network Functions

The goal of this stream of work of 3GPP is that 5G networks can ultimately be cloud-ready and leverage the advantages of cloud-native approaches that are state-of-the-art in cloud service solutions. The business goal is to lessen the risk that 5G service providers lose customers if the latter chose to work with other players such as established cloud service providers. Therefore, 5GASP embraces the CNF approach and will demonstrate how selected 5GASP NetApps – such as the PrivacyAnalyzer NetApp by Lamda Networks - can be written as CNFs and be orchestrated via Kubernetes in selected 5GASP testbeds, such as the testbed offered by partners ITAV, UoP and UnivBris.

4.1.2. NetApps descriptor

The idea for 5GASP is to create a "meta-package" that includes the Network Slice requirements (NEST information), the NSDs that conforms to the NetApp (with their respective VNFDs), and the tests the user wants to perform over them to validate the NetApp and obtain the certification. These packages can be included entirely in the meta-package or references to already uploaded packages. However, in the first stages of the project, these descriptors will be added individually and uploaded one by one. At the same time, the platform acquires the maturity to implement meta-package management.

4.1.2.1. NSD/VNFD Packages

Besides the descriptor (the YAML/TOSCA file, which includes the description of the NS/VNF respectively), the packages can include other relevant information, for example, the image to be used, cloud-init files, some after-deployment configuration, such as charms in OSM packages or scripts following the TOSCA schema, etc. These files must be included in the package, following the established format for the orchestrator they are built for (as it is not the same package or descriptor for OSM as for ONAP).

As previously said, the idea is to work with a meta-package that is compatible with both types of descriptors (OSM and TOSCA, as 5GTANGO does). The meta-package would indicate which kind of descriptor each one is (for example, OSM and which release). Afterwards, based on this meta-package, the descriptors for the specific orchestrator would be generated. Nevertheless, both packages will be formed independently and separately in the first stages of the project, depending on the testbed — or orchestrator — to be used.

Depending on the type of orchestrator required (OSM or ONAP), the format of the descriptor will vary, likewise in accordance with the used version of the orchestrator (it is not the same format between OSM Rel EIGHT than OSM Rel NINE, as the first uses ETSI SOL005 and the second ETSI SOL006). In this way, proper verifications must be performed to ensure the correctness of the package. Package examples are available in the orchestrator's repositories, as well as they will be in the 5GASP repository to be used as a template. Thus, developers won't need to create their packages from scratch, as they will have a guideline to perform it.

4.1.2.2. GST/NEST

Network slicing is one of the key enablers of the development of 5G technologies. 3GPP defined it as a dedicated logical network providing specific network capabilities and characteristics. These slices are usually bonded to a Service Level Agreement (SLA) agreed between a Network Slice Customer (NSC) and a Network Slice Provider (NSP). A network slice can be instantiated across multiple parts of the network, and it is composed of a set of dedicated or shared resources. The slice can be isolated from other network slices if dedicated resources are used.

The Generic Network Slice Template (GST) is a set of attributes that can characterize a type of network slice or service; it is generic and is not tied to any specific network deployment [18]. This template was introduced by the GSM Alliance (GSMA) with the aim of introducing guidelines for verticals on how to address their service requirements and to facilitate the establishment of SLAs between operators and business customers.

The NEtwork Slice Type (NEST) is a GST filled with values. This set of attributes with certain values complies with a given collection of requirements derived from a use case defined by a network slice customer. The NEST is used as an input for preparing a network slice instantiation performed by the network slice provider. Furthermore, multiple network slice instances can be created from the same NEST, and existing instances can also be reused.

In Figure 9, the network slice lifecycle is illustrated. The NSC provides the requirements of its particular use case to the NSP. Then, the latter generates a NEST by mapping these service requirements into the attributes of the GST.

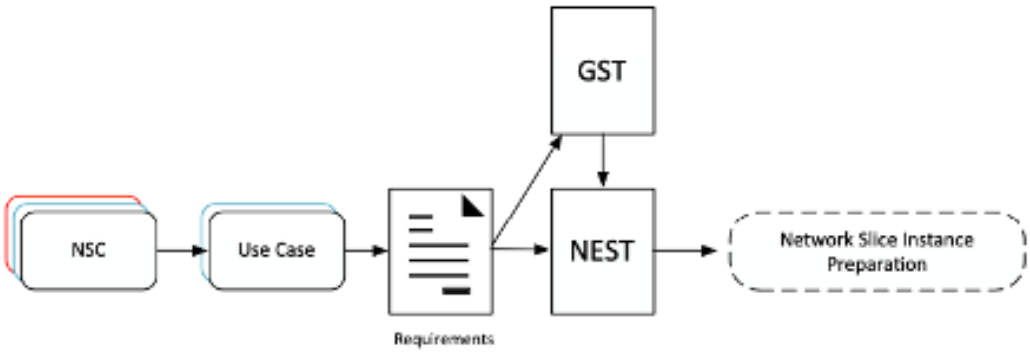

Figure 9. GST Network slice lifecycle

GST/NEST is the selected template to be used as the network slice descriptor in the NetApp's definition. Each designated testing facility of the 5GASP project will provide information about the network requirements they support by means of NESTs. Therefore, the template will be perfectly aligned with this GSMA standard. By doing so, NetApp's developers can follow two different approaches. The first one is defining and fulfilling the NEST associated with its NetApp and including the network requirements that they think the NetApp will demand. The second one is to select a pre-defined NEST that is associated with a certain 5GASP facility. While the latter is the best starting point and do not raise any issues, the former has to be handled with care, as the facility test sites need to support and implement the network requirements defined by the NESTs. Thus, in the case that none of the sites can fulfill the required network resources as a whole, a best-effort policy will be followed, and the testbed with the most similar network characteristics will be selected. By doing so, the design and development phase of the NetApp is simplified from the point of view of the experimenter.

In this way, in an earlier stage of the project, whenever a developer wants to onboard a NetApp, it may select the site that will host the NetApp deployment from a pre-defined list that will show the network specifications and characteristics of each of the facilities. Thus, the network slice descriptor of the NetApp will be a NEST with the selected site capabilities. Further on, the 5GASP system will automatically select the most appropriate site (or sites) to deploy the onboarded NetApp based on the network requirements introduced by the developer. If there is more than one option, the developer will be able to select the one with a better match.

An example of a NEST from a NetApp of the Project is shown in Table 4.1. It corresponds to the NetApp 4: Multi-domain Migration NetApp.

Multi-domain NETAPP's NEST	
Area of service	SP, PT, UK, RO (AUTO-V use case)
Area of service: Region Specification	Murcia, Aveiro, Bristol, Bucharest
Downlink maximum throughput per UE	100 Mbps
Uplink maximum throughput per UE	100 Mbps
Isolation level	3: Tenant/service isolation
Slice quality of service parameters: 3GPP QoS Identifier (5QI)	9
Maximum Packet Loss Rate	1%
Supported device velocity	3: Vehicular: 10 km/h to 120 km/h

Table 1 NetApp4: Multi-domain NetApps NEST

4.1.2.3. Test Descriptor

To simplify the Test phase, 5GASP aims to design its own test descriptor, using as a template Test Descriptor files inspired by previous related descriptors presented in projects such as 5GEVE [19] or 5GTANGO [20]. The concrete format of the descriptor will be specified in

further phases of the project, though in this deliverable, the basis of what the descriptor must contain and its approached structure will be exposed.

In 5GASP, there will be two different types of tests: the tests included as scripts (basically code that will be executed) and Test VNFs. A test VNF is a VNF which purpose is to perform some test and that it's not involved in the functionality of the NetApp itself. In this sense, the difference between them is that the Test VNF is a separate VNF that is requested to execute the tests (as they are included inside the VNF), whereas the scripts can be executed inside the NetApp VNFs (the ones we want to test) or in a different place. It is also remarkable the possibility of using simple tests (available in the platform), such as iperf, ping, or similarly to check the basic connectivity and operation.

Moreover, the platform will include a test repository, with pre-defined tests and simple tests to be used, and the possibility of creating custom tests using the Test Descriptor. In this way, the main characteristic of the proposed Test Descriptor will be its ability to be divided into different steps or phases (5GTANGO alike), for example, "Setup", "Execution", and "Validation". The other steps are described below:

- Setup: This phase includes the definition of the scripts and the VNFs that will be used to perform the test and also the deployment of the test VNFs. Regarding scripts, they could be included in the package as executable files, or the commands could be directly included inside the test descriptor, similarly as 5GEVE.

- Execution: This phase includes launching the scripts (both VNF and bare scripts, in the preferred order or simultaneously), and later the collection of metrics obtained from the executions.

- Validation: Finally, this phase includes the analysis of the obtained data, the computing of the KPIs using the obtained metrics as input, and lastly, the validation of both KPIs and tests.

However, these phases are still under study and consideration, and they may undergo some changes during the first stages of the project, as we may find adjustments that better fit the project's needs.

4.2. NetApp PoC Design, Development and Onboarding examples

To illustrate the designed methodology presented in this deliverable, in this subsection, we provide an introduction to Generic NetApp development followed by a detailed example of the application of this methodology to the design, development and onboarding of three of the NetApps that correspond to the three verticals use-cases are proposed in the 5GASP project.

4.2.1. Generic NetApp

The guidelines provided by the generic Proof-Of-Concept (PoC) NetApp are the base introduction to developers to get into the 5GASP platform and bootstrap the development of

their own NetApp. At this level, the basic implementation PoC might not have an objective to provide a particular functionality. Still, the generic approach can be of great value by allowing the newcomer to use the 5GASP platform as a playground and provide a step-by-step guide for design, development, and onboarding and further perform the tests of their choice on a PoC NetApp corresponding to vertical of interest.

4.2.1.1. Generic NetApp design

The step-by-step approach to building a PoC NetApp is tuned to consider no more than the basic tools and technologies that are understandable by any network engineer or developer and meant not to be based on any advanced knowledge.

