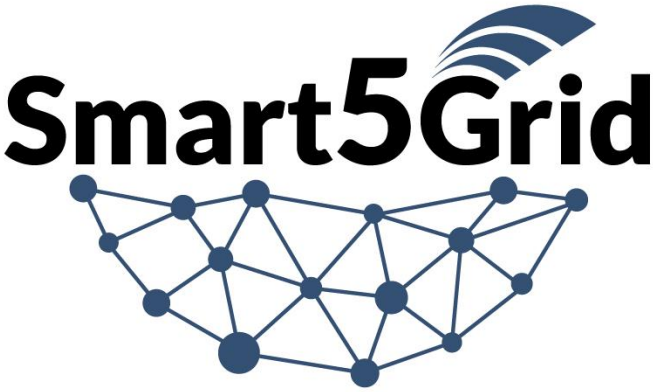


Demonstration of **5G** solutions for
SMART energy **GRID**s of the future

Deliverable 3.1

Interim Report for the development of the 5G network facilities

Version 1.0 - Date 31/01/2022

The 5G Infrastructure Public Private Partnership

This project has received funding from the European Union's *Horizon 2020 research and innovation programme* under grant agreement n° 101016912

D3.1 – Interim Report for the development of the 5G network facilities

Document Information

Programme	Horizon 2020 Framework Programme – Information and Communication Technologies
Project acronym	Smart5Grid
Grant agreement number	101016912
Number of the Deliverable	D3.1
WP/Task related	WP3
Type (distribution level)	PU Public
Date of delivery	31-01-2022
Status and Version	Version 1.0
Number of pages	80 pages
Document Responsible	Antonello Corsi ENG
Author(s)	Anastasios Lytos – SID Andrés Cárdenas – i2CAT Angelos Antonopoulos – NBC Athanasios Bachoumis – UBE Arif Ishaq – ATH August Betzler – i2CAT Borja Otura – ATOS Daniele Munaretto – ATH Dimitris Brodimas – IPTO Eugenia Vergi – INF Gaetano Scibilia – W3

	Gianluca Rizzi – W3
	Ioannis Chochliouros – OTE
	Irina Ciornei – UCY
	Juliana Teixeira – UW
	Lenos Hadjidemetriou – UCY
	Luigi Sechi – STAM
	Rita Santiago – UW
	Marco Centenaro – ATH
	Markos Asprou – UCY
	Nicola Cadenelli – NBC
	Nicola di Pietro – ATH
	Paula Encinar – ATOS
	Sonia Castro – ATOS
	Theocharis Saoulidis – SID
	Theoni Dounia – INF
	Yanos Angelopoulos – AXON
	Michalis Rantopoulos – OTE
Reviewers	Giovanni Frattini – ENG
	Gaetano Scibilia – WI3

Revision History

Version	Date	Author/Reviewer	Notes
0.1	09/09/2021	Antonello Corsi	Initial table of content for feedback
0.2	20/09/2021	Antonello Corsi	First round of feedback received and applied
0.3	10/10/2021	Andrés Cárdenas	First round of information regarding NetApp controller
0.4	1/11/2021	Otura Borja	First round of contributions related to the virtualized infrastructure.
0.5	10/11/2021	Angelos Antonopoulos Nicola Cadenelli	Add NetApp controller info
0.6	15/11/2021	Rita Santiago	Add V&V related info
0.7	30/11/2021	Nicola di Pietro	Acceptance of all previous major modifications. Major updates to Section 2 and 3.
0.8	21/12/2021	Nicola di Pietro	Acceptance of all previous modifications. Final major contributions to Section 2, 3, 4.
0.9	11/01/2022	Otura Garcia, Borja	Editorial work
0.10	12/01/2022	Antonello Corsi	Final version of the deliverable and acceptance and verification of all comments. First version available to the peer reviewers.
0.11	21/01/2022	Giovanni Frattini	Peer reviers
0.12	21/01/2022	Giampaolo Fiorentino	Peer reviers
0.13	21/01/2022	Gaetano Scibilia	Peer reviers
1.0	25/01/2022	Antonello Corsi	Final version available

Executive summary

The objective of this deliverable is to present the initial version of the work done in WP3 as Interim report for the development of the 5G network facility. As key contributions, it defines the initial version of components development and interfaces realization so to cover the programming interfaces and the information model that allow the proper operations of the Smart5Grid platform in coordination with the NFVO and energy infrastructures. The performed analysis has a project-wide resonance and motivates the decision to adapt different technological and development choices in platform in the light of the specific Smart5Grid ecosystem requirements. The provided document will direct the development work of the Smart5Grid platform from the WP2 (architecture), to WP4 & WP5 in which the NetApp and the fine tune of the UC pilot installation will be managed. Further details about the actual implementation will be described in future deliverable releases named as D3.2 & D3.3. Also, the knowledge that is likely to be learned during the development and integration phases will be taken into account and retransferred in future deliverables.

Table of contents

Revision History 4
Executive summary 5
Table of contents 6
List of figures 9
List of tables 11
1. Introduction 12
1.1. Scope of the document 12
1.2. Document Structure 12
1.2.1. Notations, abbreviations and acronyms 12
2. Architectural components and interfaces 18
2.1. Smart5Grid Platform 19
2.1.1. User Interfaces 19
2.1.2. Open Service Repository 21
2.1.2.1. OSR Road Map 25
2.1.3. Verification and Validation Framework 25
2.1.3.1. V&V Roadmap 28
2.1.3.2. V&V Engine 28
2.1.3.3. Request Handler/API 31
2.1.3.4. Results Manager 32
2.1.3.5. M&O Adapter/Translator 32
2.2. NetApp Controller & MEC Orchestrator 32
2.2.1. NetApp Controller and MEC Orchestrator Roadmap 34
2.3. Slice Manager 35
2.4. NFV central and edge infrastructure 39
2.4.1. NFV architecture 39
2.4.2. Cloud infrastructure for the deployment of the Smart5Grid platform 43
2.4.3. Virtualization infrastructure in UC1 45
2.4.4. Virtualization infrastructure in UC2 45
2.4.5. Virtualization infrastructure in UC3 45
2.4.6. Virtualization infrastructure in UC4 45
2.5. 5G infrastructure 46

2.5.1. Core network 46
2.5.2. Radio access network 48
2.5.2.1. The evolution of cellular technology 48
2.5.2.2. 5G Spectrum: Frequency Bands 49
2.5.2.3. RAN infrastructure in Smart5Grid's use cases 49
2.6. Telemetry and data analytics 55
2.6.1. Technological choice for the telemetry and data analytics module 56
2.6.2. Data pipeline 56
3. NetApp operation support and orchestration 60
3.1. Infrastructure management 60
3.1.1. Resource registration 60
3.1.1.1. Slice manager: registration of infrastructure resources 60
3.1.1.2. NetApp Controller & MECO and VIM: Node registration and provisioning 61
3.1.2. Resource monitoring 61
3.2. NetApp lifecycle 62
3.2.1. NetApp onboarding 63
3.2.2. NetApp instantiation 64
3.2.3. NetApp operation 66
3.2.4. NetApp termination and decommissioning 67
4. Integration guidelines 68
4.1. Introduction 68
4.2. Software 68
4.2.1. Conventions 68
4.2.2. Development phase 68
4.2.3. Patterns 68
4.2.4. Code comments 68
4.3. Conventions related to integration 69
4.3.1. Respect over Interfaces and Data Models 69
4.3.2. Source Commits 69
4.4. CI/CD 70
4.4.1. Continuous Integration in Smart5Grid 70
4.5. Testing and Integration of the different components 72
4.5.1. Integration testing 72

4.6. Smart5Grid platform integration 74
4.6.1. Interaction between OSR and V&V 74
5. Pre-piloting /Hardware in the loop /Energy Infrastructure 76
6. Conclusion and next steps 79
7. References 80

List of figures

Figure 2-1 - Smart5Grid functional architecture. 18

Figure 2-2 - V&V platform architecture. 29

Figure 2-3 - NetApp components example. 30

Figure 2-4 – NetApp Controller Components. 33

Figure 2-5 - NFV reference architectural framework [3]. 39

Figure 2-6 - High-level representation of a 5G system. 45

Figure 2-7 - The 5GC network and its interfaces with the elements of a 5GS. 47

Figure 2-8 - UPF deployment for traffic localization at the edge. 48

Figure 2-9 - UC3 Wind farm demo case topology 51

Figure 2-10 - 5G Hat technical specification. 52

Figure 2-11 - 5G NSA architecture of UC3. 52

Figure 2-12 - 5G SA architecture of UC3. 53

Figure 2-13 - Zyxel NR7101 5G New Radio outdoor router to be used for PMUs coverage at UC4. 53

Figure 2-14 - Interface description, Zyxel NR7101. 54

Figure 2-15 - Ericsson's Radio 4408 to be installed at IPTO sub-station. 54

Figure 2-16 - Confluent platform components extracted from [9] 56

Figure 2-17 - Apache Kafka architecture. 57

Figure 2-18 - Telemetry pipeline. 58

Figure 3-1 - Compute node registration workflow. 60

Figure 3-2 - RAN device registration workflow. 61

Figure 3-3 - NetApp lifecycle. 63

Figure 3-4 - Network preparation and resource allocation. 65

Figure 4-1 : CI/CD 71

Figure 4-2 : Integration template 74

Figure 4-3 - Data flow 74

Figure 5-1 - Pre-piloting testbed facility. 76

List of tables

Table 1: Acronyms list 17

Table 2-1 – Interfaces and methods required by the UI. 19

Table 2-2 – Interfaces and methods provided by the OSR. 21

Table 2-3 – Interfaces and methods required by the OSR. 24

Table 2-4 – Interfaces and methods provided by the V&V framework. 26

Table 2-5 – Interfaces and methods required by the V&V framework. 26

Table 2-6 – Interfaces and methods provided by the NetApp Controller & MECO. 33

Table 2-7 – Interfaces and methods required by the NetApp Controller & MECO. 34

Table 2-8 – Interfaces and methods provided by the slice manager. 37

Table 2-9 – Interfaces and methods required by the slice manager. 38

Table 2-10 – Interfaces and methods provided by the NFVO. 40

Table 2-11 – Interfaces and methods required by the NFVO. 42

Table 2-12 - Hardware employed for the lab testing of the Smart5Grid platform. 44

Table 2-13 - Evolution of networks and services for each generation. 48

Table 2-14 - Collectable metrics. 58

Table 4-1 : Time schedule 75

1. Introduction

1.1. Scope of the document

As stated in the executive summary, D3.1 aims to provide collectively the work performed and the results of WP3 in the first year of the project. In particular, it sets the groundwork for implement the Smart5Grid 5G platform and to enable network slicing, multi-access edge computing, for services in the energy vertical as well as the Smart5Grid OSR for NetApp development specifications. An initial plan regarding how the different modules and layers of the Smart5Grid platform will be integrated and how the pre-piloting testing facilities will be managed.

These overall concepts will serve the scope to provide an open and flexible platform for secure testing, validation, verification, and operation of NetApp not only to Smart5Grid partners but also to third parties.

This deliverable covers the activities performed as part of the "*Task 3.1: Smart5Grid network access/core platform building and orchestration*", "*Task 3.2: Open Service Repository*", "*T3.3: Platform integration and interfacing between 5G and energy infrastructure*" and finally "*Task 3.4 Pre-piloting via Hardware-in-the-loop demonstration*"

1.2. Document Structure

The deliverable is organized in the following manner:

Section 2 proposes an overview of the architectural components and interfaces of the Smart5Grid platform and the relationship with the NFV and Energy layer. This analysis allows the role of the different modules to be situated precisely within the main object to serve the deployment of the NetApp. With this aim the section 3 is devoted to analyse the NetApp operation and related orchestrations functionalities offered by the NetApp Controller. Section 4 pave the way to define the CI/CD approach followed with the main intentions to provide artefact software ready and tested and on boarded on the operational phase. With this aim a first version of the road map for the modules that compose the Smart5Grid Platform is provided.

In section 5 is illustrated the pre-piloting testing phase that refers to the testing of the use-case specific NetApp from the point of view of ensuring the necessary quality and functionality of the energy services they were designed for.

The deliverable is then closed by the final section about conclusions and next steps.

1.2.1. Notations, abbreviations and acronyms

Item	Description
3GPP	3rd Generation Partnership Project
5G	5th Generation (of mobile telecommunication networks)
5G PPP	5G Infrastructure Public Private Partnership

A&A	Authentication and Authorization
AAA	Authentication, Authorization and Accounting
AB	Advisory Board
AF	Application Function
AKA	Authentication and Key Agreement
AMF	Access Management Functions
AMI	Advanced Metering Infrastructure
API	Application Programming Interface
BIOS	Basic Input/Output System
BRP	Balancing Responsible Party
BSP	Balancing Service Provider
BSS	Battery Storage System
BBU	Base Band Unit
CA	Consortium Agreement
CIR	Container Image Registry
CISM	Container Infrastructure Service Management
CFS	Customer Facing Service
CLI	Command Line Interface
CM	Configuration Management
CoE	Centre of Excellence
CN	Core Network
CNF	Containerized Network Function
CP	Connection Point
CPD	Connection Point Descriptor
CPF	Control Plane Function
CT	Current Transformer
CRAN	Cloud RAN
CRUD	Create, Read, Update, Delete
CSMF	Communication Service Management Function
CUPS	Control and User Plane Separation
DCN	Data Communication Network
DER	Distributed Energy Resources
DevOps	Development and Operations
DHCP	Dynamic Host Configuration Protocol
DLT	Distributed Ledger Technology
DNAI	Data Network Access Identifier
DNN	Data Network Name
DNS	Domain Name System
DoW	Description of Work
DRES	Distributed renewable energy sources

DSO	Distribution System Operator
EDSO	European Distribution System Operators for Smart Grids (non-profit association)
EEGI	European Electricity Grid Initiative
ENTSO-E	European Network of Transmission System Operators for Electricity
EPC	Evolved Packet Core
EPIA	European Photovoltaic Industry Association
ETSI	European Telecommunications Standards Institute
EU	European Union
EWEA	European Wind Energy Association
FM	Fault Management
FP7	Seventh Framework Program
GA	Grant Agreement
GDS	Global Digital Services
GIT	Global Information Tracker
GOOSE	Generic Object-Oriented Substation Even
GPIO	General Purpose Input Output
GPS	Global Positioning System
gRPC	Google Remote Procedure Calls
GUI	Graphical User Interface (GUI)
HSS	Home Subscriber Server
HTTP(S)	Hypertext Transfer Protocol (Secure)
IAP	Identity and Access Proxy
ICT	Information and Communications Technologies
IMT	Information Model Translation
IPMI	Intelligent Platform Management Interface
iPXE	Preboot eXecution Environment
ISO	Optical Disc Image
IT	Information Technology
KPI	Key Performance Indicator
LAN	Local Area Network
LBO	Local Break-Out
LCM	LifeCycle Management
LI	Lawful Interception
LTE	Long-Term Evolution
LV	Low Voltage
M&O	Management and Orchestration[1]

MANO (NFV)	Management and Orchestration1
MCIO	Managed Container Infrastructure Object
MCIOP	Managed Container Infrastructure Object Package
MECO	Multi-access Edge Computing Orchestrator
MME	Mobility Management Entity
MMS	Manufacturing Message Specification
MNO	Mobile Network Operator
MSA	Micro-Service Architecture
MV	Medium Voltage
NBI	Northbound Interface
NetApps	Network Applications
NEF	Network Exposure Function
NFMF	Network Function Management Function
NFVI	Network Function Virtualization Infrastructure
NFVIaaS	NFV Infrastructure-as-a-Service
NFVO	Network Function Virtualization Orchestrator
NGMN	Next Generation Mobile Networks
NR	New Radio
NRF	Network Resource Function
NS	Network Service
NSaaS	Network Slice-as-a-Service
NSD	Network Service Descriptor
NSFVal	Network Services and Functions Validator
NSI	Network Slice Instance
NSMF	Network Slice Management Function
NSSI	Network Slice Subnet Instances
NSSF	Network Slice Selection Function
NSSMF	Network Slice Subnet Management Function
nZTP	near Zero-Touch Provisioning
OAM	Operations, Administration, and

1 Through this deliverable, "Management & Orchestration" is abbreviated as MANO (or NFV MANO) when referring to the Management and Orchestration as defined by ETSI NFV and M&O when referring to the Management and Orchestration framework of the NFV/TELCO layer of the Smart5Grid Architecture.

	Management
ONAP	Open Network Automation Platform
OPC	Open Platform Communications
OPC-UA	OPC Unified Architecture
OSM	Open-Source Mano
OSR	Open Service Repository
OSS/BSS	Operational Support Systems / Business Support Systems
PAS IEC	Publicly Available Specification
PC	Personal Computer
PCF	Policy Control Function
PDC	Phasor Data Concentrator
PFD	Packet Flow Description
PGW	Packet Data Network Gateway
PM	Performance Management
PMU	Phasor Measurement Unit
PoP	Points-of-Presence
PPDR	Public Protection and Disaster Relief
PST	Power System Testbed
PV	Photovoltaics
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RES	Renewable Energy Sources
REST	REpresentational State Transfer
ROCOF	Rate of Change of Frequency
ROS	Robotics Operating System
RSC	Regional Security Coordinator
RSU	Road Side Units
RTD	Research and Technology Development.
RT-HIL	Real-Time Hardware-In-the Loop
RTS	Real-Time Simulator
RTU	Remote Terminal Unit
SAP	Service Access Point
SBA	Service-Based Architecture
SBI	Southbound Interface
SCADA	Supervisory Control and Data Acquisition
SDK	Service Development Kit
SEAF	Security Anchor Functionality
SGW	Serving Gateway
SLI	Service Level Indicator
SLO	Service Level Objective

SME	Small and Medium Enterprise
SMF	Session Management Function
SOA	Service-Oriented Architecture
SP	Service Providers
SSH	Secure Shell
T&D	Transmission and Distribution
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TPM	Trusted Platform Module
TSO	Transmission System Operator
UC	Use Case
UCY	University of Cyprus
UE	User Equipment
UDM	Unified Data Management
UL	Up-link
UPF	User Plane Function
UWB	Ultra-Wideband
V&V	Verification and Validation
VDU	Virtual Deployment Unit
VIM	Virtual Infrastructure Manager
VNF	Virtual Network Function
VNFC	Virtual Network Function Component
VNFaaS	Virtual Network Function-as-a-Service
VNFM	Virtual Network Function Manager
VDU	Virtual Deployment Unit
VLD	Virtual Link Descriptor
VSB	Vertical Service Blueprint
VSD	Vertical Service Descriptor
vPDC	virtual Phasor Data Concentrator
VPN	Virtual Private Network
VT	Voltage Transformer
WAM	Wide Area Monitoring
WP	Work Package

Table 1: Acronyms list

2. Architectural components and interfaces

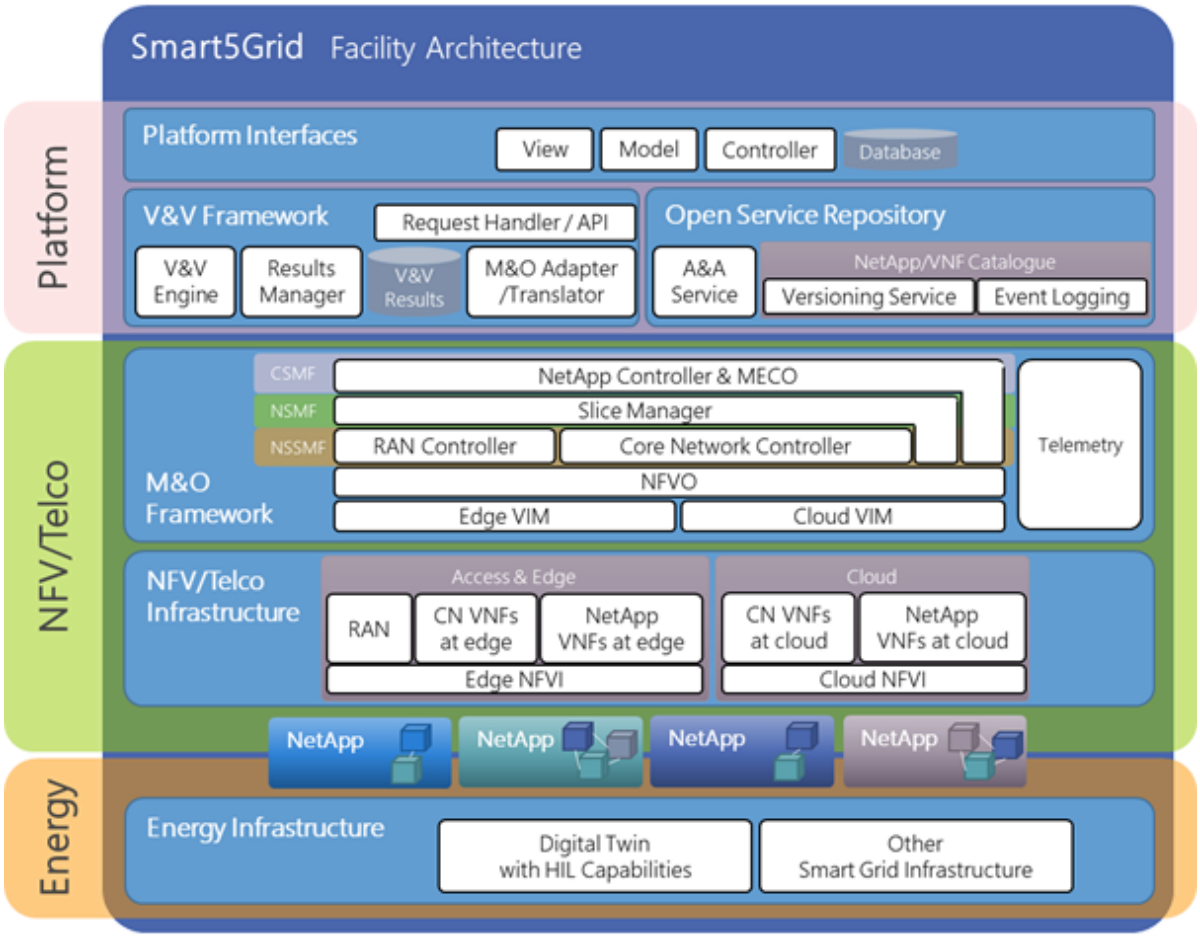


Figure 2-1 - Smart5Grid functional architecture.

One of the objectives of the Smart5Grid platform is to provide a common place for application developers and consumers within the market of energy grid applications. Towards this goal, Smart5Grid will develop a platform containing a repository of thoroughly tested NetApps, made available for consumers to use. This platform and the infrastructure upon which NetApps are deployed constitutes the overall Smart5Grid reference architecture, depicted in Figure 2-1, and described in detail in deliverable D2.1 & D2.2 [1] [2]. Such architecture is logically divided into three main layers:

1. The first layer, that we call the Smart5Grid platform, contains the Open Service Repository (OSR), the Verification and Validation (V&V) framework, and the platform interfaces from which users have access to them.
2. The layer containing the virtualization and telecommunications infrastructure, with its associated management and orchestration functions.
3. The energy infrastructure containing the grid components that connect to the NetApp services.

In the following, we will briefly recall the role and features of the main architectural components (for further details, please refer to D2.2). Furthermore, as a specific outcome of the work of T3.1, we are providing a definition of the interfaces that interconnect the components within the Smart5Grid platform, then NetApp Controller and MEC Orchestrator, the Slice Manager, and the other elements of the lower part of the M&O framework. Notice that throughout the whole document, the names of each interface within the Smart5Grid framework obey the following convention: *<interface_provider_acronym>_ <interface_consumer_acronym>*. For instance, the interface provided by the Slice Manager to the NetApp Controller is denoted SM_NAC.

2.1. Smart5Grid Platform

This section summarizes the roles and functionalities of the elements that constitute the Smart5Grid platform. Moreover, we are defining here the interfaces through which such elements interact with each other, and through which the V&V framework acts upon the M&O layer of the overall reference architecture.

2.1.1. User Interfaces

The User Interface (UI) provides the users the capability to intuitively perform operations on the NetApps and their subcomponents through the direct manipulation of graphical objects displayed on the web browser. The UI design principles follow the model–view–controller software pattern, which separates internal representations of information from the way information is presented to the user, resulting in a platform where users are shown which functions are possible rather than requiring the input of command in a terminal.

The main functionality of the UI comes from the underlying components of the OSR and the V&V platforms. In order to provide a unified experience, it adopts the same Authentication and Authorization (A&A) mechanism used in the OSR (see also Section Errore. L'origine riferimento non è stata trovata.), meaning that the users will only be able to view the resources they have access to according to the roles defined in the A&A service. The interfaces between the UI and the other Smart5Grid platform's components (the OSR and the V&V) can be seen in the Table 2-1. Notice that CLI stands for command line interface.

Table 2-1 – Interfaces and methods required by the UI.

Interface Name	Method Name	Description	Transport Information	Requires Interface From
OSR_UI-CLI	List NetApps	Get list of available NetApps	HTTPS	OSR
OSR_UI-CLI	Show NetApp Details	Show details of specified NetApp	HTTPS	OSR
OSR_UI-CLI	Update NetApp	Updated specified NetApp information	HTTPS	OSR
OSR_UI-CLI	Delete NetApp	Delete specified	HTTPS	OSR

		NetApp		
OSR_UI-CLI	Download NetApp	Download specified NetApp. The NetApp can be downloaded as code or as an image file	HTTPS	OSR
OSR_UI-CLI	Upload NetApp	Upload NetApp. The NetApp can be uploaded as code or as an image file	HTTPS	OSR
OSR_UI-CLI	List Virtual Network Functions (VNFs)	Get List of available VNFs	HTTPS	OSR
OSR_UI-CLI	Show VNF Details	Show details of specified VNF	HTTPS	OSR
OSR_UI-CLI	Update VNF	Updated specified VNF information	HTTPS	OSR
OSR_UI-CLI	Delete VNF	Delete specified VNF	HTTPS	OSR
OSR_UI-CLI	Download VNF	Download specified VNF. The VNF can be downloaded as code or as an image file	HTTPS	OSR
OSR_UI-CLI	Upload VNF	Upload VNF. The VNF can be uploaded as code or as an image file	HTTPS	OSR
OSR_UI-CLI	Show Event Logs	Show Event Logs concerning the specified resource	HTTPS	OSR
V&V_UI	Launch V&V Test	Initiate a test in the V&V platform concerning the	HTTPS	V&V

		specified NetApp		

2.1.2. Open Service Repository

The OSR service is the component of the Smart5Grid platform that offers the Application Programming Interfaces (APIs) allowing the interaction with the NetApps either in the form of packaged application or source code. Moreover, the OSR is responsible to preserve logs on the actions performed on the stored NetApps and VNFs. As it is elaborated in previous deliverables (D2.1, D2.2) the OSR is a collection of components interacting with each other to provide a unified service to its users. User access to the OSR resources is handled by the OSR's A&A Service in a role-based access control method. For the handling of NetApps and VNFs there is a component called NetApp/VNF Catalogue that contains two subcomponents: the OSR Code Versioning Service and the Event Logging component. As the name suggests, the former has the role of handling the NetApp/VNF code keeping track of different versions and providing packages of NetApps and VNFs. The Event Logging component keeps track of actions being performed on the NetApp/VNF code and packages. The following Table 2-2 list the interfaces provided and required by the OSR.

Table 2-2 – Interfaces and methods provided by the OSR.

Interface Name	Method Name	Description	Transport Information	Provides Interface To
OSR_UI-CLI	List Users	List registered users	HTTPS	UI, CLI
OSR_UI-CLI	Show User Details	Show details of specified user	HTTPS	UI, CLI
OSR_UI-CLI	Update User	Update specified user information	HTTPS	UI, CLI
OSR_UI-CLI	Delete User	Delete specified user	HTTPS	UI, CLI
OSR_UI-CLI	List NetApps	Get List of available NetApps	HTTPS	UI, CLI
OSR_UI-CLI OSR_V&V	Show NetApp Details	Show details of specified NetApp	HTTPS	UI, CLI, V&V
OSR_UI-CLI	Update NetApp	Update specified NetApp information	HTTPS	UI, CLI
OSR_UI-CLI	Delete NetApp	Delete specified NetApp	HTTPS	UI, CLI

OSR_UI-CLI	Download NetApp	Download specified NetApp. The NetApp can be downloaded as code or as an image file	HTTPS	UI, CLI
OSR_UI-CLI	Upload NetApp	Upload specified NetApp. The NetApp can be uploaded as code or as an image file	HTTPS	UI, CLI
OSR_UI-CLI	List VNFs	Get List of available VNFs	HTTPS	UI, CLI
OSR_UI-CLI	Show VNF Details	Show details of specified VNF	HTTPS	UI, CLI
OSR_UI-CLI	Update VNF	Update specified VNF information	HTTPS	UI, CLI
OSR_UI-CLI	Delete VNF	Delete specified VNF	HTTPS	UI, CLI
OSR_UI-CLI	Download VNF	Download specified VNF. The VNF can be downloaded as code or as an image file	HTTPS	UI, CLI
OSR_UI-CLI	Upload VNF	Upload VNF. The VNF can be uploaded as code or as an image file	HTTPS	UI, CLI
OSR_UI-CLI	Show Event Logs	Show Event Logs concerning the specified resource	HTTPS	UI, CLI
OSR_NAC	Download VNF Image data	Download image or Helm chart of the associated VNF	HTTPS	NAC

Table 2-3 – Interfaces and methods required by the OSR.