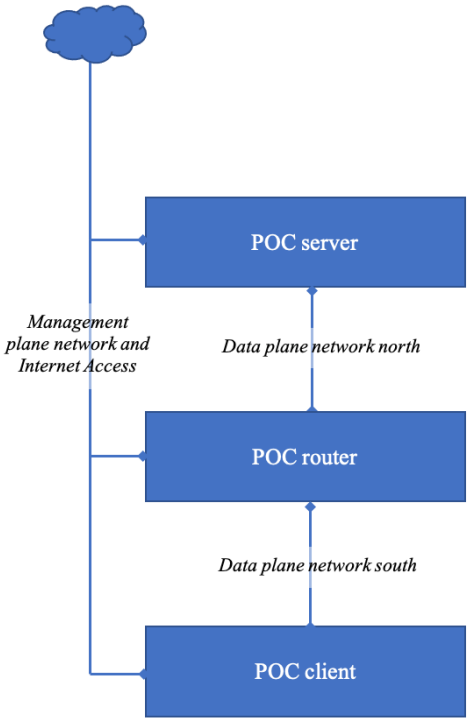

Figure 10. PoC Generic NetApp

The generic NetApp PoC design must assure the following basic functionalities:

- Being able to establish a basic communication between a client and a server (e.g. ICMP ping, DNS, HTTP, or user-defined packets).
- Being capable of handling basic routing inside a NetApp infrastructure.
- The NetApp VM must be remotely accessible after deployment and is capable of basic manual tests and captures.
- The remote configurations and packages installation must be guaranteed.

To this level, no applications are defined; the generic PoC NetApp will guide the developer for the first step to bootstrapping the development of a future use-case NetApp. Thus, no significant resources might be required. A single VNF descriptor template is used to define a

VNF constituted of the network interfaces alongside the NSD descriptor template for different OSM versions.

For the generic PoC NetApp, the following basic resources are used:

- Packaging: VM
- Internet connectivity: yes (for SSH access and installation of additional packages)
- Hardware resources required: 1 CPU, 1 GB RAM, 10 GB HDD
- Placement latency: 30 seconds

For generic purposes, and in order to be compatible with most use-cases, the KPIs and network slice placements are defined using the default values (see table 2) and meant to be use-case independent, the following PoC NetApps examples will provide more precise use-case dependent KPIs. However, as the generic PoC NetApp is not intended to offer features targeting the production environment, it is not expected to be certified.

Generic KPI name	Metric Indicator (How)	KPI value	KPI unit
Initial time	Time needed to deploy for first time the entity (Generic NetApp)	<5	Minutes
Service response time	Delay ratio while including Generic NetApp instantiation	<30	s
Service downtime	Ratio of time the Generic NetApp is not up and running	<10	%

Table 2 Generic NetApp KPIs

As for the inoperability, especially for the cross-verticals use-case, the generic NetApp is intended to be used independently from any other NetApp; however, for the purpose of troubleshooting unexpected test results, a NetApp developer may want to deploy this NetApp along with their own NetApp in order to use specific network tools already instead of preparing a custom image for this need.

4.2.1.2. Generic NetApp development

To serve the objectives depicted above, the NetApp comprises 3 VMs: a client, a server, and a router. These 3 VMs are supported by a single VM image, able to endorse each role.

The creation of the generic VM image is performed to be completely reproducible. The VM image is recommended to be built using the following procedure:

1. The installation of the qemu-kvm.
2. Download a CD/DVD image of the distribution (Debian 11 in the Generic PoC example)
3. Run qemu with the CD/DVD attached and install the distribution.
4. Install and configure the VM dependencies for running and auto-configuring under OpenStack (cloud-init)
5. Install and configure on the VM packages for applications/testbed (nftables, netcat-openbsd, socat, mtr, radvd, isc-dhcp-server, unbound, tshark, nmap, nginx, curl in the example)

NSD and VNF files are provided both for OSM versions 8 and 9, in order to support a broader number of infrastructures. Network names may need to be adapted to match the ones defined on target infrastructures.

The NSD file defines two data networks that allow a basic routed setup: data-plane-network-north and data-plane-network-south. These networks connect respectively the client-vnf with the router-vnf, and the router-vnf with the server-vnf. A management network is also defined as allowing to connect to the VMs and to get internet connectivity: management-plane-network.

The following default NSD template defines a generic NetApp for OSM 8:

```
nsd:nsd-catalog:
  nsd:
  - constituent-vnfd:
    - member-vnf-index: client-vnf
      vnfd-id-ref: yogoko_vnfd
    - member-vnf-index: server-vnf
      vnfd-id-ref: yogoko_vnfd
    - member-vnf-index: router-vnf
      vnfd-id-ref: yogoko_vnfd
  description: NS with 3 VNFs connected by mgmt-plane-network and 2 data-
plane-network VLs
  id: yogoko_nsd
  name: yogoko_nsd
  short-name: yogoko_nsd
  version: '1.0'
  vld:
  - id: management-plane-network
    mgmt-network: true
    vnfd-connection-point-ref:
    - member-vnf-index-ref: client-vnf
      vnfd-connection-point-ref: eth0
      vnfd-id-ref: yogoko_vnfd
    - member-vnf-index-ref: server-vnf
      vnfd-connection-point-ref: eth0
      vnfd-id-ref: yogoko_vnfd
    - member-vnf-index-ref: router-vnf
      vnfd-connection-point-ref: eth0
      vnfd-id-ref: yogoko_vnfd
  - id: data-plane-network-north
    vnfd-connection-point-ref:
    - member-vnf-index-ref: server-vnf
      vnfd-connection-point-ref: eth1
      vnfd-id-ref: yogoko_vnfd
    - member-vnf-index-ref: router-vnf
      vnfd-connection-point-ref: eth1
      vnfd-id-ref: yogoko_vnfd
  - id: data-plane-network-south
    vnfd-connection-point-ref:
    - member-vnf-index-ref: client-vnf
      vnfd-connection-point-ref: eth2
      vnfd-id-ref: yogoko_vnfd
    - member-vnf-index-ref: router-vnf
      vnfd-connection-point-ref: eth2
      vnfd-id-ref: yogoko_vnfd
```

Table 1 – NSD considering OSM 8

The following file defines NSD for OSM 9:

```
nsd:
  nsd:
  - description: NS with 3 VNFs connected by mgmt-plane-network and 2
data-plane-network VLs
    df:
    - id: default-df
    virtual-link-profile:
    - id: vlp-management-plane-network
      virtual-link-desc-id: management-plane-network
    - id: vlp-data-plane-network-north
      virtual-link-desc-id: data-plane-network-north
    - id: vlp-data-plane-network-south
      virtual-link-desc-id: data-plane-network-south

      vnf-profile:
      # VNF for Client
      - id: client-vnf
        vnfd-id: yogoko_vnfd
        virtual-link-connectivity:
        - constituent-cpd-id:
          - constituent-base-element-id: client-vnf
            constituent-cpd-id: vnf-mgmt-ext
          virtual-link-profile-id: management-plane-network
        - constituent-cpd-id:
          - constituent-base-element-id: client_vnf
            constituent-cpd-id: vnf-data-south-ext
          virtual-link-profile-id: data-plane-network-south

      # VNF for Server
      - id: server-vnf
        vnfd-id: yogoko_vnfd
        virtual-link-connectivity:
        - constituent-cpd-id:
          - constituent-base-element-id: server-vnf
            constituent-cpd-id: vnf-mgmt-ext
          virtual-link-profile-id: management-plane-network
        - constituent-cpd-id:
          - constituent-base-element-id: server-vnf
            constituent-cpd-id: vnf-data-north-ext
          virtual-link-profile-id: data-plane-network-north

      # VNF for router
      - id: vnf-router
        vnfd-id: yogoko_vnfd
        virtual-link-connectivity:
        - constituent-cpd-id:
          - constituent-base-element-id: router-vnf
            constituent-cpd-id: vnf-mgmt-ext
          virtual-link-profile-id: management-plane-network
        - constituent-cpd-id:
          - constituent-base-element-id: router-vnf
            constituent-cpd-id: vnf-data-south-ext
          virtual-link-profile-id: data-plane-network-south
        - constituent-cpd-id:
          - constituent-base-element-id: router-vnf
            constituent-cpd-id: vnf-data-north-ext
          virtual-link-profile-id: data-plane-network-north
```

```
    id: yogoko_nsd
    name: yogoko_nsd
    version: '1.0'
    virtual-link-desc:
    # Management Network
    - id: management-plane-network
      mgmt-network: true
    # Data Network (Server Side)
    - id: data-plane-network-north
    # Data Network (Client Side)
    - id: data-plane-network-south
    vnfd-id:
    - yogoko_vnfd
```

Table 3 The Generic NetApp NSD file considering OSM 9

A single VNF definition is used for both three VNFs. The VNF descriptor defines a VNF constituted of three network interfaces, one to be connected to the management network, and the two others to be connected to data networks, when applicable. In particular, the router-vnf will have its two data network interfaces connected, while the client-vnf and the server-vnf will only have one data network interface connected (see figure 9). Considering OSM 8, the following represents the generic VNF definition:

```
vnfd-catalog:
  vnfd:
  - connection-point:
    - name: eth0
      type: VPORT
    - name: eth1
      type: VPORT
    - name: eth2
      type: VPORT
  description: A VNF constisting of 1 VDU with 3 interfaces
  id: yogoko_vnfd
  mgmt-interface:
    cp: eth0
  name: yogoko_vnfd
  short-name: yogoko_vnfd
  vdu:
  - count: 1
    description: yogoko_vnfd-VM
    id: yogoko_vnfd-VM
    image: yogoko_base_image
    interface:
    - external-connection-point-ref: eth0
      name: eth0
      type: EXTERNAL
      virtual-interface:
        bandwith: '0'
        type: PARAVIRT
        vpci: 0000:00:0a.0
    - external-connection-point-ref: eth1
      name: eth1
      type: EXTERNAL
      virtual-interface:
        bandwith: '0'
```

```
        type: PARAVIRT
        vpci: 0000:00:0a.0
    - external-connection-point-ref: eth2
      name: eth2
      type: EXTERNAL
      virtual-interface:
        bandwith: '0'
        type: PARAVIRT
        vpci: 0000:00:0a.0
    name: yogoko_base_image
    vm-flavor:
      memory-mb: '1024'
      storage-gb: '10'
      vcpu-count: '1'
  vendor: OSM
  version: '1.0'
```