Interface Name	Method Name	Description	Transport Information	Requires Interface From
V&V_OSR	Validate NetApp	Initiate NetApp Validation process in the V&V platform	HTTPS	V&V
V&V_OSR	Verify NetApp	Initiate NetApp Verification process in the V&V platform	HTTPS	V&V
V&V_OSR	Show V&V test results	Show results of the specified test	HTTPS	V&V

External parties such as users or other Smart5Grid platform components can retrieve NetApps, access and manipulate their information, delete them, or create new ones. NetApps and VNFs will be stored in the OSR NetApp/VNF Catalogue. NetApp and VNF information must be defined.

The first part of information is the descriptor. The descriptor information is stored in text format, using a data representation format like JSON. It contains a description of all the components that constitute a NetApp or VNF and the way they are connected between them. In the case of the NetApp, it can contain VNFs or Physical Network Functions (PNFs) and relations between them. In the case of the VNF it can contain Virtual Deployment Units (VDUs) and its needed relations.

Having mentioned VDUs, we understand that the units to be deployed that interact to provide the service of the NetApp will be pieces of software applications. The second part of information is the VDU software application source code.

Such software applications will be deployed in virtual machines or containers. In order to facilitate deployment of the VDUs, the OSR NetApp/VNF Catalogue will offer an image registry service. Which leads us to the third part of information that is the VDU software application image.

The descriptor text files and NetApp source code, when publicly available, will be stored in the OSR Code Versioning service. Developers will be able to use the OSR Code Versioning service to collaborate on the development of NetApps and publish images to the image registry service. The image registry service will offer three types of commonly used image registries: a virtual machine image registry, a container image registry, and a Helm chart package registry. Source code and images of the included VNFs can also be hosted in the OSR and will be provided with similar APIs for interaction. The interfaces provided by the OSR can be mapped to the interfaces consumed by the UI. However, the OSR interfaces will be available both internally to the other Smart5Grid platform components and externally to the users for direct interaction. Also, a CLI tool will be provided for convenience.

The OSR extends its capabilities through the V&V_OSR interface. Through this integration NetApp Validation and Verification testing becomes possible from the OSR. The results of the V&V tests will be used to update the NetApp information to indicate successful or failed tests.

2.1.2.1. OSR Road Map

In this section (and similarly throughout the whole document), the n-th month of the project's lifetime is denoted Mn.

1) Define functionality and architecture (M4-M12)

The first months have been dedicated to the definition of the functionality provided by the OSR as well as the definition of the OSR service architecture. Relevant work is in deliverables D2.1, D2.2, and in this document.

2) Assess software options (open-source or custom) (M10-M14)

The OSR architecture is already defined, and the needed software components are determined. The next step is to carry out research on open-source software that can accommodate the functionalities that need to be provided by the OSR and compare such options. The main part of the service will need to be developed. The programming languages and frameworks that will be used need, also, to be defined. These activities will be reported in more detail in D3.3.

3) Development - Setup all OSR components (M10-M20)

Initial development activities have begun since M10. This allows to better understand the software needs and discover possible architecture design gaps early in development process. Once the software decisions are finalized, the main development phase can begin. Also, open-source projects that are planned to be included need to be set up and configured to meet the project needs.

4) Internal component integration (M10-M22)

The integration of the components takes place in parallel to development. The integration phase includes the software configuration and the development of additional interfaces to make the components inter-connectable.

5) Parametrization for external component integration (M16-M24)

The development of the OSR will take into account other Smart5Grid components that require communication with the OSR, such as the NetApp Controller and MEC Orchestrator. The required interfaces must be in place and generic enough to allow the parametrization needed for the integration.

6) Testing (M16-M24)

Testing of the OSR service will include tests that concern each OSR component separately and tests that verify OSR's functionality as a whole.

2.1.3. Verification and Validation Framework

The V&V Framework of the Smart5Grid platform enables intensive NetApp testing. The V&V Framework is supported by a telecommunications and virtualization infrastructure, the second layer of the Smart5Grid Platform, that hosts the deployment of NetApp instances for validation purposes; both the

infrastructure and the deployment of NetApps are controlled by a Management and Orchestration (M&O) framework, which includes the systems required to interpret, deploy, and monitor the functions described by the NetApp descriptor across its lifecycle.

The V&V platform interacts with several entities, namely the NetApp developers, the OSR, and the M&O framework. The developers may be able to request just the verification or both verification and validation of their NetApps to accelerate the development of the VNFs and services. The OSR must request the verification and validation of a NetApp before it is stored in its own repository. This request will lead to a verification field set as "false" while the verification step is not performed by the V&V Framework. To perform the validation process, the validation engine will request the onboarding and instantiation of NetApps, which will be terminated automatically once all the results and KPIs are gathered, to properly validate the NetApp. The interaction between the V&V platform and the other architectural components happens through the interfaces defined in Table 2-4 and Table 2-5.

Table 2-4 – Interfaces and methods provided by the V&V framework.

Interface Name	Method Name	Description	Transport Information	Provides Interface To
V&V_OSR	Validate NetApp	Validate a specific NetApp	HTTPS	OSR
V&V_OSR	Verify NetApp	Verify a specific NetApp	HTTPS	OSR
V&V_OSR	Give V&V test results	Provide results of the specified test	HTTPS	OSR
V&V_UI	Provide V&V test results	Provide results of the specified test	HTTPS	UI
V&V_M&O-Adapter	Provide a NetApp validation test	Interaction with M&O layer through the M&O Adapter	HTTPS	M&O

Table 2-5 – Interfaces and methods required by the V&V framework.

Interface Name	Method Name	Description	Transport Information	Requires Interface From
OSR_V&V	(Optional) Show NetApp Details	Show details of specified NetApp	HTTPS	OSR
VIM_V&V	Launch VIM operations	Allows operations between the V&V framework and	HTTPS	VIM

		the Virtualized Infrastructure Manager		
NFVI_V&V	Launch NFVI operations	Allows operations on NFVs between the V&V Framework and the NFVI Operations Manager	HTTPS	NFVI
NAC_V&V	NetApp onboarding	Allows operations on onboarded NetApps between the V&V Framework and the NAC	HTTPS	NAC

The V&V framework is composed of the verification component that addresses the composition of the NetApp Smart5Grid descriptors and provides the verification of the following elements: syntax verification, integrity verification, and topology verification and by the validation component that comprises a deployment and instantiation of a NetApp in an NFV M&O framework. In Smart5Grid it is expected that the V&V platform has multiple associated M&O frameworks to perform the NetApp validation tests which can be of multiple types as well, e.g., Open Source MANO (OSM)[2], Open Network Automation Platform (ONAP)[3], and others.

The NetApps and VNFs are stored in the OSR, they must undergo a testing process that provides guarantees on said NetApps to the consumers. This testing is realized by the V&V framework. The V&V framework provides the platform with a tool that enables automated testing of NetApps in two senses: 1) Verification, which ensures that the NetApp packages and all its components and files are well-formed, syntactically correct, and complete; and 2) Validation, which performs tests on live instances of NetApps guaranteeing that it can perform its function with the required performance levels. The validation phase of the NetApps is supported by the NFV/Telco layer and the V&V framework for validation is one scenario on which a NetApp deployment is considered.

Following the ETSI NFV Architectural Framework[4], also depicted in Section 0 the V&V Framework will be the VNF Management block and will interact with the NFV Orchestrator, the Virtualized Infrastructure Manager, the NFVI Operations Management, and the Working Domains. Hence, the V&V Framework will be responsible for managing all the relevant operations of each NetApp and communicating the results with other components.

[2] https://osm.etsi.org/
[3] https://www.onap.org/
[4] https://www.etsi.org/technologies/nfv

2.1.3.1. V&V Roadmap

1) Design and define the concept and architecture of a V&V framework for the Smart5Grid services (M1 – M15)

The first 15 months were dedicated to the design and definition of the concept and functional architecture of the V&V framework. This definition accords with the Smart5Grid specifications and overall requirements, allowing the correct answer to each use case's needs.

2) Development of the NFV automatic testing & validation framework through continuous integration (M13 – M19)

Once the architecture is defined, the development can start. This development follows the established requirements and needs for Smart5Grid. Also, this development should follow standards aligning the V&V Framework with other relevant projects and solutions.

3) Definition and development of the internal and external components of the V&V framework to integrate with Smart5Grid services (M18 – 26)

The integration of the components will take place in parallel to development and it includes the software configuration and development of additional interfaces to make the components inter-connectable, allowing a complete Smart5Grid Platform.

4) Test of the V&V experimentation framework for NetApp automatic testing, certification, and integration (M25 – M30)

The testing phase will virtualize entire classes of NetApps and also each of the components separately. To address the different needs of these tests, different Test Environments will be created guaranteeing that each NetApp will be tested and the correct workload between client and servers will be addressed.

In the rest of the section, we provide further details on the subcomponents that are part of the V&V framework: the V&V Engine, the Request Handler/API, the Results Manager, and the M&O Adapter/Translator.

2.1.3.2. V&V Engine

The V&V Engine is split into two sub-modules, the *Verification Engine* and the *Validation Engine* (cf. D2.2 [2]) and it is described in Figure 2-2. The *Verification Engine* will assess the syntax, the integrity and the topology of the NetApps and/or VNFs. The *Validation Engine* will address the Information Model Translation (*IMT* module), the onboarding and instantiation in a particular M&O framework and the Key Performance Indicator (KPI) retrieval and validation. The *IMT* submodule will be designed in a plugin-based approach to suit future needs of adding the support of more information models and thus assuring the growing compatibility of more M&O frameworks. The *NFV MANO Connection Handler* module will be designed in a similar, plugin-based, way to add more M&O frameworks in the future. The interaction between this module and the remaining components of the V&V Framework will be based on HTTP requests and responses, making use of functions and methods (1) HTTP GET for validation requests and/or information about services provided; (2) HTTP POST for sending and storing results.

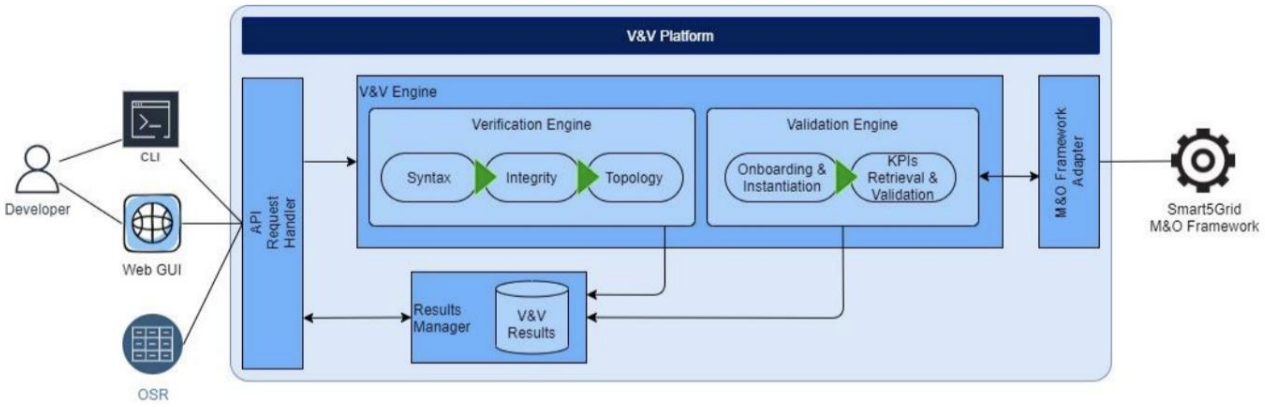


Figure 2-2 - V&V platform architecture.

2.1.3.2.1. VERIFICATION ENGINE

The verification component of the V&V platform will address the composition of the Smart5Grid NetApp descriptors. It will provide the verification on the following elements.

- Syntax verification: The NetApp descriptor and corresponding VNF descriptors (VNFDs) are syntactically validated against the scheme templates specified by data model of Smart5Grid.
- Integrity verification: The validation of integrity verifies the overall structure of descriptors through the inspection of references and identifiers both within and outside the individual descriptors. As NetApp and VNFDs have different information scopes, the validation goals of this type of activity either in the context of a VNF or NetApp validation are provided as follows:
    - NetApp Integrity: NetApp descriptors typically contain references to multiple VNFs, which are identified by the ID of the VNF. The integrity validation ensures that the references are valid by checking the existence of the targeted VNFs. Integrity validation also verifies the connection points (CPs) of the NetApp. This comprises the virtual interfaces of the NetApp itself and the interfaces linked to the referenced VNFs. All CPs referenced in the virtual links of the NetApp must be defined, whether in the NetApp descriptor or in its VNFD.
    - VNF Integrity: Similarly, VNFs may also contain multiple subcomponents, usually referred to as the Virtual Deployment Units (VDUs) or Virtual Network Function Components (VNFCs). As a result, the integrity validation of a VNF follows a similar procedure of a NetApp integrity validation, with the difference of VDUs being defined inside the VNFD itself. Again, all the CPs used in virtual links must exist and must belong to the VNF or its VDUs.
- Topology verification: This tool provides a set of mechanisms to validate the network connectivity graph to aid the development of connectivity logic. Typically, a NetApp contains several inter-connected VNFs and each VNF may also contain several inter-connected VDUs. The connection topology between VNFs and VDUs (within VNFs) must be analysed to ensure a correct connectivity topology. To present this rationale, Figure 2-3 depicts a NetApp example used to better illustrate validation issues. This figure is followed by a summary of the set of issues that this software can detect:

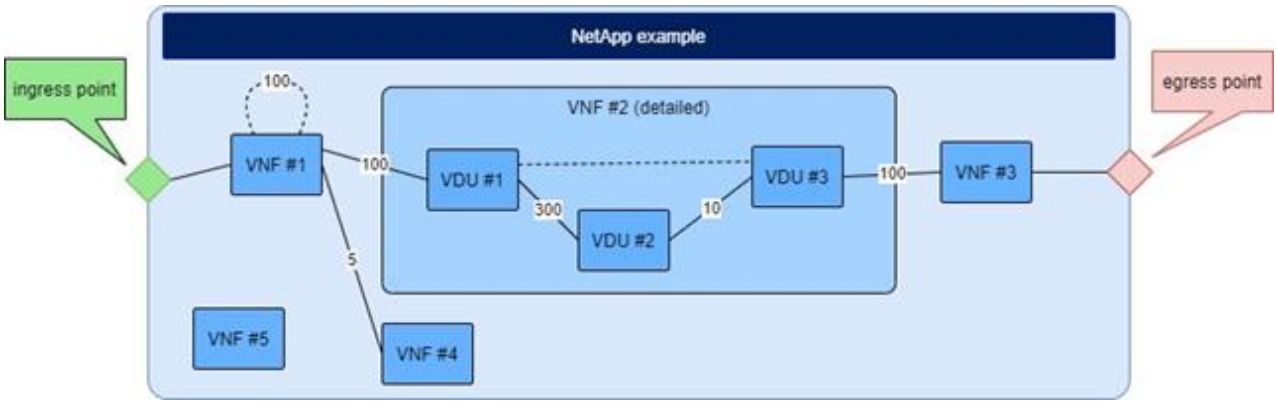


Figure 2-3 - NetApp components example.

- Unlinked VNFs, VDUs and CPs: Unconnected VNFs, VDUs and unreferenced CPs will trigger alerts to inform the developer of an incomplete service definition. In this case, VNF #5 would trigger a message to inform that it is not being used;
- Network loops/cycles: The existence of cycles in the network graph of the service may not be intentional, particularly in the case of self-loops. For instance, VNF #1 contains a self-linking loop, which was probably not intended. Another example is the connection between VDU #1 and VDU #3 which may not be deliberate. This tool analyses the network graph and returns a list of existing cycles to help the developer in the topology design. In this example, it would return the cycles:
  - [VNF #1, VNF #1],
  - [VDU #1, VDU #2, VDU #3, VDU #1];
- Node bottlenecks: Warnings about possible network congestions, associated with nodes, are provided. Considering the bandwidth specified for the interfaces, weights are assigned to the edges of the network graph in order to assess possible bottlenecks in the path. As specified in the example, the inter-connection between VDU #2 and VDU #3 represents a significant bandwidth loss when compared with the remaining links along the path.

2.1.3.2.2. VALIDATION ENGINE

The validation process comprises the deployment and instantiation of the to be validated NetApp in the M&O framework. In Smart5Grid, it is expected the V&V Platform to have the associated M&O framework to perform the NetApp validation tests which can be of multiple types as well (e.g.: OSM, ONAP, and others).

- NetApp onboarding: Once the translation process is finished, the next logical step is to onboard the NetApp in the corresponding M&O framework.
- NetApp instantiation: Here is where the Validation engine will request the instantiation of the NetApp.
- Retrieve KPIs: To properly validate the NetApp, the Validation engine will ensure that certain specified KPIs are met. Hence, the M&O framework will have a mechanism to measure and

provide such metrics (more detailed in Section 4.4.2.6 of D2.2). The KPIs to be assessed are specified by the developers in the Smart5Grid NetApp descriptors. The end-to-end latency of a NetApp, the latency between two VNFs or the bandwidth between two CPs are examples of KPIs that will be supported. The retrieved KPIs are then stored in the V&V results database for later analysis.

- Validation results analysis: This process verifies that the retrieved KPIs are within the thresholds specified by the developer. It is in this process that is determined if a NetApp is successfully validated.

2.1.3.2.3. TEST STRATEGY

Network Functions Virtualization (NFV) is a network architecture concept that proposes using IT virtualization-related technologies to virtualize entire classes of network node functions into building blocks that may be connected, or chained, together to create communication services. Therefore, a specific strategy should be defined in order to define how to test the Network Functions.

Following the ETSI ISG NFV Standards[5], the strategy to be applied will follow Testing and QoE (Quality of Experience) as the main use cases to address. To perform a compliance test, each component is analysed independently (M&O, V&V Framework, VIM, NFVI, and other interfaces). Besides analysing the different components separately, an NFV Test Environment should be built in parallel in order to address different needs.

2.1.3.2.4. TEST SCHEDULE

After building an NFV Test Environment, different Test Appliances should be defined to simulate valid workload traffic on the client and server, simulate both data plane and control plane traffic, and measure key metrics. One of the things that should be considered during the Test schedule is choosing the correct Test Appliance: Virtual or Physical.

Physical test appliances may be used, but most test equipment vendors have developed virtual test appliances that work on separate Virtual Machines in the NFV Test Environment. Virtual test appliances offer equivalent capabilities for almost all scenarios and meet the flexibility required when testing NFV on multiple, geographically disparate. On the other hand, Physical test appliances are mainly recommended for testing virtual environments that require the highest levels of data-plane performance (line-rate) or microsecond-level timing accuracy.

Finally, several metrics can be retrieved: (1) NFV Service Metrics such as the VM Provisioning Latency, instantiation latency, and time between VM instantiation and first available packet; (2) QoS and Data-plane metrics such as the latency on each of tens of thousands of data streams and throughput and forwarding rate; and (3) QoE and Control-plane metrics.

2.1.3.3. Request Handler/API

The V&V Framework will require a handler to indicate that a specific NetApp should be validated or/and verified, this "starting target" will be performed by the Request Handler that will be responsible to provide the necessary information about the NetApps and the specification needed by the V&V Framework.

[5] https://www.etsi.org/technologies/nfv

With this in mind, the Request Handler handles each one of the validation requests and constructs a response. This module contains servlet base classes to define the request handler functions and a derived servlet to implement the function or functions that handle the requests. Each one of these is based on HTTP protocol and contains default implementations of the handlers for standard HTTP requests, implementing the functions to accept or reject requests. Finally, a servlet takes an HTTP Request and HTTP Response as parameters. To respond to a request, a servlet overrides the functions that correspond to the request methods the servlet accepts. For example, an HTTP servlet that accepts the HTTP GET method implements the *doGet()* function.

2.1.3.4. Results Manager

After validating and verifying all NetApps, ensuring their integrity and typology, the V&V Framework is responsible for storing all results in order to maintain a validation history. This Result Manager will make it easy to manage and interpret future NetApps, according to the created history. Having this information will provide faster results, with less risk of error and in fewer manual steps, thus minimizing the overall hands-on time. Moreover, it will permit a higher flexibility to define validation and verification rules.

Connected with the V&V Engine and the Request Handler, the Results Manager will receive HTTP GET methods when a NetApp needs to be Validated or Verified creating a new entry into the database where the result is obtained from the validation and verification will be placed later, through HTTP POST methods. These methods are used to send data to a server to create/update a resource.

2.1.3.5. M&O Adapter/Translator

In order to translate the results obtained from the validation and verification of NetApps, the V&V framework is responsible for sending these results to the M&O Framework through an adapter. This allows a succinct and automatic translation of results and validated NetApps, guaranteeing that users will have access to trustworthy services and with the guarantee of traffic quality they expect.

This will be performed by HTTP Requests each time a user needs to use a service and a NetApp should be validated. Every time the V&V Framework finishes the validation and verification, an HTTP Response is generated, and an HTTP POST method is used to give the result and provide the necessary information about the service to the user. Besides, it will be also possible for the users to request information about the available service in each NetApp as HTTP GET methods.

2.2. NetApp Controller & MEC Orchestrator

As described in Section 4.4.2 of [2] the NetApp Controller & MEC Orchestrator (MECO) has the role to manage the NetApps and MEC servers' lifecycles. This includes NetApp-specific functions but also resource management functions.

NetApp-specific functionalities include NetApp onboarding and its NetApp lifecycle management; that is: deployment, placement, monitoring, re-deployment and deletion. Instead, resource management functions include node management (onboarding and provisioning) and network slice management.

While some of these functionalities can be carried out by the NetApp Controller & MECO itself, others need to interact with external components that are part of the Smart5Grid architecture. For instance, the

NetApp Controller & MECO needs to interface with the Telco Slice Manager (SM. Similarly, when onboarding a new node (or cluster) the NetApp Controller & MECO needs to start coordinating with the NFVO present on the node; NFVO that can be different from node to node and from site to site.

To prevent vendor lock-in, Smart5Grid proposes a reference architecture that can be implemented using different components that cover the same role. To support this flexibility offered by the reference architecture, a NetApp Controller & MECO can extend its support for a specific component by adding an adapter for it. Figure 2-4 shows the internal architecture of the NetApp Controller and MECO including its main components and the adapters.

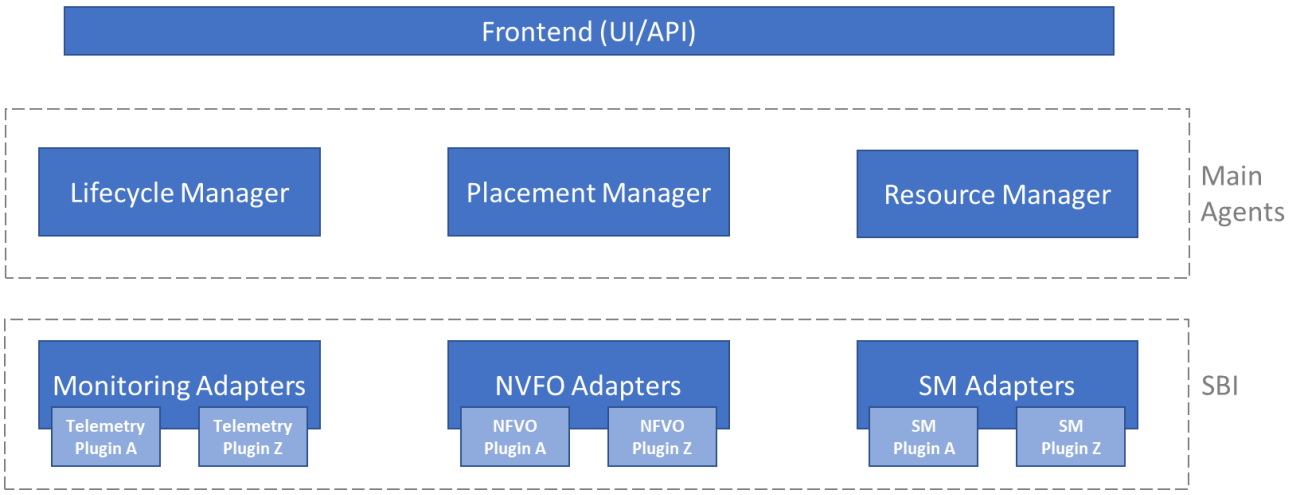


Figure 2-4 – NetApp Controller Components.

The following tables, instead, provide a list of the main interfaces between the NetApp Controller & MECO and the other architectural components.

Table 2-6 – Interfaces and methods provided by the NetApp Controller & MECO.

Interface Name	Method Name	Description	Transport Information	Provides Interface To
NAC_FE	NetApp Onboarding	Create, read, update, and delete onboarded NetApps	HTTPS	Frontend (Mainly to the V&V Framework)
NAC_FE	NetApp Deployment	Create, read, update, and delete NetApp deployments	HTTPS	Frontend (V&V Framework and API)

Table 2-7 – Interfaces and methods required by the NetApp Controller & MECO.

Interface Name	Name	Description	Transport Information	Requires Interface from
OSR_NAC	Download resource from registry	Download helm charts and docker images from OSR	HTTPS via helm and docker pull[6]	OSR
AF_NAC	Set traffic rules	Create, read, update, and delete traffic rules to Core Network	5G Naf[7]	5G Core
SM_NAC	Set slices	Create, read, update, and delete slices (core, network, and resources)	HTTPS	Slice Manager
NFVO_NAC	NVF deployment	Create, read, update, and delete VNFs	HTTPS	NFVO
VIM_NAC	Node registration and provisioning	Register and provision nodes	HTTP(S)	VIMs
Nodes_NAC	Node metrics scraping	Collect nodes metrics and liveness	HTTP(S)	VIMs
NetApps_NAC	NetApp metrics scraping	Collect NetApps metrics and liveness	HTTP(S)	NetApp instances

2.2.1. NetApp Controller and MEC Orchestrator Roadmap

1) Define functionality and architecture (M4-M12)

We dedicated the first months to the definition of the functionalities provided by the NetApp Controller & MECO as well as the definition of the OSR service architecture. Relevant work was provided in [1][2]

2) Assess software options (open-source or custom) (M10-M14)

Next, we defined a modular and flexible architecture. This flexible solution based on plugins was defined for two main reasons: to prevent vendor lock-in and to fit the needs of all UCs. Being a central

[6] https://docs.docker.com/engine/reference/commandline/pull
[7] https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf

component of the architecture that touches many other components, we a flexible architecture based on plugins was needed.

The partners have identified two candidates: one, based on NearbyOne, NearbyComputing's proprietary solution. A second, based on Neutroon's proprietary solution which performs the role of a 5G Private Network provider. This solution is provided by i2CAT.

3) Development and integration (M10-M26)

Partners started the development of the solutions they provide. The integration phase includes development of the plugins to make the component inter-connectable with the others Smart5Grid components.

4) Field setup and testing (M16-M24)

During this phase the partners will obtain access to the edge sites to install their solution on the field and start testing. Development and testing are taking place in parallel to favour fast prototyping and agile development.

2.3. Slice Manager

The SM within the Smart5Grid architecture is the component in charge of managing the lifecycle of network slices. The main function is to create logical partitions consisting of several network segments (e.g., radio, transport, clouds) by reserving isolated computational, network, core, and radio resources to be assigned to a particular service or customer requiring an end-to-end communication service.

That logical partitioning of the resources assigned to each network slice ensures that the service requirements, as well as KPIs defined in the NetApp's descriptors are met. For this purpose, SM uses two interfaces that allow it to interact with the rest of the platform components. For example, through the northbound interface (NBI), it communicates with the NetApp Controller & MECO. SM via its NBI, on the one hand, exposes the information related to the users, infrastructure/network resources, slices instantiated in terms of chunks, and network services deployed on each slice. And on the other hand, SM can receive requests from the NetApp Controller to execute CRUD (create, read, update, delete) operations on the network slices deployed over the underlying network infrastructure. In this light, the northbound API implements several REST methods that allow SM to manage the NetApp Controller requests and the lifecycle of network slices.

Furthermore, SM implements a southbound interface (SBI) to apply the NetApp Controller requests in terms of network resources. Through its SBI, SM interacts with several components, such as RAN Controller, 5G Core Network (CN) Controller, NFVO, and VIMs. In essence, SM uses this interface to validate, reserve, configure, control, and orchestrate the resources associated with a network slice. The main functionalities of the SM are described in [2]. The management of RAN resources and the core are two functionalities that are directly related to the slice creation and management. We argue that these functionalities might not need dedicated software modules to be implemented, but that the Slice Manager could be integrate the required functionalities of the RAN Controller and 5G CN Controller, a design choice that depends on each particular UC and which will be evaluated separately. In the following, for a better understanding of the responsibilities of these two modules and why they are so tightly bound to the Slice Manager, we review their functional roles in the Smart5Grid architecture.

The RAN Controller is a module that in the Smart5Grid architecture positions itself below the Slice Manager and CN Controller and which communicates with the radio access network devices. The main task of the RAN Controller is to provide an abstraction to the upper layers of the architecture to manage and configure the radio devices that generally speaking rely on specific configuration protocols, such as SNMP, NETCONF or TR-069. The RAN Controller is able to register such devices and their status, as well as to expose the RAN infrastructure and offer control over it to other managing architecture elements, such as the Slice Manager or the 5G CN Controller.