Table 4 The Generic NetApp VNF considering OSM 8

Where for OSM 9, a slight difference in the VNF definition is used as follow:

```
vnfd:
  description: A VNF consisting of 1 VDU with 3 paravirt interfaces
  df:
  - id: default-df
    instantiation-level:
    - id: default-instantiation-level
      vdu-level:
      - number-of-instances: 1
        vdu-id: dataVM
    vdu-profile:
    - id: dataVM
      min-number-of-instances: 1
  ext-cpd:
  - id: vnf-mgmt-ext
    int-cpd:
      cpd: mgmt-eth0-int
      vdu-id: dataVM
  - id: vnf-data-south-ext
    int-cpd:
      cpd: data-eth1-int
      vdu-id: dataVM
  - id: vnf-data-north-ext
    int-cpd:
      cpd: data-eth2-int
      vdu-id: dataVM

  id: yogoko_vnfd
  mgmt-cp: vnf-mgmt-ext
  product-name: yogoko_vnfd
  sw-image-desc:
  - id: yogoko_base_img
    image: yogoko_base_img
    name: yogoko_base_img
  vdu:
  - id: dataVM
    int-cpd: mgmt-eth0-int
```

```
    - id: mgmt-eth0-int
      virtual-network-interface-requirement:
      - name: eth0
        position: 1
        virtual-interface:
          type: PARAVIRT

    - id: data-eth1-int
      virtual-network-interface-requirement:
      - name: eth1
        position: 2
        virtual-interface:
          type: PARAVIRT

    - id: data-eth2-int
      virtual-network-interface-requirement:
      - name: eth2
        position: 3
        virtual-interface:
          type: PARAVIRT

    name: dataVM
    sw-image-desc: yogoko_base_img
    virtual-compute-desc: dataVM-compute
    virtual-storage-desc:
    - dataVM-storage
  version: '1.0'
  virtalcompute-desc:
  - id: dataVM-compute
    virtual-cpu:
      num-virtual-cpu: 1
    virtual-storage-desc:
    - id: dataVM-storage
      size-of-storage: 10
    virtual-memory:
      mempage-size: LARGE
      numa-enabled: true
      numa-node-policy:
        mem-policy: STRICT
        node:
          memory-mb: 1024
```

Table 5 The Generic NetApp VNF considering OSM 9

Ongoing documentation is conducted during the lifetime of WP-4 to report the step-by-step process of the onboarding of the Generic PoC NetApp to Murcia testbed, where a learned-lessons report is to be provided as guidance to the newly joining SMEs and developers.

4.2.2. vOBU NetApp

The first step to designing the NetApp is to analyze its functioning and its architecture. This is because the programmer must clearly differentiate the multiple network functions and services that conform to the application. In this way, in Figure 11, the architecture of the vOBU NetApp can be seen.

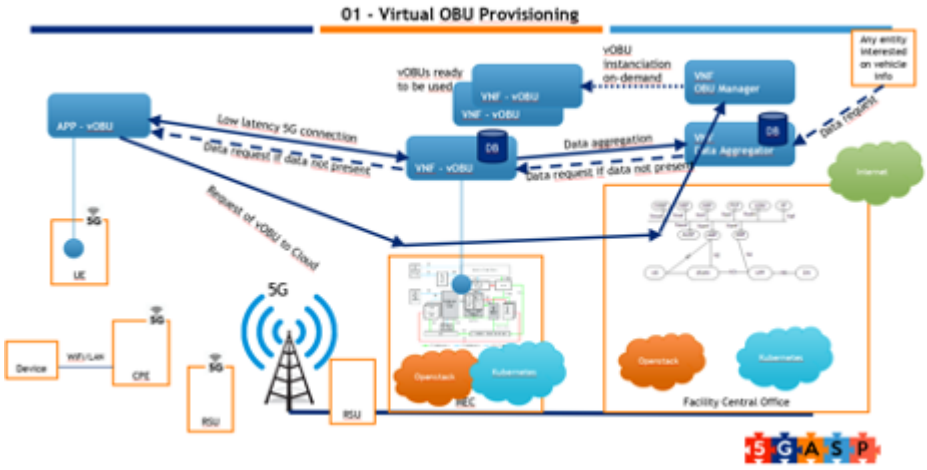

Figure 11. Virtual On-Board Unit (vOBU) NetApp architecture

4.2.2.1. vOBU NetApp design

Once the number of NSs and VNFs is decided, it has to be discussed the information the NSDs should include, taking into account the considerations detailed in Section 3.2. Some of the following items are examples of the kind of requirements that have to be defined in the design phase:

- Packaging: the VNFs will be deployed as VMs, ready to be used in OSM.
- Internet connectivity: yes.
- Hardware resources required: 2 CPU, 1 GB RAM, 10 GB HDD.
- Placement latency: 500ms.

The last step regarding the design of the NS and VNFs is to consider the dependencies with other NetApps. In this case, this NetApp does not have any dependency on other developments, however, in the vertical use case of 5GASP project, the vOBU NetApp will cooperate with other NetApps and that should be considered when designing the use case.

In the design of the network requirements of the NetApps, the developer has to define how many network slices are needed and their characteristics of them. In this case, the vOBU will simply need one network slice. Next, it has to be decided the requirements that will include the NEST template that will define the network slice on which the NetApp will be hosted. Among multiple parameters, below we present some of the requirements:

- Area of service and region specification: Murcia (SP).
- Isolation level: virtual resources.
- Mission-critical support: inter-user prioritization.
- Slice quality 5GPP 5QI: 69.
- Maximum PLR: 1%.
- Supported UE velocity: vehicular.

Finally, the last step of the design phase is the definition of the set of tests that must be performed in the 5GASP platform against the vOBU NetApp in order to ensure a valid functioning and, if successful, resulting in the certification of the NetApp. As the testing

procedure and its insights are currently under design and development, here we will focus on simple tests, mainly focused in the infrastructure, that will be used to validate the KPIs of the vOBU NetApp. To this end, a series of infrastructure tests will be defined to evaluate the metrics from which the KPIs are computed. These KPIs are the initial deployment time, the transaction speed of the messages, the PLR of the messages exchanged between the OBU and the vOBU, the service response time at the instantiation and the vOBU service downtime. These KPIs are shown in Table 2.

Generic KPI name	Metric Indicator (How)	KPI value	KPI unit
Initial time	Time needed to deploy for first time the entity (vOBU)	<5	Minutes
Transaction speed	Each message sent from an OBU needs to be redirected to the vOBU	500	Milliseconds
Packet Loss Ratio	Ratio of packets loss between the OBU and vOBU. Packets loss above packets sent	1	%
Service response time	Delay ratio while including vOBU instantiation	<30	%
Service downtime	Ratio of time the vOBU is not up and running	<10	%

Table 6 vOBU NetApp KPIs

4.2.2.2. vOBU NetApp development

Once the design phase has finished, the development of the NetApp can start. As the majority of the hard work has been already done in the previous steps, now it is time to reflect it in the multiple descriptors.

The first one is the NSD that defines the vOBU Network Service, it can be seen in Table 3. As previously mentioned, it includes the three VNFs that compose the network service and the network to which they are attached. This network is already present in the descriptor as it is known that the Murcia facility offer it. However, as previously discussed, in case the developer is unaware of the available networks of the test sites, a list of the networks ready to be used in each facility will be presented to the experimenters beforehand.

```
nsd:nsd-catalog:
 nsd:
 - constituent-vnfd:
   - member-vnf-index: '1'
     vnfd-id-ref: surrogates_mgmt_vnfd
   - member-vnf-index: '2'
     vnfd-id-ref: surrogates_vobu_vnfd
   - member-vnf-index: '3'
     vnfd-id-ref: surrogates_agg_vnfd
   description: Surrogates NS prepared for OSM version 8.
```

```
id: surrogates_nsd
name: surrogates_nsd
short-name: surrogates_nsd
version: '1.0'
vld:
- id: red800
  mgmt-network: true
  name: Red800BigNAT
  short-name: Red800BigNAT
  type: ELAN
  vim-network-name: Red800BigNAT
  vnfd-connection-point-ref:
  - member-vnf-index-ref: '1'
    vnfd-connection-point-ref: eth0
    vnfd-id-ref: surrogates_mgmt_vnfd
  - member-vnf-index-ref: '2'
    vnfd-connection-point-ref: eth0
    vnfd-id-ref: surrogates_vobu_vnfd
  - member-vnf-index-ref: '3'
    vnfd-connection-point-ref: eth0
    vnfd-id-ref: surrogates_agg_vnfd
```

Table 7 NetApp4: vOBU NetApp NSD

Next, we present in Table 4 one of the VNFs that compose the NetApp, the one corresponding to the vOBU entity itself. Here it is important to highlight some of the fields in relation to the design phase, such as the interface that will be connected to the network defined in the NSD, the image used to host the VNF and the resources required for that image. Here, the situation is similar to the network that is defined in the NSD, but with regard to the image and the flavors, in this VNF, the image and vm-flavor defined are known in advance, as they are present in the Murcia test site, although a list of offered images and flavors in each facility will be provided beforehand.

```
vnfd-catalog:
  vnfd:
  - connection-point:
    - name: eth0
      type: VPORT
    description: Surrogates vOBU entity prepared for OSM version 8.
    id: surrogates_vobu_vnfd
    mgmt-interface:
      cp: eth0
    name: surrogates_vobu_vnfd
    short-name: surrogates_vobu_vnfd
    vdu:
    - count: 1
      description: surrogates_vobu_vnfd-VM
      id: surrogates_vobu_vnfd-VM
      image: debian-baseSurrogates-certs
```

```
  interface:
  - external-connection-point-ref: eth0
    name: eth0
    type: EXTERNAL
    virtual-interface:
      bandwidth: '0'
      type: PARAVIRT
      vpci: 0000:00:0a.0
  name: surrogates_vobu_vnfd-VM
  vm-flavor:
    memory-mb: '1024'
    storage-gb: '10'
    vcpu-count: '2'
vendor: SURROGATES
version: '1.0'
```

Table 8 vOBU NetApp vOBU VNFD

Another thing to note in these descriptors is that they are designed and developed following the guidelines of OSM to be compatible with OSM Rel EIGHT, as the Murcia test site in which the NetApp is intended to be hosted counts with an instance of OSM8. In the same line, the supported OSM releases in each testbed will be exposed to the experimenters, as well as any other available NFVO.