In Smart5Grid, the RAN Controller may be used for day-0 configuration of the radio devices, as an alternative to their manual configuration, and later on for the reservation of radio resources for slices end-to-end infrastructure slices, as well as the instantiation of radio services to which users of the service implemented in a NetApp can connect. Depending on the particular HW and SW deployments of a UC, the RAN Controller functions might be implemented by the Slice Manager or in case of the optional day-0 configuration, they might be implemented by an external component, such as a mobile network operator's management system. A 5G core, as part of a private 5G network deployment, has to be managed as well, if radio connectivity is to be provided for a service. The 5G CN Controller component is responsible for managing the 5G core configurations. In the Smart5Grid architecture, it is located between the 5G core and Slice Manager within the NFV/Telco layer. Mainly, 5G CN Controller implements some M&O functionalities such as NSSMF defined by 3GPP, for example, resource reservation, handle the registration and status of control and plane functions of 5G core, traffic configuration and accessibility policies. This allows it to have an active interaction with the NSMF component, which in the present project, is represented by Slice Manager.

In this light, 5G CN Controller can be implemented as a stand-alone component or integrated within Slice Manager. In the present project 5G CN Controller is designed as a subcomponent that is part of SM. Therefore, the creation and configuration functionalities of the 5G core can be performed as day-0 configurations from SM.

In short, the main interfaces implemented in SM are based on the relationship between the different components of the Smart5Grid platform, as shown in Figure 2-1. The interfaces described above reflect the main communication methods that SM establishes with the rest of the platform components to execute different operations.

It is useful to notice that in network deployments that do not include a SM, dynamic network slicing can still be jointly carried out by the NetApp Controller & MECO, the RAN Controller, and the 5G CN controller, which will separately offer the slicing for their resources. Static network slicing, instead, can be set up manually at the moment of network configuration.

The main methods of each interface are listed below in the following tables.

Table 2-8 – Interfaces and methods provided by the slice manager.

Interface Name	Method Name	Description	Transport Information	Provides Interface To
SM_NAC	Register Computing Resource	Registration of the computing resources with specific quotas	HTTP	NetApp Controller & MECO
SM_NAC	List Computing Resources	Shows the computing resources registered	HTTP	NetApp Controller & MECO
SM_NAC	Delete Computing Resources	Deletes specific computing resources	HTTP	NetApp Controller & MECO
SM_NAC	Register Network Resources	Registration of the cloud networks resources	HTTP	NetApp Controller & MECO
SM_NAC	List Network Resources	Shows cloud networks information	HTTP	NetApp Controller & MECO
SM_NAC	Delete Network Resources	Deletes specified cloud network resources	HTTP	NetApp Controller & MECO
SM_NAC	(Optional) Register RAN Resources	Registers radio infrastructure and creates new cells	HTTP	NetApp Controller & MECO
SM_NAC	(Optional) List RAN Resources	Shows the RAN devices information	HTTP	NetApp Controller & MECO
SM_NAC	(Optional) Delete RAN Resources	Deletes specific RAN devices and resources	HTTP	NetApp Controller & MECO
SM_NAC	NetSlice Creation	Creates Network Slices	HTTP	NetApp Controller & MECO
SM_NAC	List Network Slices	Shows information of the created	HTTP	NetApp Controller

		network slices		& MECO
SM_NAC	Delete Network Slices	Delete specific network slices	HTTP	NetApp Controller & MECO
SM_NAC	(Optional) Add 5GCN Resource and Radio Service	Creates a new 5GCN and radio service into the specific network slice	HTTP	NetApp Controller & MECO
SM_NAC	(Optional) List 5GCN and Radio Service	Lists the 5GCN and radio service information	HTTP	NetApp Controller & MECO
SM_NAC	(Optional) Delete 5GCN and Radio Service	Deletes specific 5GCN and radio services	HTTP	NetApp Controller & MECO
SM_NAC	(Optional) NetApp deployment	NetApps instantiation over a specific network slice	HTTP	NetApp Controller & MECO
SM_NAC	(Optional) List NetApps	Provides NetApp instances information	HTTP	NetApp Controller & MECO
SM_NAC	(Optional) Delete NetApps	Deletes a specific NetApp	HTTP	NetApp Controller & MECO

Table 2-9 – Interfaces and methods required by the slice manager.

Interface name	Method name	Description	Transport information	Requires interface from
NFVO_SM	(Optional) NetApp Deployment (NFVO)	Requests CRUD operations for the NetApps	HTTP	NFVO
VIM_SM	Register Computing Resources (VIM)	Requests the creation of quotas over the VIM	HTTP	VIM

RAN-Controller_SM	(Optional) Register Radio Resources (RAN-Controller)	Requests provision/configuration of RAN devices	HTTP	RAN Controller
5GCN-Controller_SM	(Optional) Provide 5GC Resources (5GCN-Controller)	Creates/Applies the forwarding traffic rules	HTTP	5G CN Controller

2.4. NFV central and edge infrastructure

2.4.1. NFV architecture

The NFV layer's main objective is to separate, through virtualisation techniques, the software functions (VNFs) from the dedicated hardware they run on, allowing them to operate on Commercial-Off-The-Shelf (COTS) equipment, thus improving costs. This separation improves deployment flexibility, scalability and, in general, the agility of operational procedures, resulting in fast innovation.

The NFV framework in Figure 2-5 focuses on providing the necessary methods and equipping VNFs with the capabilities required to be dynamically managed and orchestrated across the infrastructure. To achieve this, the NFV framework can be decomposed in three main components:

- VNFs: Virtualized functions that are enabled by the NFV framework.
- NFV Infrastructure: Infrastructure where VNFs run (physical resources).
- NFV Management and Orchestration: Provides the necessary functionality to manage the lifecycle of both the VNFs and the resources they run on through virtualization.

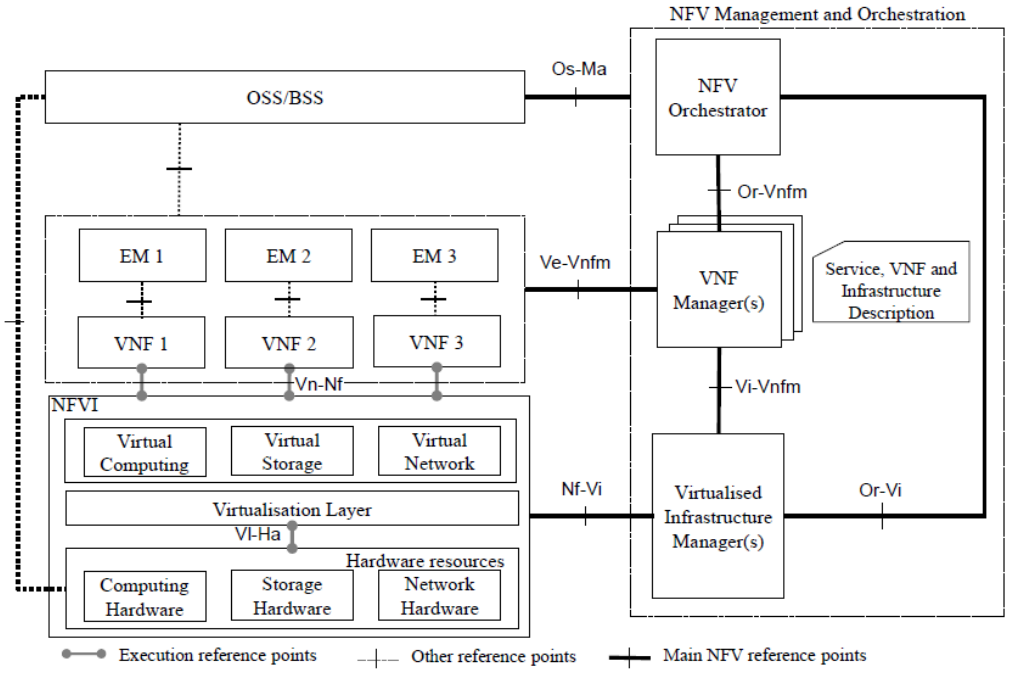


Figure 2-5 - NFV reference architectural framework [3].

The NFV MANO framework is subsequently divided in three components in ETSI reference architecture: the NFVO, VNF Manager (VNFM), and Virtualized Infrastructure Manager (VIM). Specific implementations, including the ones planned in Smart5Grid UCs, may take varying approaches, or combine several of these functions into a single component, e.g., an NFVO may also perform the VNFM functions.

There are 6 reference points relevant to the NFV MANO as specified by ETSI[3]. These should serve as a reference and different implementations of NFV MANO components may take varying approaches. These reference points are:

- OSS/BSS - NFV Management and Orchestration (Os-Ma)
- NFV Orchestrator - Virtualised Infrastructure Manager (Or-Vi)
- NFV Orchestrator - VNF Manager (Or-Vnfm)
- Virtualised Infrastructure Manager - VNF Manager (Vi-Vnfm)
- NFVI - Virtualised Infrastructure Manager (Nf-Vi)
- VNF/EM - VNF Manager (Ve-Vnfm)

In this section we highlight in Table 2-10 and Table 2-11 for reference, the NFV MANO northbound interface (from [3]) provided by the NFVO to an OSS/BSS, which will be consumed by the NetApp Controller, corresponding to Os-Ma reference point, and the interfaces provided by the VIM, corresponding to Or-Vi and Vi-Vnfm.

Table 2-10 – Interfaces and methods provided by the NFVO.

Interface Name	Method Name	Description	Transport Information	Provides Interface To
NFVO_ OSS/BSS	NSD Management	Management operations on NSDs (such as CRUD). Associated to Os-Ma reference point.	HTTP	NetApp Controller & MECO, Slice Manager
NFVO_ OSS/BSS	NS Lifecycle Management	Enables the OSS/BSS to perform NS Lifecycle Management operations. Associated to Os-Ma reference	HTTP	NetApp Controller & MECO, Slice Manager

		point.		
NFVO_ OSS/BSS	NS Performance Management	Allows for the management of performance monitoring jobs and collection/reporting of metrics related to the NS. Associated to Os-Ma reference point.	HTTP	NetApp Controller & MECO, Slice Manager
NFVO_ OSS/BSS	NS Fault Management	Management and reporting of alarms on NS faults. Associated to Os-Ma reference point.	HTTP	NetApp Controller & MECO, Slice Manager
NFVO_ OSS/BSS	VNF Package Management	Management of VNF packages (CRUD operations). Associated to Os-Ma reference point.	HTTP	NetApp Controller & MECO, Slice Manager
NFVO_ OSS/BSS	NFVI Capacity Information	Provides information on NFVI Capacity to the OSS/BSS. Associated to Os-Ma reference point.	HTTP	NetApp Controller & MECO, Slice Manager
NFVO_ OSS/BSS	VNF Snapshot Package Management	Management of VNF Snapshot Packages (CRUD operations). Associated to Os-Ma reference point.	HTTP	NetApp Controller & MECO, Slice Manager
NFVO_ OSS/BSS	Policy	Policiy Magement Operations (CRUD,	HTTP	NetApp Controller &

	Management	Activation, Deactication, Transfer). Associated to Os-Ma reference point.		MECO, Slice Manager
NFVO_VNFM	VNF Package Management	Used to obtain VNF Package information from the NFVO	HTTP	VNFM
NFVO_VNFM	VNF Lifecycle Operation Granting	Allows the VNFM to request NFVO to authorize a lyfecycle operation	HTTP	VNFM
NFVO_VNFM	Virtualized Resources Management	Receive Virtualized hardware resource reservation and/or allocation requests by the VNFM.	HTTP	VNFM

Table 2-11 – Interfaces and methods required by the NFVO.

Interface name	Method name	Description	Transport information	Requires interface from
NFVO_VNFM	VNF Lifecycle Management	Allos the NFVO to trigger lifecycle operations on VNFs (such as create, instantiate, scale, heal, terminate)	HTTP	VNFM
NFVO_VNFM	VNF Performance Management	Enables the NFVO to obtain VNF performance information	HTTP	VNFM
NFVO_VNFM	VNF Fault Management	Allows the NFVO to obtain alarms related to VNFs	HTTP	VNFM

NFVO_VNFM	VNF Indicator	Through this interface, the NFVO receives updates related to VNF Indicators	HTTP	VNFM
NFVO_VNFM	Policy Management	Allows the NFVO to manage (such as transfer, delete, query, activate, deactivate) policies related to the VNF lifecycle	HTTP	VNFM
NFVO_VIM	Virtualized hardware resource Management	Virtualized hardware resource reservation and/or allocation requests by the NFV Orchestrator or VNFM.	HTTP	VIM
NFVO_VIM	SW Image management	The NFVO uses this interface to query the VIM for software images.	HTTP	VIM

2.4.2. Cloud infrastructure for the deployment of the Smart5Grid platform

The Smart5Grid platform consists of numerous software components. Each one requires a certain amount of computational, network, and storage resources. The technology chosen to deploy these components is single application Linux containers since they simplify the developers' workflow and the process of solution delivery to the environment. As the container orchestrator, we have chosen Kubernetes[8] to understand requests such as deploying and scaling multiple instances of containers, operating multiple services, and scaling them up and down. For data persistence the containers use volumes provided by a storage back-end system. To ensure high availability of the services deployed on the containers the storage volume requires to support concurrent access from containers residing on separate hosts. A network storage solution exposing NFS shares to Kubernetes is suitable for such a purpose. Also, most of the Smart5Grid platform components require a relational database. The database chosen is PostgreSQL[9], allowing the storage of large and sophisticated data safely. It will help building complex applications, run administrative tasks, and create integral environments

[8] https://kubernetes.io/

[9] https://www.postgresql.org/

At the time of writing, we have deployed the core elements of the Smart5Grid platform in a laboratory environment. We, also, have experimented with and tested other software components to assess whether or not they cover the project's needs and requirements. The testing environment consists of three general purpose hardware nodes for computing resources and a storage appliance.

The nodes are deployed as follows:

The storage appliance is a QNAP server providing storage NFS storage for the Kubernetes volumes.

One hardware node is set up as a KVM host to support running Virtual Machines (VMs).

a. A VM is a single Kubernetes (k3s implementation) master node.
b. PostgreSQL database for k3s.
c. PostgreSQL for services running on k3s.
d. The other VMs are supporting services (GitLab, ArgoCD[10], Bind DNS server[11], HAproxy[12]).

The other two hardware nodes are set up as Kubernetes (k3s implementation) worker nodes.

Details of the hardware used in the laboratory environment can be seen in Table 2-12.

Table 2-12 - Hardware employed for the lab testing of the Smart5Grid platform.