Now that the NSD and the VNFs are defined, we can move on to the NEST. This is a straightforward step, as the heavy work has been performed in the design phase, analyzing the network requirements of the NetApp. Therefore, the only step here is to map them to the NEST template. In Table 5, the NEST template filled with the values discussed in the previous section is presented. The values are defined according to the guidelines of the GSM's GST. As in the previous case, a network slice table to fulfill these requirements (or almost identical) will be available in Murcia. In another case, the NEST would be selected from the list of predefined ones offered by the 5GASP portal.

vOBU NETAPP's NEST	
Area of service	SP
Area of service: Region Specification	Murcia
Slice quality of service parameters: 3GPP QoS Identifier (5QI)	9
Maximum Packet Loss Rate	1%
Supported device velocity	3: Vehicular: 10 km/h to 120 km/h

Table 9 vOBU NETAPP's NEST

Finally, the last stage of the development is the implementation of the tests. As presented before, 5GASP considers three types of tests: pre-provided infrastructure tests, custom test scripts and custom test VNFs. Due to the early stages of the project, at the moment there are

only some infrastructure tests developed specifically for certain NetApps, thus they cannot be considered as general pre-provided infrastructure tests.

As mentioned, for now there are only available some infrastructure tests prepared for the vOBU NetApp. These tests have been made ready to be used with Jenkins and Robot Framework. The latter is in charge of the test itself, which has been developed using Python, and the former's responsibility is to trigger the execution of the test. In this way, three different tests have been developed with the aim of evaluating three of the KPIs defined for the vOBU NetApp: the deploy time, the transaction speed and the PLR. In the following, we present as an example one of them, specifically, the one that tests the transaction speed between the OBU and the vOBU.

In first place, Table 6 shows the Robot test definition, which defines the test script in charge of obtaining the metric value and the condition to validate the test.

```
*** Settings ***
Library       TransactionSpeed.py

*** Test Cases ***
Testing the transaction speed between OBU and vOBU

    ${milliseconds}=    Transaction Speed
    Should Be Equal    ${milliseconds}    Less than 500 milliseconds
```

Table 10 Robot test definition

Secondly, Table 11 presents the Python script in charge of obtaining the average transaction time between the OBU and the vOBU. It is a simple idea that uses the ping command to obtain the values.

```
import paramiko, re

host = "1.1.1.1"
username = "jenkins-testing"
password = "password"

def transmission_speed():
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    try:
        client.connect(hostname=host, username=username, password=password)
    except:
        print("[!] Cannot connect to the SSH Server")
        exit()

    stdin, stdout, stderr = client.exec_command("ping -c 5 1.1.1.2")

    pingResult = stdout.read().decode()
    regex = re.compile(r"(.*?) = (.*?)\/(.*?)\/(.*?)\/(.*?) ms")
    result = regex.search(pingResult)
    print(pingResult)

    if result:
        avgTimeTransmission = float(result.group(3))
        print("AVG:", avgTimeTransmission)
        if avgTimeTransmission < 500:
            return "Less than 500 milliseconds"
```

```
        else:
            return "More than 500 milliseconds"
    else:
        return "Not found"

if __name__ == "__main__":
    transmission_speed()
```

Table 11 Python test script

In the future, these tests will be generalized to be offered as pre-defined tests in the facility test sites. Furthermore, more complex tests will be developed in the form of custom test scripts and custom test VNFs. For example, a custom test VNF is currently under design for the vOBU NetApp (see Figure 12). This VNF will act as a client of the NetApp, consuming the service it offers. In this way, the idea is that the test VNF will be deployed together with the NetApp and it will play the role of a NetApp client. By doing this, we can embed in the VNF any type of functional test we may think of. For example, the VNF can perform request batches to simulate multiple clients accessing the service at the same time, or it can be combined with other VNFs to create a realistic network environment. The advantage of this kind of tests is that the vOBU NetApp is considered as a "black box" and it is transparent to the test VNF. In a similar way, for the NetApp, the test VNF is simply another client.

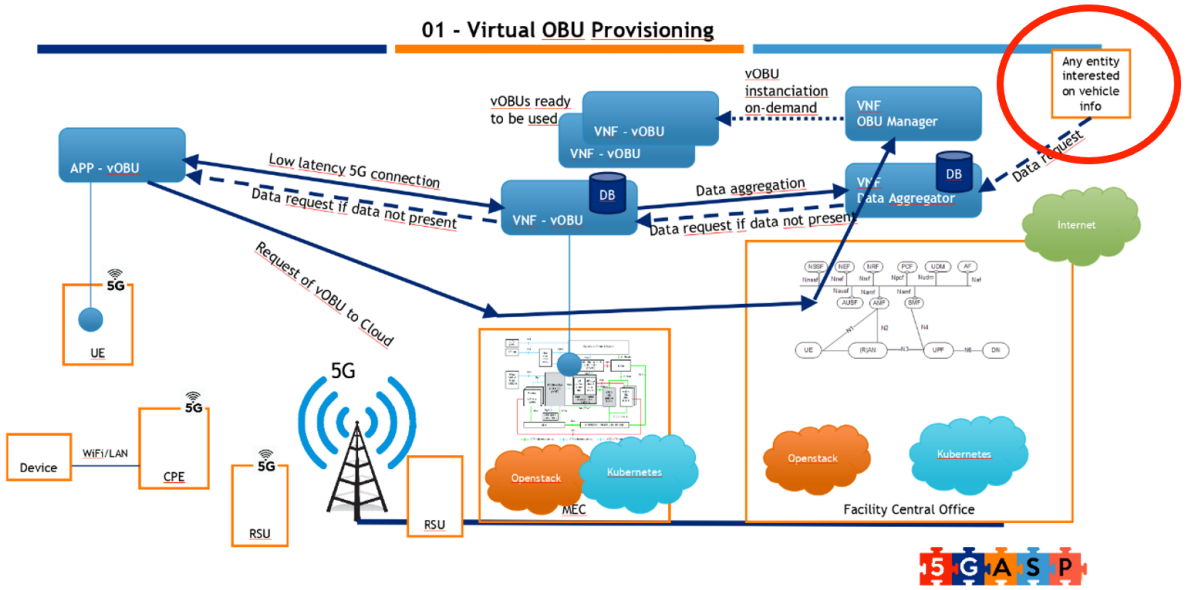

Figure 12. Virtual On-Board Unit Custom test VNF

4.2.2.3. vOBU NetApp onboarding

With the development completed, it is time to onboard the NetApp. As explained in Section 2.1.2, the NetApp is composed by the three descriptors presented above. To enable the onboarding of them, 5GASP portal will offer an interactive procedure to upload the descriptors. Once done, the triplet will be converted into a Service Deployment Order and the CI/CD pipeline will be triggered. In this way, the procedure will be as follows:

1. Uploading of the NSDs and VNFDs to the portal.
2. Uploading of the NEST representing the required network slice. Although, at this early stage, the NEST will be selected from a predefined list.
3. Selection of the KPIs to be tested in the infrastructure for the NetApp.
4. Uploading of the tests. Although, at this early stage, the tests will be selected from a predefined list.

However, while the 5GASP portal is still developed and deployed, we can test the onboarding of the NetApp by doing it manually through the NFVO and VIM of the Murcia test site. To do so, we used as NFVO the OSM 8 provided in the Gaia-5G test site directly connected to their Openstack instance. In Figure 13, it can be seen the successful onboard and deployment of the vOBU NetApp in the OSM 8 dashboard. Furthermore, Figure 14 shows the Openstack dashboard and the four VMs that have been instantiated according to the NS descriptor, as well as, all the networks that have been established.

Figure 13. OSM 8 Dashboard

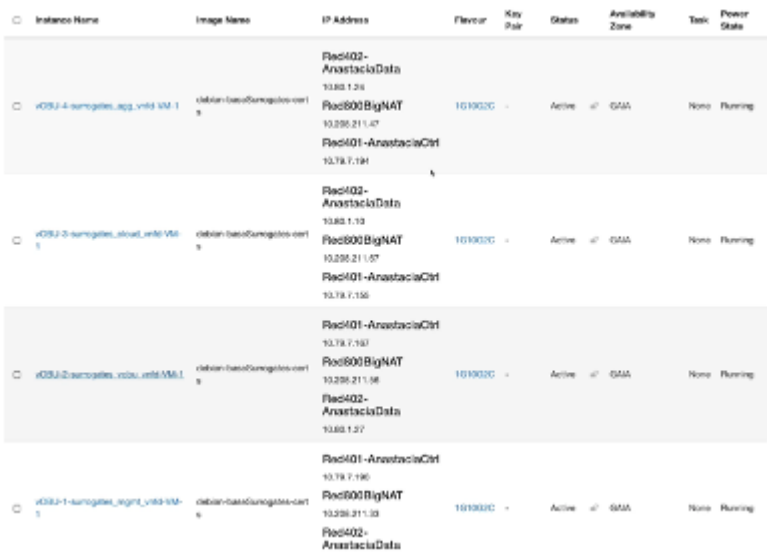

Figure 14. Openstack dashboard

4.2.2.4. vOBU NetApp testing [OdinS]

Once the vOBU NetApp is already manually deployed in the Murcia test site, the testing procedure can start. Again, as the automatic CI/CD service of the 5GASP portal is still under its design and development stages, a proof of concept can be showcased by doing it manually using the locally deployed Jenkins and Robot Framework tools. In Figure 15, it can be seen how the parametrized test information is obtained from the test descriptor and used to create the Jenkins pipeline. Once the pipeline is set up, the tests can be performed.

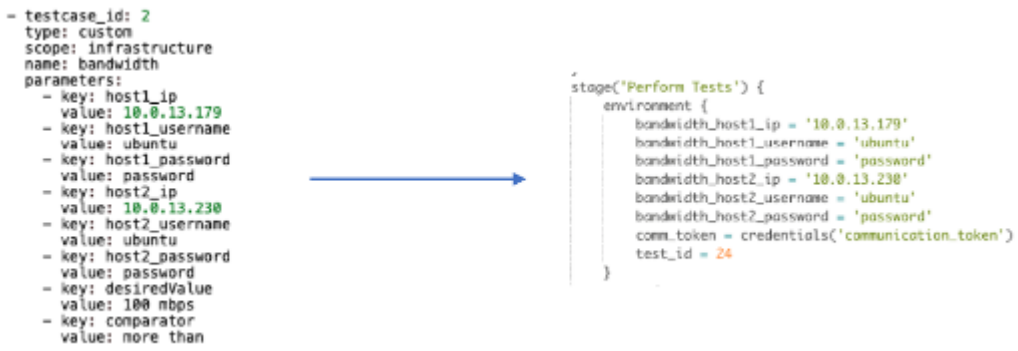

Figure 15. Test descriptor information mapped to the Jenkins pipeline

Once done, Robot Framework provides details of the execution and results of the tests both specifically for each test case and generally for the whole pipeline. In this case, the executed test measures the latency between the OBU and the vOBU using a simple tool such as ping. In Figure 16, it can be seen the output of the ping performed by the test and the average result obtained, which complies the imposed requirement of being less than 500 milliseconds. Thus, the test is successfully passed. Also, Figure 17 shows a general overview of the performed tests, although, in this PoC, it only shows the statistics of one test case.

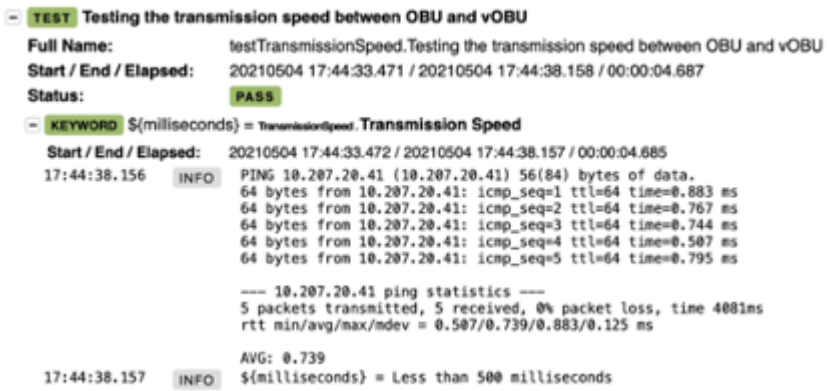

Figure 16. X Robot Framework test-case results

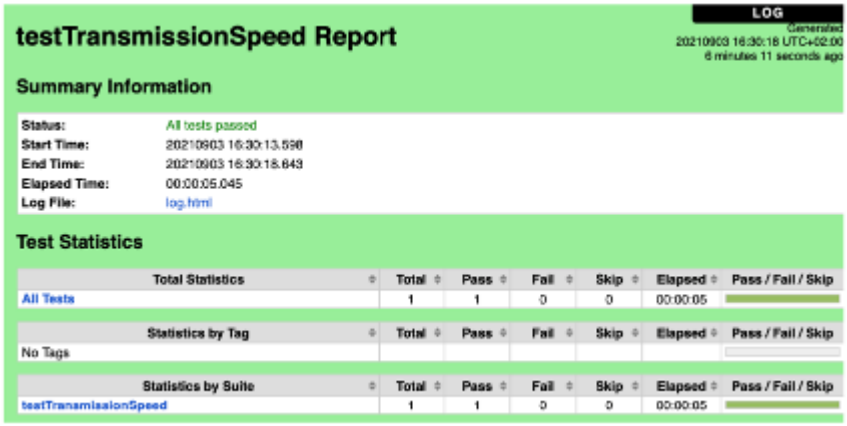

Figure 17. Robot Framework general results

4.2.3. PPDR IOPS PoC NetApp

The Isolated Operations NetApp should be provisioned via two deployment scenarios, i.e. standalone and distributed. Standalone deployment represents the basic deployment scenario where both NetApp components are placed in the edge and are providing isolated 5G services to PPDR users while the distributed deployment scenario supports separating 5G core function from the RAN function on a network function level.

4.2.3.1. PPDR IOPS PoC NetApp design

To support both deployment scenarios, two distinct VNF will be developed: Mobile Core VNF (CN VNF) and Cloud Baseband Unit VNF (Cloud BBU VNF). VNFs can be instantiated via various network service templates to support mentioned deployment options. For easier distinction, we will address Mobile Core VNF with regards to where it is deployed, i.e. Public CN VNF and IOPS CN VNF.

To better illustrate the applicability of the PPDR IOPS NetApp the scenario is broken down into several phases:

- Regular day-to-day operation phase, where the NetApp is deployed in the distributed manner, internet access and 5G services are provided to PDDR users from the public 5G core network (IOPS CN VNF is deployed but not active).

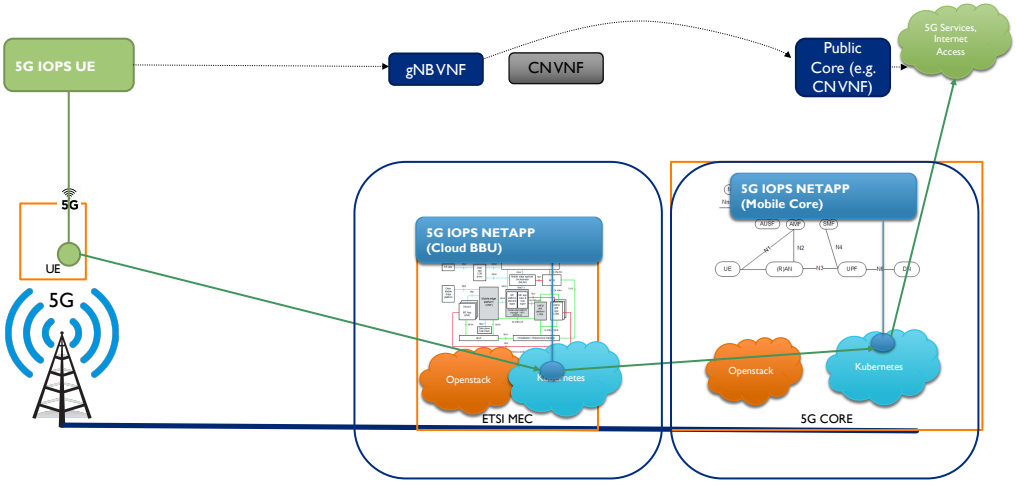

Figure 18. PPDR IOPS PoC NetApp Scenario 1

- Disaster phase that causes the uplink failure from gNB to public 5G mobile core, users can not use the 5G services and internet access.

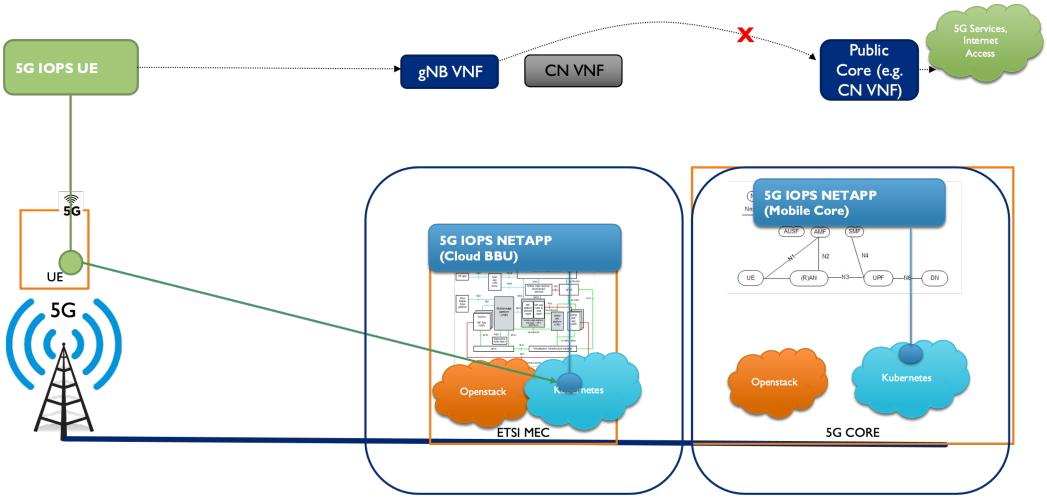

Figure 19. PPDR IOPS PoC NetApp Scenario 2

Isolated operations phase where gNB detects uplink failure (based on the healthcheck mechanism implemented in the gNB VNF) and it triggers the reconfiguration of the gNB to use the IOPS CN VNF provisioned on the edge and provide users with the basic 5G services.