Role	Type/Model	CPU	RAM	Disks	Network Interfaces
NFS Storage	QNAP TS-451D2	2x2GHz	16GB	4x3TB HDD (RAID6)	2x1Gbps
KVM host	General Purpose PC	8x3.2GHz	32GB	2X512GB SSD (RAID1)	2x1Gbps
Compute Node 1	General Purpose PC	8x3.2GHz	32GB	2X512GB SSD (RAID1)	2x1Gbps
Compute Node 2	General Purpose PC	8x3.2GHz	32GB	2X512GB SSD (RAID1)	2x1Gbps

This set up should be considered adequate for testing purposes. The production environment shall accommodate a cluster of three nodes for the PostgreSQL database destined for the services running on the Kubernetes platform. For the Kubernetes platform another implementation can be considered (e.g., kubeadm, kubespray) and the PostgreSQL might be replaced by an etcd cluster. Better response times could be offered if we could use SSDs instead of HDDs in the NFS storage, also, NFS drives could be of

[10] https://argoproj.github.io/cd/
[11] https://www.isc.org/bind/
[12] http://www.haproxy.org/

lower storage capacity. Finally, in terms of network, in order make use of the full speed of NFS SSD drives, 10 Gigabit Ethernet interfaces could be used instead of the ones of 1 Gigabit used in the test environment. All the nodes in the production environment could be and would be preferable to be Virtual Machines instead of hardware nodes for easier of management.

2.4.3. Virtualization infrastructure in UC1

For UC1 there will be an edge computing node (NFVI) where NetApps will run as containers (CNFs). The edge node will be hosted in Wind3 data centre and it will be managed and orchestrated by NearbyOne NetApp Controller & MECO. NearbyOne offers multi-site NFVO and NVFI functionalities like node provisioning, node and NetApp monitoring, traffic steering, etc.

2.4.4. Virtualization infrastructure in UC2

For UC2 there will be an edge computing node (NFVI) which contains Openstack as Virtual Infrastructure Manager (VIM). Inside of Openstack a Kubernetes cluster will be launched where NetApps will run as containers (CNFs). This infrastructure will be managed and orchestrated by the Neutroon solution, which implements several of the components of the M&O framework.

2.4.5. Virtualization infrastructure in UC3

The support of the telecommunication network as well as for the edge node will be addressed with both Kubernetes for container and Openstack for virtual machines in a hybrid cloud environment.

2.4.6. Virtualization infrastructure in UC4

For UC4 there will be two edge computing nodes (NFVI) grouped in 2 sites (Kubernetes clusters) where NetApps will run as containers (CNFs). This infrastructure will be managed and orchestrated by one instance NearbyOne NetApp Controller and Orchestrator. NearbyOne offers multi-site NFVO and NVFI functionalities like node provisioning, node and NetApp monitoring, traffic steering, et cetera. While the edge nodes will be kernel-based VMs hosted in Vivacom's datacentre, the NearbyOne instance is going to be hosted in a public cloud (e.g., Amazon Web Services).

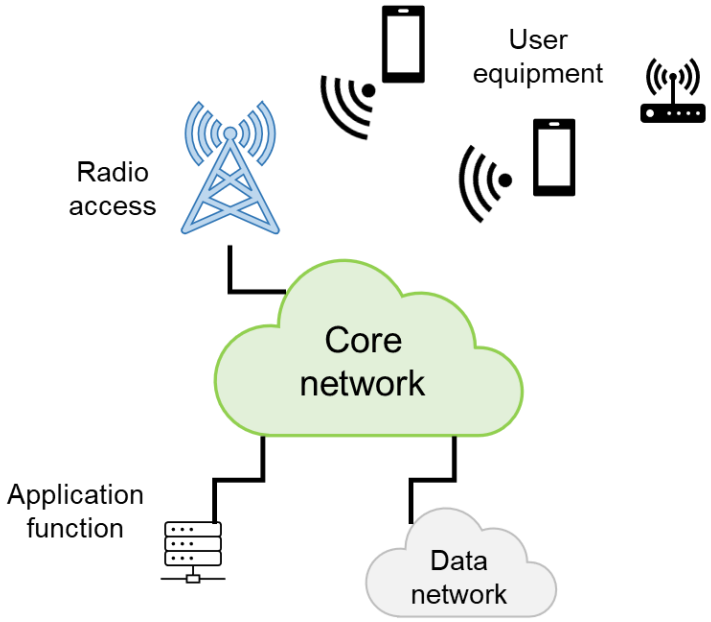


Figure 2-6 - High-level representation of a 5G system.

2.5. 5G infrastructure

A 5G system (5GS) provides data connectivity service to transfer packets over a radio channel between the user equipment (UE) and data networks (DNs) or application functions (AFs), as represented in Figure 2-6.

In Smart5Grid's use cases, 5G networks are going to be deployed. Each deployment has a dedicated core network and features local breakout towards services hosted on edge computing nodes. In the following, we are recalling the main functionalities that compose the core network and the radio access network.

2.5.1. Core network

In a 5G system, the 5G core (5GC) network comprises a wide and quite complex set of functionalities, which belong to two main categories: the so-called *user plane* and *control plane* functionalities. The former handle the connection between the RAN – and therefore the UEs – with external data networks (the Internet, a cloud computing facility, an application server, a LAN, etc.) via the User Plan Function (UPF); the latter, instead, consist in overseeing all the network's processes and functionalities that go beyond the radio access, e.g., subscriber data management and user authentication, policy control, connectivity and mobility management, or exposure of services towards application functions. Such functionalities are defined and standardized by 3GPP, and the main network functions (NFs) that compose a 5GC are the following:

- The Access and Mobility Management Function (AMF), which is responsible for handling connection and mobility management tasks. The AMF can be accessed by any number of gNBs in the RAN.
- The Session Management Function (SMF), primarily responsible for interacting with the decoupled data plane, handling Protocol Data Unit (PDU) sessions and managing session context with the UPF.
- The UPF, responsible for the processing and forwarding of PDUs between the UEs and the data networks.
- The Unified Data Management (UDM), which is a centralized way to control network user data.
- The Authentication Server Function (AUSF), in charge of performing the authentication function.
- The Unified Data Repository (UDR), which is a converged repository of subscriber information.
- The Policy Control Function (PCF), responsible for the determination and communication of operator policy to the control plane NFs.
- The Network Repository Function (NRF), which maintains an updated repository of all the 5G elements available in the operator's network along with the services provided by each of the elements in the 5GC.
- The Network Slice Selection Function (NSSF), which has the role to assist the AMF with the selection of the Network Slice instances that will serve a particular device.

- The Network Exposure Function (NEF), that provides capability and events exposure, and makes 5GC functionalities and network analytics available to third parties (service providers, verticals) via Application Functions (AFs).

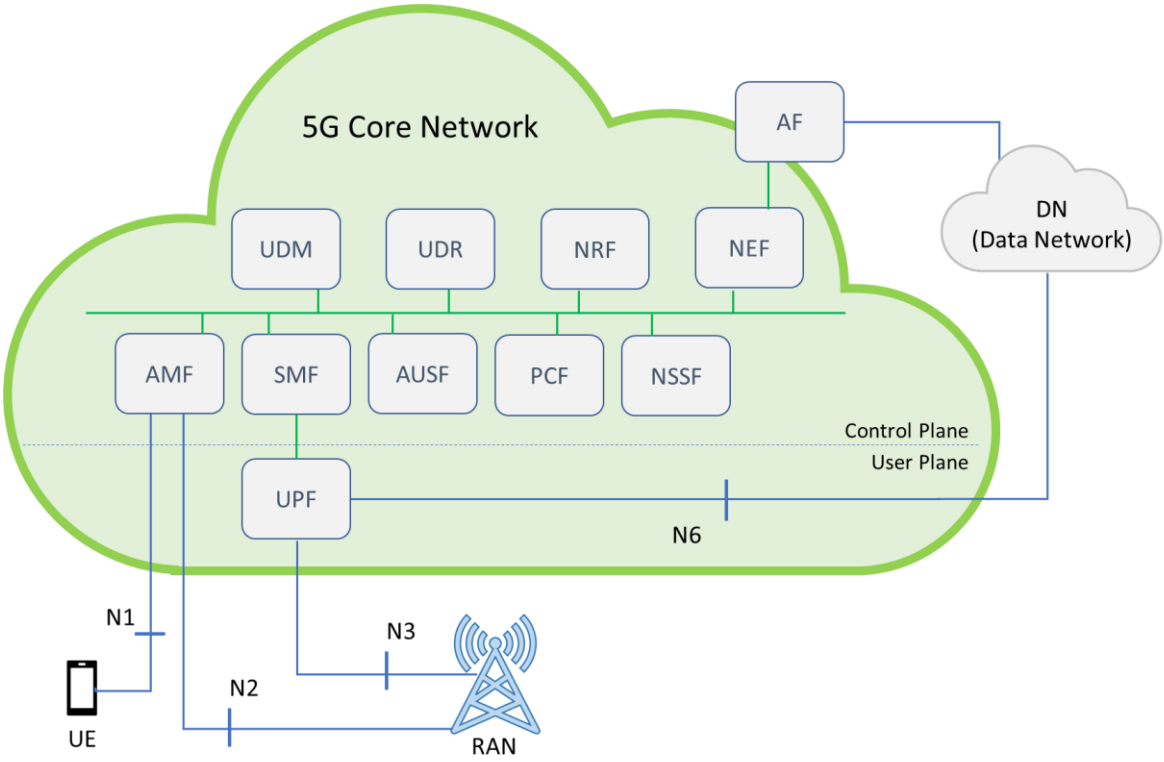


Figure 2-7 - The 5GC network and its interfaces with the elements of a 5GS.

Figure 2-7 represent the 5GC architecture with the NFs that we have just introduced (notice, though, that the list is not exhaustive). The control-plane NFs are characterized by service-based interfaces. Figure 2-7 also points out the reference points between the 5GC, the UE, the RAN, and the data network, labelled according to the 3GPP standard notation [4]

State-of-the-art deployments of the 5GC are fully virtualized, enhancing its flexibility and adaptability to different scenarios. Among the numerous features of a virtualized 5GC, Smart5Grid will leverage the possibility of distributing core NFs at the edge of the mobile network. In this manner, NFs are brought as close as possible to the RAN equipment and the end users. Notice that the edge deployment of the UPF is a key enabler for edge computing. Indeed, the UPF can select specific traffic coming from the data plane and maintain it local (with respect to the UE) via appropriate IP routing, directing it straight towards the servers where edge applications run. In this manner, Smart5Grid is enabling the implementation of the system architectures standardized by the ETSI MEC ISG (Multi-access Edge Computing). Figure 2-8 depicts a possible manner to implement such traffic localization within the corresponding blocks of the Smart5Grid reference architecture.

Equally importantly, the 5GC supports network slicing, which is a logical partitioning of the telecommunication network's physical and virtual resources. Network slicing has several purposes and advantages, and it is generally used to isolate independent services with different performance requirements, reserving the allocation of selected network resources to specific users or use cases, and mutualizing instead those resources that can be efficiently shared. For instance, in a deployment like that

of Figure 2-8, the UPF and the RAN at the edge may be part of a network slice strictly dedicated to serve the connected equipment of the energy infrastructure, whereas the centralized control-plane core NFs can be shared with other network slices. Such a choice would guarantee a high-performance exchange of data at the user plane, without subtracting resources to the slice serving the UEs of the smart grid. Simultaneously, this would happen without redundantly instantiating a whole dedicated core network, allowing to share the control plane with other network slices.

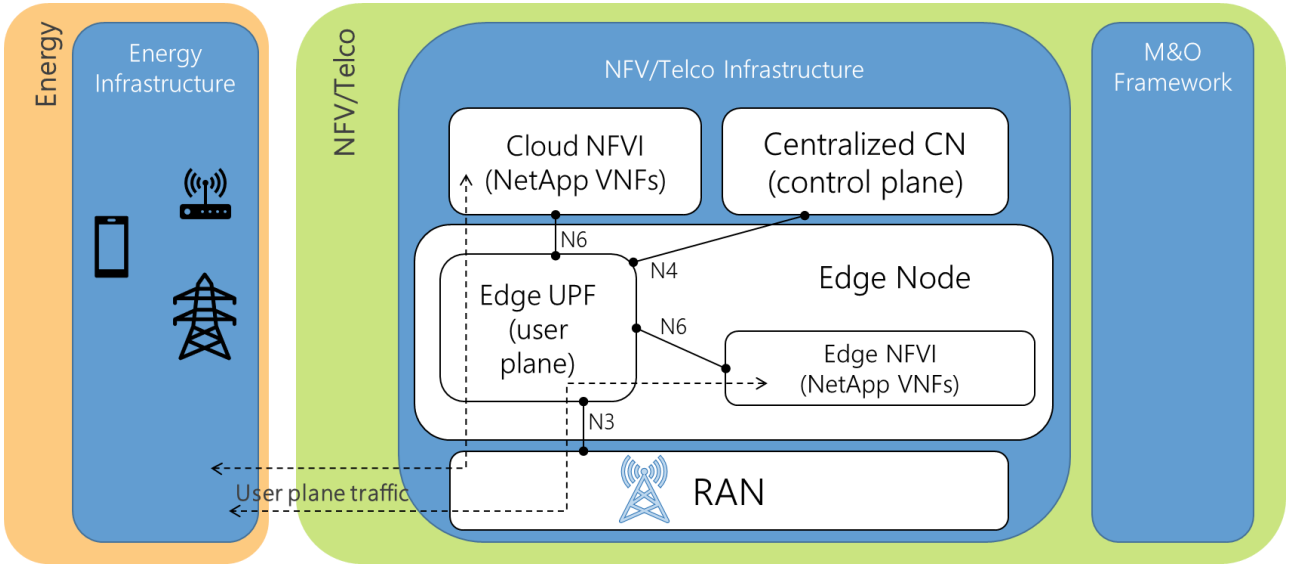


Figure 2-8 - UPF deployment for traffic localization at the edge.

2.5.2. Radio access network

2.5.2.1. The evolution of cellular technology

As referred in [5][6] 5G Base Station uses New Radio (NR) technology and is referred to as a gNodeB (gNB). The new radio access technology for 5G is called New Radio (NR) and replaces LTE. The gNodeB and replaces the eNB (or eNodeB/Evolved Node B). The evolution of Networks with the corresponding Services Evolution are illustrated in Table 2-13.

Table 2-13 - Evolution of networks and services for each generation.

Generation	Radio Technology	Base Station Name	Evolution of Services
2G	GSM	BTS (Base Transceiver Station)	Improved voice and text messaging over digital cellular technology
3G	UMTS	NodeB	Integrated voice and affordable mobile internet up to 2 MBps

4G	LTE	eNB, Evolved NodeB	High-Capacity mobile multimedia up to **100 Mbps**
5G	NR	gNB, gNodeB	5th generation network for people,1st generation for machines. In May 2020 NOKIA broke the world record of **4.7 Gbps**

2.5.2.2. 5G Spectrum: Frequency Bands

Frequency bands for 5G NR are being separated into two different frequency ranges:

- Frequency Range 1 (FR1) includes sub-6GHz frequency bands, some of which are bands traditionally used by previous standards but has been extended to cover potential new spectrum offerings from 410 MHz to 7125 MHz.
- Frequency Range 2 (FR2) includes frequency bands from 24.25 GHz to 52.6 GHz. Bands in this millimeter wave range have shorter range but higher available bandwidth than bands in the FR1.

2.5.2.3. RAN infrastructure in Smart5Grid's use cases

We provide in this section a brief overview of the RAN deployments in the project's use cases. Further details and updates will be the object of the reporting activities of WP5 and WP6. A description and analysis of the use cases is available in [1]

2.5.2.3.1. UC1 DEPLOYMENT

The 5G NR access network of WI3 is currently based on a Non-Standalone (NSA) architecture, specifically according to the Option 3x, which makes use of the E-UTRAN – NR Dual Connectivity (EN-DC) functionality introduced by 3GPP in Rel15. Basically, such architecture allows to address use cases related to eMBB (enhanced Mobile Broadband). In 2023/2024, W3 is to introduce its 5G CN, having the W3 NR aligned with Rel16 3GPP. Therefore, the 5G whole solution will also support the Standalone (SA) architecture, precisely making use of the Option 2, so as to deal with specific requirements related to URLLC (Ultra Reliable Low Latency Communications) and mMTC (massive Machine Type Communications) use cases.

With regard to the radio equipment used as gNodeBs in the WI3 NR, these, in most cases, operate in the FR1, as defined by 3GPP. In this sense, WI3 has already made a massive rollout of 5G FDD, making use of Base Band Units (BBUs) and Radio Remote Units (RRUs) supporting the "Dynamic Spectrum Sharing" (DSS) solution between LTE and NR. In addition, WI3 is also deploying gNodeBs in band N78 TDDD, using BBUs and Active Antenna Units (AAUs) that support the Massive MIMO technology to increase spectrum efficiency.

WI3's network offers 5G coverage in 95.4% population with Dynamic Spectrum Sharing in B3 and B7 frequency bands. 5G TDD coverage is offered to 45.6% of population in N78 band (60 MHz). The municipality of Olbia is currently served in 5G DSS technology with more than 20 sites reaching 96%

population, 5G TDD is instead offered to 65% of population. 5G TDD is expected to strongly increase due to a massive roll out plan for N78 deployment in existing sites.

2.5.2.3.2. UC2 DEPLOYMENT

For UC2, a single gNodeB is deployed to provide radio access network coverage for the UEs in the substation premises. Depending on the spectrum that will be available (current options are N77 and N78) radio units from the vendors Sunwave or AW2S are targeted, equipped with directive panel antenna to provide coverage to the substation. A dedicated optical fibre connects the gNodeB with the BBU hosted close to the radio access network deployment inside the technical room of the substation. The final model of both gNodeB and the antenna will depend on the spectrum available to the UC and the final location of the user equipment (see next subsection).

To provide 5G connectivity to the cameras and sensors, two CPEs will be deployed in the substation. Each CPE supports SA mode and will be connected over Ethernet to the sensors and cameras used to detect workers and equipment: the set of cameras will be connected to one CPE and the industrial PC serving the UWB sensors will be connected to the other CPE. This way, different types of traffic towards/from the cameras and UWB sensors, respectively, can be differentiated by CPE. The SIM cards featured in UC2 will be programmed with testing PLMNIDs that will correspond to the 5G Core deployed in the private 5G network.

2.5.2.3.3. UC3 DEPLOYMENT

UC3 wind turbine is situated in South-Eastern Bulgaria. Currently there is no 5G coverage at the power plant site and as Vivacom's 5G network rollout plan is still under development, a physical duplication of the power plant signals will be made on a server, located in Vivacom's lab in Sofia, emulating a "digital twin" of the wind farm's signal list.

Wind farm top level demo case topology:

I. All data from turbine sensors is received by the SCADA system. The measurements from selected sensors are sent via OPC-UA protocol to the raspberry pi which is located at Sofia.

II. The data from the sensors are processed and reproduced by raspberry pi as an output signals on the raspberry's pins (as a PWM wave).

III. Another raspberry pi with 5G connectivity will read the values from the first Raspberry pi pins

OPC (interoperability standard for secure and reliable exchange of data) is implemented as a server/client pair. The communication between Server and the Client is VPN secured.

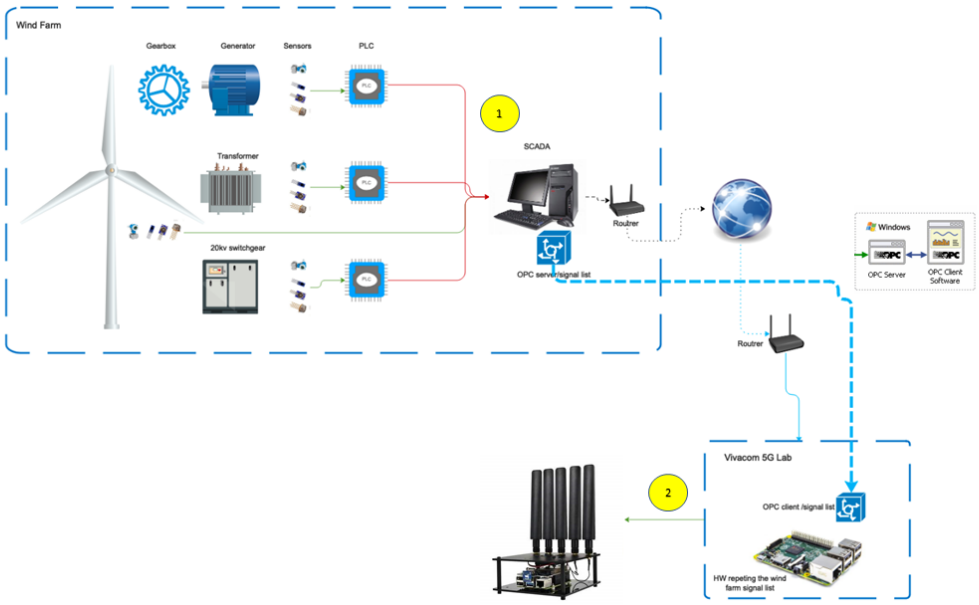


Figure 2-9 - UC3 Wind farm demo case topology

Infrastructure at Wind turbine site (1)

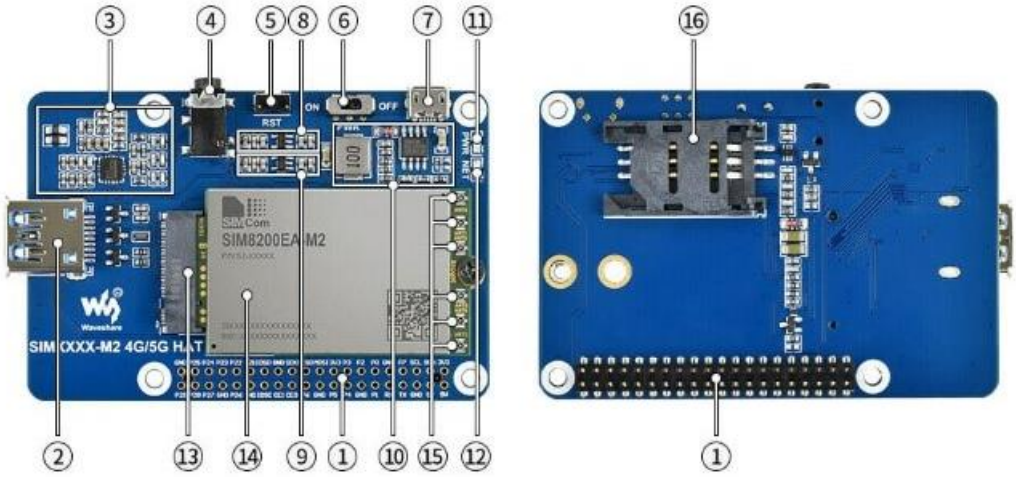
1. We have installed OPC server on the SACDA system on-site infrastructure
2. OPC server is reading the selected signals from the Turbine (using SCADA)

Infrastructure in 5G Lab (2)

1. We have installed OPC client on Raspberry Pi in the lab
2. OPC client is reading the selected signals from the OPC server
3. Selected signals are emulated on the GPIO pins.
4. Another raspberry pi, equipped with 5G Raspberry pi hat, reads the signals (also process them if needed) via its own GPIO pins.

The data will then be sent to the NetApp through 5G.

There would be a slight delay in the data due to the communication between OPC Server >> OPC Client and emulation of the signals on the first Raspberry Pi. That is of no concern for the demo case though, as two raspberries will be physically connected to each other through wires and the data will be read in real time and thus 5G lab conditions will be in line with demo case objectives and requirements.



1. Raspberry Pi GPIO header
2. USB3.1 port
3. Audio drive circuit
4. Audio jack
5. Reset
6. Power switch
7. 5V 3A input
8. 5V to 3.3V translator
9. 5V to 1.8V translator
10. 5V to 4.3V translator
11. Power indicator
12. Network indicator
13. M.2 connector
14. SIM8200EA-M2
15. Antenna connector
16. SIM card slot

Figure 2-10 - 5G Hat technical specification.

Results of this demo will be taken from the point in which the 5G-Hat-enabled Raspberry Pi (equipped with 5G SIM card) reads and receives the data in a millisecond precision.

For UC3, a single gNodeB is deployed to provide radio access network coverage for the UEs in the 5G lab premises. The HW chosen for this setup is a Huawei AAU5639 (TX64/RX64), equipped with a directive panel antenna (coverage angle with horizontal scanning range of broadcast beams of 120 degrees), gain 25 dBi, frequency band N78 (3500-3700MHz). A dedicated optical fibre connects the gNodeB with the BBU hosted close to the radio access network deployment inside the 5G lab. As per the current 5G architecture (NSA) at Vivacom this is the expected network diagram:

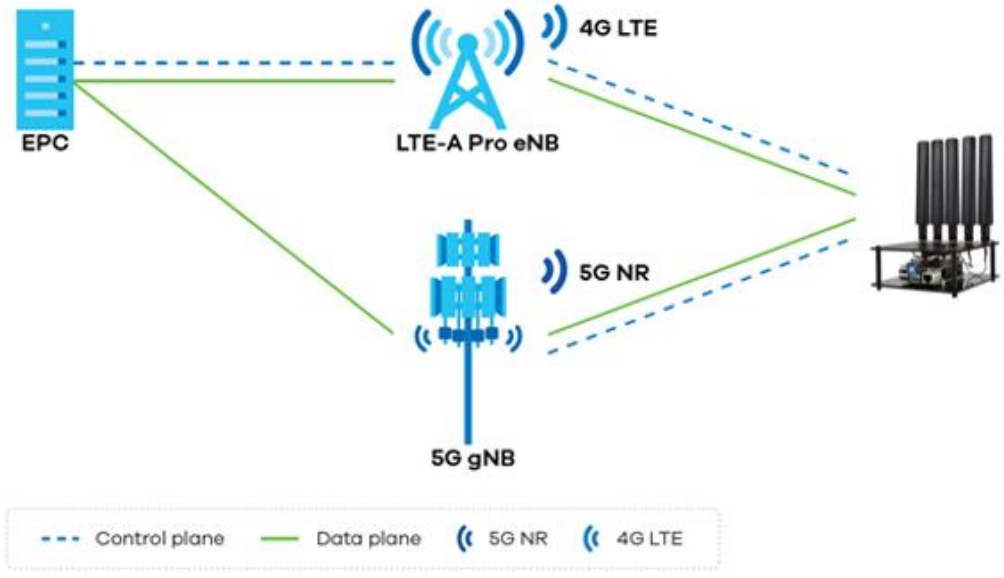


Figure 2-11 - 5G NSA architecture of UC3.

In case of deployment of SA 5G architecture, potential solution will change as follows:

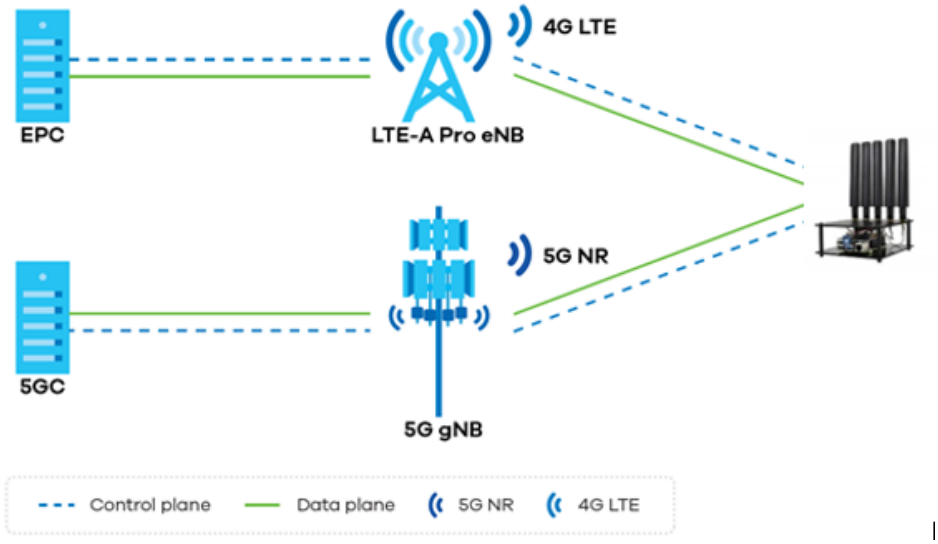


Figure 2-12 - 5G SA architecture of UC3.

2.5.2.3.4. UC4 DEPLOYMENT

For UC4, in order to provide access to the 5G network for the PMUs under IPTOs premises, it is decided to use the NR7101 5G New Radio Outdoor Router [7]. The Zyxel NR7101 5G New Radio Outdoor Router supports the latest 5G NR technology and at the same time remaining full compatibility with 4G/3G networks:

Figure 2-13 - Zyxel NR7101 5G New Radio outdoor router to be used for PMUs coverage at UC4.

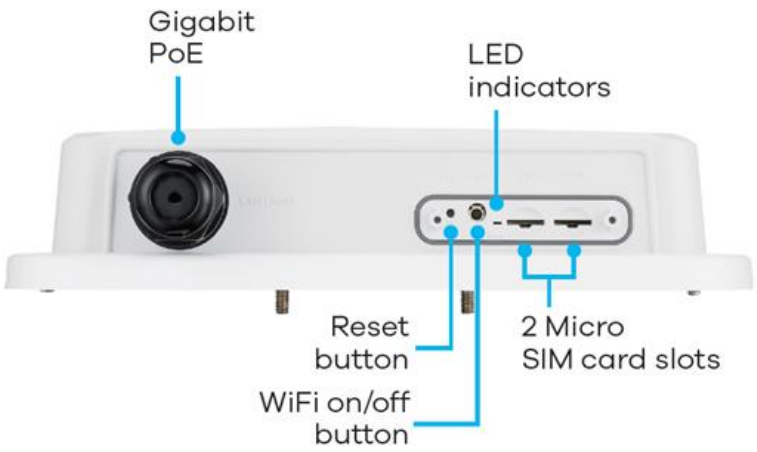


Figure 2-14 - Interface description, Zyxel NR7101.