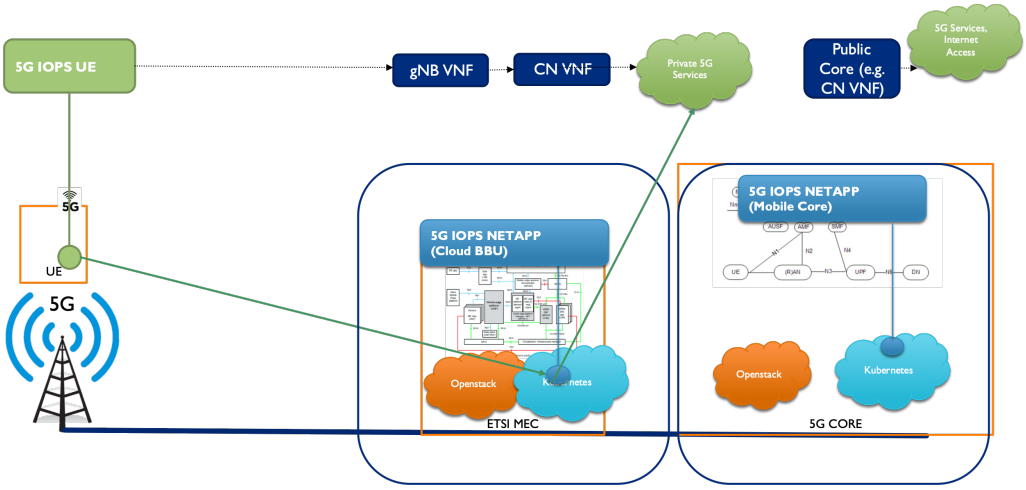

Figure 20. PPDR IOPS PoC NetApp Scenario 3

Based on the presented scenario some of the generic requirements for the PPDR IOPS VNFs can be extracted and specified:

- Packaging: VM (container inside VM) ready to run on OSM10 as a VNF
- Hardware resources: 4x CPU cores, 16GB RAM, 20GB storage
- Placement: 10 ms
- Internet access: yes (only if 5G CN VNF placed in core network)

The PPDR IOPS NetApp does not have any dependencies on the other NetApps but can serve as a PPDR platform for other NetApps and use cases.

The NetApp will use a single network slice based on the some well-known PPDR-related requirements which can be described with the NEST template. Below are stated some of the requirements:

- Area of service and region specification: Ljubljana (Slovenia).
- Isolation level: virtual resources.
- Mission-critical support: inter-user prioritization.
- Slice quality 5GPP 5QI: 69, 70.
- Maximum PLR: 10^-6.
- Supported UE velocity: pedestrian.

In the end, the design phase includes a series of tests that should be executed to assure proper operation and performance of the NetApp which will allow PPDR IOPS NetApp certification in later stages. A collection of test cases will be defined and implemented to measure the KPIs regarding the infrastructure and deployment (i.e. NetApp can be deployed, the time need to deploy), functional testing (i.e. if the NetApp is operating as intended) and performance testing (i.e. latency, throughput). The selected KPIs are presented in the following table.

Generic KPI name	**Metric Indicator (How)**	**KPI value**	**KPI unit**
Instantiation time	Time required to provision and deploy the 5G IOPS NetApp	900	Seconds
Reconfiguration time	Time required to reconfigure the 5G IOPS NetApp (i.e. change the 5G slice)	300	Seconds
RTT	Round trip time measured with qMON monitoring application from the client device connected to 5G IOPS NetApp to the 5G Core	<10	Miliseconds
DNS Reply Time	Time required to get reply from a DNS server measured with qMON monitoring application from the client device connected to 5G IOPS NetApp to the 5G Core	< 20	Miliseconds
L3 Bandwidth	Available IP download and upload bandwidth measured with qMON monitoring application from the client device connected to 5G IOPS NetApp to the 5G Core	>300 (SA mode, 2x2 MIMO, n78 TDD)	Mbps

L4 Bandwidth	Available HTTP/FTP download bandwidth and FTP upload measured with qMON monitoring application from the client device connected to 5G IOPS NetApp to the 5G Core	>250 (SA mode, 2x2 MIMO, n78 TDD)	Mbps
Web	Mean opinion score (MOS) for a selected website(s) measured with qMON monitoring application from the client device connected to 5G IOPS NetApp to the 5G Core	>4	1-5

Table 12 PPDR IOPS NetApp testing KPIs

4.2.3.2. PPDR IOPS PoC NetApp development

The design phase steps should provide enough details to begin the development of NS and VNF descriptors for NetApp components.

The NS descriptors for Cloud BBU and Mobile Core components differ only in regards to the number of network interfaces since the Mobile Core component should also have the ability to provide internet access if the uplink is provided (thus additional "external" interface). Presented examples assume that both NS will be deployed at ININ's site in Ljubljana and internet access is available (i.e. PPDR users can reach public services).

```
nsd:nsd-catalog:
  nsd:
  - constituent-vnfd:
    - member-vnf-index: '1'
      vnfd-id-ref: ppdrone_5giops_cn_vnfd
    description: PPDRONE 5G IOPS CN NS
    id: ppdrone_5giops_cn_nsd
    name: ppdrone_5giops_cn_nsd
    short-name: ppdrone_5giops_cn_nsd
    version: '1.0'
    vld:
    - id: 5giops_mgmt
      mgmt-network: true
      name: 5G-IOPS-MGMT-FLAT
      short-name: 5G-IOPS-MGMT-FLAT
      type: ELAN
      vim-network-name: 5G-IOPS-MGMT-FLAT
      vnfd-connection-point-ref:
      - member-vnf-index-ref: '1'
        vnfd-connection-point-ref: eth0
        vnfd-id-ref: ppdrone_5giops_cn_vnfd
    - id: 5giops_cn_out
      name: 5G-IOPS-OUT
      short-name: 5G-IOPS-OUT
      type: ELAN
      vim-network-name: 5G-IOPS-FLAT
      vnfd-connection-point-ref:
```

```
    - member-vnf-index-ref: '1'
      vnfd-connection-point-ref: eth1
      vnfd-id-ref: ppdrone_5giops_cn_vnfd
```

Table 13 5G IOPS CN NSD

```
nsd:nsd-catalog:
  nsd:
  - constituent-vnfd:
    - member-vnf-index: '1'
      vnfd-id-ref: ppdrone_5giops_bbu_vnfd
    description: PPDRONE 5G IOPS BBU NS
    id: ppdrone_5giops_bbu_nsd
    name: ppdrone_5giops_bbu_nsd
    short-name: ppdrone_5giops_bbu_nsd
    version: '1.0'
    vld:
    - id: 5giops_mgmt
      mgmt-network: true
      name: 5G-IOPS-MGMT-FLAT
      short-name: 5G-IOPS-MGMT-FLAT
      type: ELAN
      vim-network-name: 5G-IOPS-MGMT-FLAT
      vnfd-connection-point-ref:
      - member-vnf-index-ref: '1'
        vnfd-connection-point-ref: eth0
        vnfd-id-ref: ppdrone_5giops_bbu_vnfd
```

Table 14 5G IOPS BBU NSD

As already mentioned, the NetApp can be deployed in standalone or distributed mode. In principle, this only affects the gNB VNF which has to intelligently trigger the reconfiguration based on the connectivity status between the gNB and the core. For this, some configuration parameters need to be passed by the gNB VNF descriptor as day 1 parameters. There are no special requirements for the CN VNF descriptors.

```
vnfd-catalog:
  vnfd:
  - connection-point:
    - name: eth0
      type: VPORT
    - name: eth1
      type: VPORT
    description: PPDRONE 5G IOPS CN VNF prepared for OSM version 10.
    id: ppdrone_5giops_cn_vnfd
    mgmt-interface:
      cp: eth0
    name: ppdrone_5giops_cn_vnfd
```

```
  short-name: ppdrone_5giops_cn_vnfd
  vdu:
  - count: 1
    description: ppdrone_5giops_cn_vnfd-VM
    id: ppdrone_5giops_cn_vnfd-VM
    image: ppdrone-5giops-cn
    interface:
    - external-connection-point-ref: eth0
      name: eth0
      type: EXTERNAL
      virtual-interface:
        bandwidth: '0'
        type: PARAVIRT
        vpci: 0000:00:0a.0
    - external-connection-point-ref: eth1
      name: eth1
      type: EXTERNAL
      virtual-interface:
        bandwidth: '0'
        type: PARAVIRT
        vpci: 0000:00:0a.0
    name: ppdrone_5giops_cn_vnfd-VM
    vm-flavor:
      memory-mb: '16384'
      storage-gb: '20'
      vcpu-count: '4'
  vendor: PPDRONE
  version: '1.0'
```

Table 15 5G IOPS CN VNFD

```
vnfd-catalog:
  vnfd:
  - connection-point:
    - name: eth0
      type: VPORT
    description: PPDRONE 5G IOPS BBU VNF prepared for OSM version 10.
    id: ppdrone_5giops_bbu_vnfd
    mgmt-interface:
      cp: eth0
    name: ppdrone_5giops_bbu_vnfd
    short-name: ppdrone_5giops_bbu_vnfd
    vdu:
    - count: 1
      description: ppdrone_5giops_bbu_vnfd-VM
      id: ppdrone_5giops_bbu_vnfd-VM
      image: ppdrone-5giops-bbu
      interface:
      - external-connection-point-ref: eth0
```

```
      name: eth0
      type: EXTERNAL
      virtual-interface:
        bandwidth: '0'
        type: PARAVIRT
        vpci: 0000:00:0a.0
    name: ppdrone_5giops_bbu_vnfd-VM
    vm-flavor:
      memory-mb: '16384'
      storage-gb: '20'
      vcpu-count: '4'
  vendor: PPDRONE
  version: '1.0'
  vnf-configuration:
    initial-config-primitive:
    - name: config
      seq: '1'
      parameter:
      # general
      - name: ssh-hostname
        value: <rw_mgmt_ip>
      - name: ssh-username
        value: ubuntu
      - name: ssh-password
        value: password
      # Define AMF address of the public core
      - name: public_core_upf_address
        value: '1.2.3.4'
      # Define AMF address of the IOPS  core
      - name: iops_core_upf_address
        value: '2.2.3.4'
      # Define PLMN for the public network
      - name: public_core_plmn
        value: '00101'
      # Define PLMN for the IOPS network
      - name: iops_core_plmn
        value: '00102'
      # Define if IOPS operation enabled by default (gNB connects directly to IOPS core)
      - name: iops_operation
        value: True
    - name: start-gnb-service
      seq: '2'
    juju:
      charm: iops
```

Table 16 5G IOPS gNB VNFD

To deploy network services also the NEST is needed. Based on the requirements gathered in the design phase the full NEST template can now be defined and is presented in the following table.

5G IOPS NETAPP's NEST	
Area of service	SI
Area of service: Region Specification	Ljubljana
Isolation level	Virtual resources isolation
Mission critical support	1: mission-critical
Mission critical support: Mission-critical capability support	1: inter-user prioritization
Mission critical support: Mission-critical service support	2: MCData
Mission critical support: Mission-critical service support	3: MCVideo
Slice quality of service parameters: 3GPP QoS Identifier (5QI)	69,70
Maximum Packet Loss Rate	10^-6
Supported device velocity	0-6km/h (pedestrianl)

Table 17 5G IOPS gNB NEST

The network slice for the 5G IOPS NetApp should be based on this template (or close as possible depending on the technology available), the example of such slice will be provisioned at ININ's site in Ljubljana.

The final step during the development phase is testing the NetApp operation. Currently, only manual end-to-end testing was performed to ensure the proper operation of the 5G IOPS services. The testing is based on qMON Monitoring Tools (commercial product by ININ) which provides network performance testing (e.g., latency, bandwidth etc.) and services testing (e.g., DNS/HTTP) by placing qMON software agents on end-user Android terminals (i.e., 5G phone) and qMON reference servers at the core (or edge).

The next steps will include preparing specific test cases, test architecture and methodology to allow automated testing of the NetApp operation by using 5GASP provided testing platform based on Jenkins and Robot Framework.

4.2.3.3. PPDR IOPS PoC NetApp onboarding

With completed steps from the development phase, the NetApp should be ready for onboarding through the 5GASP portal. The general steps that will follow are stated below:

- upload NSDs/VNFDs to the 5GASP portal,
- upload NEST template to trigger slice selection,
- upload tests.

Currently, onboarding 5G IOPS components directly to the OSM is being tested locally at ININ's premises.

4.2.4. PrivacyAnalyzer cross-vertical-PoC NetApp

PrivacyAnalyzer offers the functionalities for aiding 5G/6G/IoT integrators to assess the privacy strength of their applications or services against confidential information disclosure. Use of PrivacyAnalyzer is performed during the UAT phase before the integrators releases the solution to the clients such as the solution minimizes the risk of revealing Personally Identifiable Information (PII).

In a summary, PrivacyAnalyzer receives testing sessions' traffic between the devices (UEs, IoT devices, etc.) participating in the integrator's test sessions, analyzes the sessions' message contents for privacy-sensitive content and reports the outcomes of the analysis to the end-user (integrator) with the goal to enable the end-user to resolve the privacy issues detected in the test sessions before releasing her application or service to her client, typically a 5G network/service operator or an IoT service provider.

- High level architecture

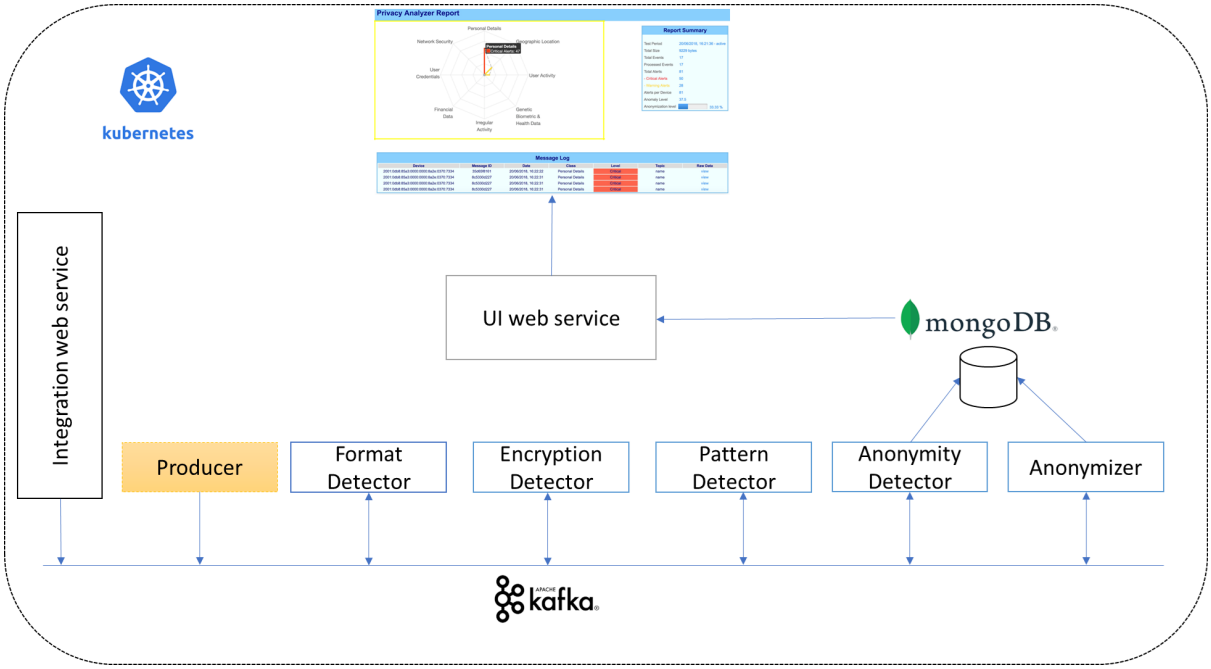

Figure 21. N High level architecture of the PrivacyAnalyzer system

4.2.4.1. PrivacyAnalyzer cross-vertical-PoC NetApp design

As depicted in the high-level architecture diagram, the system consists of a sequence of processors (a process is implemented as a docker container) that communicate with messages through a Kafka cluster with the K8s deployment of the system (Figure 21). Each processor performs specific functionality as we latter discuss. Each individual process receives messages -typically at high volume rates- from a specific Kafka topic and emits processed/enriched messages to an output Kafka topic. All topics are registered within the Kafka cluster.

In the following, we discuss the main functionality of each processor implemented in the PrivacyAnalyzer system.

Format Detector: It detects whether the data is text or binary and the actual serialization format: XML, JSON or YAML for text messages and Avro or BSON for binary messages.

Encrypt Detector: It detects whether the content is compressed. This information is important since compressed content has similar entropy properties. It returns an entropy value (as discussed in [21]; we use python3 zlib entropy implementation) in the range [0..1] where values closer to 1 denote higher entropy and a thus higher chance that the data is encrypted.

Pattern Detector: It tokenizes the message and classifies tokens to categories such as phones, addresses, emails, names, dates, etc. based on a combination of pattern matching, Named Entity Recognition (NER) and statistical analysis. The Pattern Detector also is able to detect birth years and geographic coordinates within the received messages.

Anonymity Detector: This processor classifies the detected tokens (note that tokens are produced by the Pattern Detector processor) either to de-identifiers or quasi-identifiers. E.g. email addresses are tagged as de-identifiers while detected dates are labelled as quasi-identifiers.

Anonymizer: The anonymizer processes all tokens marked as de-identifiers or quasi-identifiers by the anonymity classifier. De-identifiers are suppressed by replacing their value in the original stream with the placeholder character '*'. According to the input policy by the system administrator (see later in this sub-section) quasi-identifiers are either suppressed (gender) or generalized (dates, ages, geographic coordinates, etc.). The main objective of the Anonymizer is to detect whether the data contains de-identifiers, in which case it is de-identified, and/or it is anonymized according to the administrator policy. In case that quasi-identifiers are detected, they are either suppressed or generalized, also again according to the policy set by the administrator.

The results of the analysis performed by the 'Anonymity Detector' and 'Anonymizer' processors are stored in the MongoDB cluster. The obtained results are used by the 'Web Service' component in the above high-level architecture diagram in order to feed the Angular web app that is used by the end-user so that the latter sees the results of the overall privacy analysis. In the next sub-section, we will present implementation details on the user interface which is served by an instance of Nginx.

Note that our architecture also includes a component, named '**Stream Producer**' which is highlighted with the dashed yellow rectangle in the previous figure. 'Stream Producer' is used for testing purposes, i.e. it is not being used when the integration with the external system is performed. Its input is a large JSON file that contains a sequence of JSON-formatted records and emits a message for each record to the Kafka topic where the 'Format Detector' component is subscribed. We use various data sources to construct a large realistic JSON file that emulates behaviors containing sensitive information. Specifically, one of the data sources used is the foursquare data sets from [22]. The

'Stream Producer' and its relevant input datasets containing large volumes of JSON data will also be used in the 5GASP project in order to ensure that the system functions correctly after its onboarding. Thus, it is envisaged as a NetApp-specific testing component that will aid us to validate the correct functioning of our software when it shall operate within the 5GASP environment.

- **Administration of the PrivacyAnalyzer system**

PrivacyAnalyzer targets as end-users 5G/IoT integrator companies typically with no knowledge of privacy algorithms and technologies. The administrator User Interface must be user-friendly manner so that it can be understood without domain-specific knowledge.

The product's administrators are only required to configure a small set of configuration parameters. Specifically, the administrator is asked to choose the anonymization policy of the user's test sessions. For that purpose, a drop-down menu appears in the administrator UI from which the administrator selects one of the following policies {'Full Anonymization', 'Suppression of Identifiers', 'Suppression of User Activity}.

The administrator also configures the user access via the administrator UI. Overall, it is not mandatory that the administrator has any knowledge of the details of the privacy analysis components of the PrivacyAnalyzer 'core' sub-system as it is depicted in the high-level architecture of the system.

The product users can only view their dedicated User Interface which includes the privacy analysis of their devices and the history of the analysis of their devices. It is mandatory that no other user can access a specific user's devices and relevant datasets.

- **End-user functionality**

The PrivacyAnalyzer UI provides the following functionality to the end-user:

1. A report summary that holds aggregate statistics regarding the test session such as the number of processed messages and their total size in bytes.

2. A radar chart that renders the privacy alerts in distinct categories.

3. A message logs section. The message log contains important information such as the IPv6 address of the device, the message ID, the class, and the level of anonymization.

For the context of 5GASP use-cases, the deployment of the PrivacyAnalyzer NetApp is described in Deliverable D2.1 [23], in section 2.8.

4.2.4.2. Privacy analyzer cross-vertical-PoC NetApp development

The CNF descriptor of the 3-Tier PrivacyAnalyzer system is provided in the following table.

CNF of the PrivacyAnalyzer NetApp
nfd: id: privacyanalyzer_knf description: KNF of PrivacyAnalyzer system using KDUs packages via helm-chart v3 df: - id: default-df k8s-cluster: nets: - id: mgmtnet kdu: - name: producer helm-chart: lamdaleon/producer kdu: - name: format_detector helm-chart: lamdaleon/format_detector kdu: - name: encrypt_detector helm-chart: lamdaleon/encrypt_detector kdu: - name: pattern_detector helm-chart: lamdaleon/pattern_detector kdu: - name: anomym_detector helm-chart: lamdaleon/amonym_detector kdu: - name: anonymizer helm-chart: lamdaleon/amonymizer kdu: -name: nginx helm-chart: bitnami/nginx kdu: -name: mongodb helm-chart: bitnami/mongodb

ext-cpd: - id: mgmt-ext
k8s-cluster-net: mgmtnet
mgmt-cp: mgmt-ext
product-name: privacyanalyzer_knf
provider: Lamda Networks
version: ‘2.0’

Table 18 VNFD of PrivacyAnalyzer

The CNF consists of the KDUs described in the architecture of the PrivacyAnalyzer system. Each KDU is packaged with Helm 3 and the services to be exposed use a single external CP to the Kubernetes cluster running as VIM inside the OSM deployment of the target testbeds.

The NS of the PrivacyAnalyzer NetApp uses the above KNF definition. It is described in the following table.

NS of the PrivacyAnalyzer NetApp
nsd: nsd: - description: NS consisting of PrivacyAnalyzer KNF connected to mgmt network id: privacyanalyzer_ns name: privacyanalyzer_ns version: ‘2.0’ vnfd-id: - privacyanalzer_knf virtual-link-desc: - id: mgmtnet mgmt-network: 'true' df: - id: default-df vnf-profile: - id: privacyanalyzer vnfd-id: privacyanalyzer_knf virtual-link-connectivity:

```
- constituent-cpd-id:
  - constituent-base-element-id: privacyanalyzer
    constituent-cpd-id: mgmt-ext
    virtual-link-profile-id: mgmtnet
```

Table 19 PrivacyAnalyzer VNFD

Finally, the KPIs and the NEST associated with the PrivacyAnalyzer NetApp are described in Deliverable 2.1 [23] in section 2.8.

4.2.4.3. Privacy analyzer cross-vertical-PoC NetApp onboarding

The KNF and NS of the PrivacyAnalyzer system are have been onboarded on the private Google Cloud of Lamda Networks on a OSM 10 deployment on Ubuntu 18.04 which connects to a K8s cluster inside the Google Cloud

The onboarding shall be realized in the Patras 5G testbed on December 21, 2021.

5. NetApps status

NetApp	Owner	Codebase	VNFD/NSD	GST/NEST	Tests	Onboarding
Virtual On-board Unit (vOBU)	OdinS	Available	Developed	Developed	Ongoing (Q1-2 2022)	Local (Q1-2 2022)
Virtual RoadSide Unit (vRSU)	YoGoKo	Partially available (Commercial product)	In design (Q4 2021)	In design (Q4 2021)	No (Q2-3 2022)	No (Q3-4 2022)
Cooperative Central ITS Station (vITS-S)	YoGoKo	Partially available (Commercial product)	In design (Q4 2021)	In design (Q4 2021)	No (Q2-3 2022)	No (Q3-4 2022)
Multi-domain Migration	OdinS	Available	In design (Q4 2021)	Developed	No (Q2-3 2022)	No (Q1-2 2022)
Vehicle-to-cloud Real-Time Communication	BLB/DriveU	Partially available (Commercial product)	In design (Q2 2022)	In design (Q1 2022)	Designed (Q2 2022)	In design (Q2 2022)
Remote Human Driving	BLB/DriveU	Partially available (Commercial product)	In design (Q2 2022)	In design (Q1 2022)	Designed (Q2 2022)	In design (Q2 2022)
Efficient MEC handover	UNIVBRIS	Available	Designed	In design (Q1-2 2022)	No (Q2-3 2022)	Local (Q3-Q4 2022)
Privacy Analyzer	Lambda	Available	Designed	Designed	In design	Local (Q4 2021)
5G Isolated Operations	ININ	Available	Designed (Q4 2021)	Designed (Q4 2021)	Ongoing (Q1-2 2022)	Ongoing (Q1- Q2 2022)
Vehicle Route Optimizer (VRO)	Neobility	Available prototype (Q1 2022)	Designed (Q1 2022)	Designed (Q1 2022)	Ongoing (Q2-Q3 2022)	To be started (Q3-Q4 2022)
Fire detection and ground assistance using drones (FIDEGAD)	UoP	Available prototype (Q1 2022)	Designed (Q1 2022)	Designed (Q1 2022)	No	No

6. Conclusions

This deliverable presented the agreed and recommended final methodology for designing and developing NetApps by introducing a well-defined and example-based approach to encourage and facilitate NetApp's development by SMEs from various verticals. In addition, the proposed methodology provided ready-to-use templates to enable the automated and reproducible validation of NetApps across multiple facility sites, including inter-domain scenarios. The ultimate goal is to initiate an operational platform with a reference ecosystem to validate, deploy, and certify 5G experiments.

Progressing from the initially proposed methodology in D4.1, this deliverable concludes the feedback and recommendations from the involved SMEs in the 5GASP project and profiting from previously conducted projects, such as 5GTANGO and 5GEVE. Therefore, the proposed final methodology is an evolution of accumulated approaches that industrial and academic partners have used.

Moreover, the final methodology respects the pre-defined requirements of the project reference architecture in WP2. It was also essential to consider the feedback evolved in WP3 and conform to the preparatory steps that connect with the testing, validation, and certification phases proposed in WP5.

The onboarding and basic testing workflows are described in this methodology and exemplified with step-by-step details targeting the SMEs developers and explaining how the NetApps will be managed and how to be composed, using a provided ready-to-use descriptors templates meant to facilitate the implementation of Netapps. The provided examples and guidelines considered the design and development of three types of PoC NetApps correspond to the Automotive, PPDR, and Privacy analyzer verticals.

Bibliography

[1] 5GASP, "D4.1 Initial Methodology for Designing and Developing NetApps".

[2] [Online]. Available: https://osm.etsi.org/docs/vnf-onboarding-guidelines/01-requirements.html#day-0-requirements.

[3] O. ETSI. [Online]. Available: https://osm.etsi.org/docs/vnf-onboarding-guidelines/#welcome-to-open-source-mano-s-vnf-onboarding-guide .

[4] "IETF, “YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF),” 2010.".

[5] "OASIS, “Instance Model for TOSCA Version 1.0,” 2017.".

[6] TMF, "TMF633 Service Catalog API User Guide v4.0.0," TMF, 2020.

[7] [Online]. Available: https://helm.sh/.

[8] 5GASP, "D3.1 5GASP experimentation services, middleware and multi-domain facilities continue integration".

[9] E. G. N.-S. 0. V. (.-0. N. F. V. (. R. 2, Security and V. P. S. Specification. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/021/02.06.01_60/gs_NFV-SEC021v020601p.pdf.

[10] TMF, "TMF641 Service Ordering API User Guide v4.1.0," TMF, 2020.

[11] 5GASP, " D5.1 Initial report on test-plan creation and testing".

[12] 5GASP, "D4.1 Initial Methodology for Designing and Developing NetApps".

[13] 5GASP, "WP-5 NetApps Onboarding, Interoperability and Validation," [Online].

[14] "IEEE 610-1990 - IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries".

[15] 5GTANGO, "D4.1 First open-source release of the SDK toolset," 5GTANGO, 2017.

[16] [Online]. Available: https://landscape.cncf.io/.

[17] ETSI, "5G;System Architecture for the 5G System (3GPP TS 23.501 version 15.2.0 Release 15)," ETSI, 2018.

[18] GSMA, "Generic Network Slice TemplateVersion 4.0," GSMA, 2020.

[19] "5GEVE," [Online]. Available: https://www.5g-eve.eu/.

[20] "5GTANGO," [Online]. Available: https://www.5gtango.eu/.

[21] [Online]. Available: https://www.euccas.me/zlib/.

[22] [Online]. Available: https://sites.google.com/site/yangdingqi/home/foursquare-dataset..

[23] 5GASP, "D2.1. Architecture, Model Entities Specification and Design," 5GASP, 2021.

[24] "5GVINNI," [Online]. Available: https://www.5g-vinni.eu/.

[25] "ETSI GS NFV-SOL 005. Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point. V3.3.1," ETSI, 2020-09.

[26] M. Z. e. al., "Verification and validation framework for 5G network services and apps," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017.

[27] IEEE, "IEEE 610-1990 - IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries," IEEE.

[28] IEEE, "ETSI GS NFV-TST 001. Network Functions Virtualisation (NFV);Pre-deployment Testing; Report on Validation of NFV Environments and Services. V1.1.1," IEEE, 2016.