As it is illustrated in Figure 2-14, the Router interface consists of 2x 3FF format Micro SIM card slots, 1x GbE RJ-45 PoE LAN port, 1 WiFi on/off button, the corresponding LEDs for WiFi, Reset and power status and 1x reset to default button.

The benefits of using the NR7101 router can also be summarized as below:

1. Internet Connectivity of speeds higher than 5Gbps for 5G NR and up to 2Gbps for 4G LTE DL speeds. Environmental Durability is an important feature. The Router is contained in a hardened IP68-certified enclosure with industrial-grade components and supports deployment in wide territorial scopes.
2. Remote Management and upgrade. Through the LTE radio interface, NR7101 supports TR-069, remote GUI and FTP to be fully configurable. Support of mobile apps (iOS and Android) and interoperable with Standalone (SA) and Non-Standalone deployments (NSA). The deployments for both cases are depicted in a similar manner as is depicted for UC3 in Figure 2-11 and Figure 2-12.

For the needs of UC4, a gNodeB will have to be installed by OTE at IPTO's premises in the northern part of Greece near Thessaloniki. The product that is chosen to be installed is from Ericsson, specifically the RADIO 4408 [8] which is part of the Ericsson Radio System Portfolio.

Figure 2-15 - Ericsson's Radio 4408 to be installed at IPTO sub-station.

Among the advantages of using such a product is the great radio performance and power efficiency when it comes to medium range 3GPP radio products. It also has a great flexibility and a wide range of mounting scenarios, thus making it easier to make small and efficient single and multi-band radio installations. Radio 4408 supports LTE TDD with four duplex TX/RX branches supporting up to 4 x 5 W output power.

2.6. Telemetry and data analytics

In the Smart5Grid platform the Telemetry component is a module equipped with monitoring and analytics functionalities in charge of monitoring the resources used within a given infrastructure and making them available to the different modules if requested. It assures that the infrastructure is used efficiently and guarantee the service performance.

This module is built around streaming data pipelines that reliably get data and apply a set of transformations (e.g., estimation of rate of change of a metric, calculation of average values in specific time windows) or filtering functionalities. The Kafka distributed streaming platform is envisaged to be used for this purpose. In a more general way this module has been designed to be able to collect information from the radio infrastructure and the cloud infrastructure by collecting information from the VNFs, the NFVI, the VIM and the MANO. The development of this module is customizable so depending on the UCs different data source can be available ad exploitable.

In more detail, the component meets the following specifications, and every data source can be managed if requested by other components/modules:

1) The component can share different types of data for every instantiated NSs/VNFs orchestrated by the M&O framework of SVP, including CPU usage, memory allocation, active/passive network throughput, packet loss, etc.
2) The component can share data that allow to make possible the tracking for VNFs and NetApp instantiated over every virtualization environment (e.g. VM-based, container-based) and underlying NFVI (e.g. OpenStack, VMware,etc) which is supported by the Smart5Grid platform,
3) It can share VNF-specific and NS-specific alerting rules for notifications directed not only to internal functional blocks (e.g.NFVO) but also the external actors (e.g. developers, service providers).
4) It must ensure that the monitoring services work in unison with M&O functional blocks and provide the many monitoring information they need to fulfil their management and orchestration tasks.

All the metrics in principle can be stored in an external database for future reference and offline analysis. Confidentiality and integrity of exchanged data in the bus, as well as authentication and authorization of pub/subs clients are in line with the Smart5Grid AAA services. The management and the configuration of the Kafka[13] bus, the considered data models in messages exchanges, the traffic streaming policies, as well the types of metrics which are collected per environment (either NFVI or the application layer) will be thoroughly presented in D3.2. Note that the Kafka bus is open to all the internal services or entities that may interest to access, retrieve and consume either stored (in the database) or current metrics from the tracked environments.

[13] https://kafka.apache.org/

2.6.1. Technological choice for the telemetry and data analytics module

For the development of the telemetry and data analytics module has been chosen the Kafka Technology. Kafka, which was initially created and open-sourced by the social network company LinkedIn in 2011 was originally designed as a messaging queue middleware (i.e., software acting as mediator between applications or systems), during time Kafka has evolved into a high-performance, distributed streaming platform that provides three main functionalities: (a) publish and subscribe to real-time applications or topics, (b) store event streams or data records in a fault-tolerant way, and (c) process data as they occur. Recently, Apache Kafka was integrated in Confluent.io[14] into a real-time streaming processing ecosystem consisting of a large collection of infrastructure services such as database connectors and configuration managers.

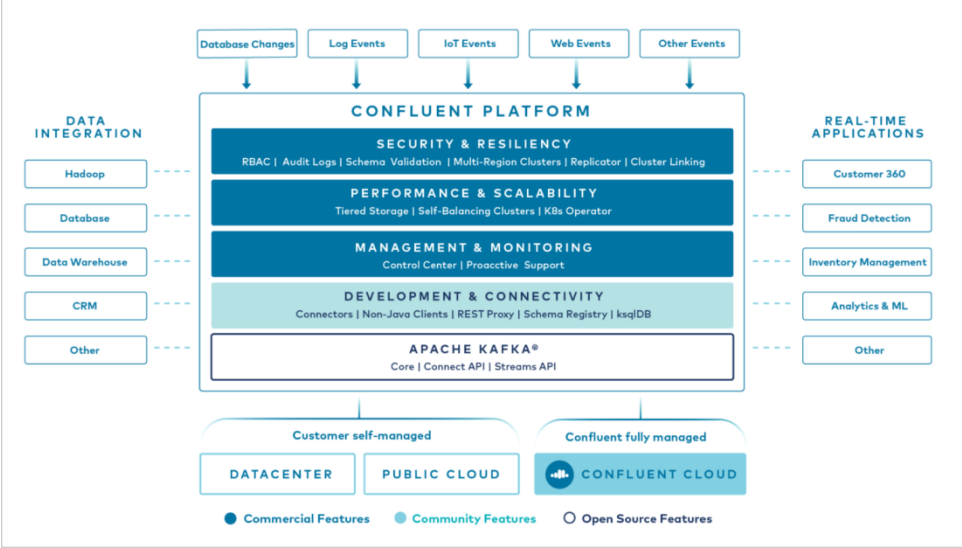


Figure 2-16 - Confluent platform components extracted from [9]

Confluent platform allows real-time streaming of data, publication and subscription of messages, data storage and processing. At the same time, each Confluent release includes the latest Kafka distribution and also other additional tools and services to manage the data streaming making it the natural project's solution choice.

2.6.2. Data pipeline

A data brokering mechanism follows the Figure 2-17 but given that Confluent Data streaming platform based on Apache Kafka, is adopted as core component for this module we can allows real-time streaming of data, publication and subscription of messages, data storage and processing. At the same time, each Confluent release includes the latest Kafka distribution and also other additional tools and services to manage the data streaming making it the natural project's solution choice

[14] https://www.confluent.io/product/confluent-open-source

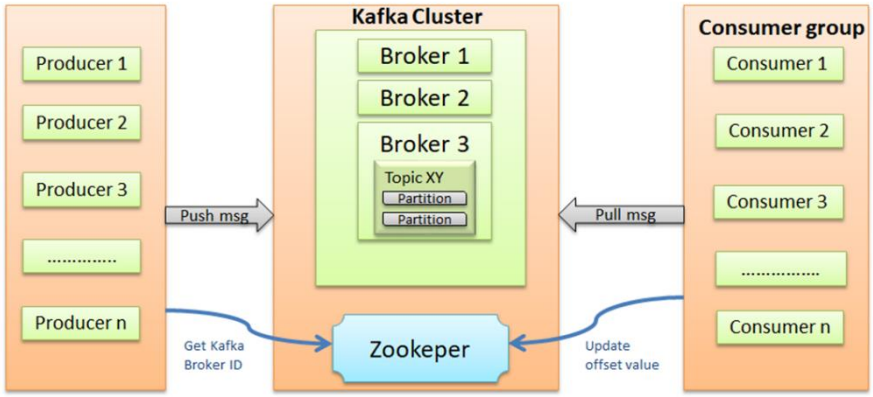


Figure 2-17 - Apache Kafka architecture.

With respect to the general architecture presented in Figure 2-17, the data pipeline flow of the telemetry component follow a more complicated dynamic that is illustrated in Figure 2-18.

Data producers could be a general source of information related to cloud or telco infrastructure as well as another program or a database that is producing information. These are generating some data and writing to a Kafka topic. Now for databases to act as source to Kafka you need to use Kafka source connector based on your database type to source in all data to a Kafka topic. Kafka Connect is a tool for scalable and reliable data streaming between Apache Kafka and other data systems. It makes it simple to quickly define connectors that move large data sets into and out of Kafka. Kafka Connect can ingest entire databases or collect metrics from all your application servers into Kafka topics, making the data available for stream processing with low latency. An export connector can deliver data from Kafka topics into secondary indexes like Elastic search or into batch systems such as Hadoop for offline analysis.

In case of the data structure shared could need an analysis with DB functionalities it is possible to provide a simple and completely interactive SQL interface for processing data in Kafka built on Confluent KSQL that enables real-time data processing against Apache Kafka. You do not need to write stream processing code in Java (as is the case with Kafka Streams API). In fact, KSQL allows you view your data as an unbounded STREAM of data on which you can run SQL queries to filter based on columns, JOIN two streams, or run some standard aggregation functions on a stream such as COUNT, SUM, MIN, MAX, AVG, TOPK etc.

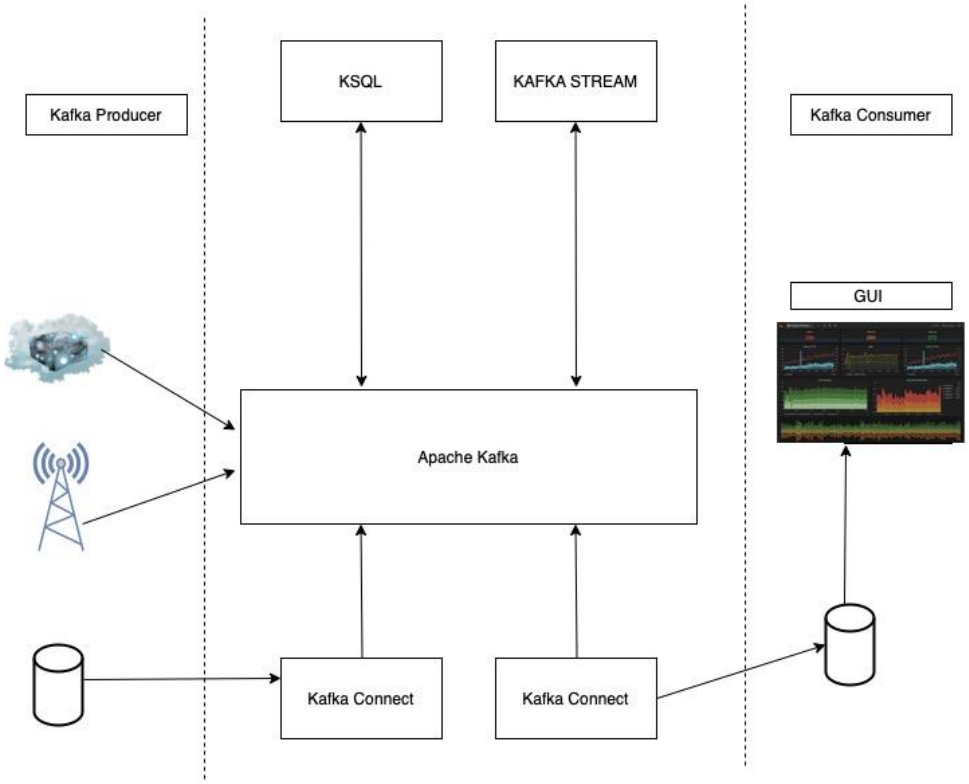


Figure 2-18 - Telemetry pipeline.

Summarizing the telemetry component will make available different topics in which the different components of the network and telco infrastructure will act as producer of message /metrics that will be then extracted from other components (consumers). In the Table 2-14 an example list of different metrics make available from the producers of data is detailed.

Table 2-14 - Collectable metrics.

Metrics	Unit	Description	Publisher of the information	Part of the infrastructure
Memory	megabytes	Volume of RAM allocated to the instance	NFVO	Cloud
Container cpu load average	Seconds	Value of the container cpu load average over time t	NFVI	Cloud
NetApp metrics	Various	Liveness, metrics and KPIs of a given NetApp instance	NetApp metric exporter	NetApp
NS resource utilization	CPUs load, memory, disk bandwidth, network bandwidth, et cetera.	All the metrics about the resource utilization of deployed network services	NFVO	Cloud/Edge

NetApp resource utilization	CPUs load, memory, disk bandwidth, network bandwidth, et cetera.	All the metrics about the resource utilization of deployed network services grouped by their NetApp	NetApp Controller & MECO	Cloud/Edge
NetApp Controller Metrics	Various	Liveness, metrics, and alarms regarding a NetApp Controller	NetApp Controller & MECO	Cloud/Edge
Various and depending on the UCs	The units depend on the different needs of the UCs. It can cover different functions and related metrics	Access to the specific data or simple analytics of each 5G system function, such as AMF, SMF, etc., as well as 5G RAN.	5G system function, such as AMF, SMF, and 5G Radio Access Network	Radio and core network

3. NetApp operation support and orchestration

This section describes the steps that are part of the management and orchestration of the NetApp lifecycle.

3.1. Infrastructure management

3.1.1. Resource registration

3.1.1.1. Slice manager: registration of infrastructure resources

One of the main functionalities of the SM is to be aware of the infrastructure resources where the network slices will be created and managed. That is, in order to execute the network slicing lifecycle management procedure, the SM needs to know the resources where the slices will be deployed. In light of this, the SM needs to register each node and network element that compose the network infrastructure layer. Therefore, to ensure that a network slice has end-to-end communication two main elements must be registered into the SM: network function virtualization infrastructures (NFVIs) and RAN devices.

1. On the computing side, the SM by using the SBI contacts the NBI of the VIM available in the infrastructure layer (cf. Section 2.3. The SM basically performs access-authorization procedures to register the computing resources. It sends requests with the right credentials (e.g., user, password, URL) to create a client/project with admin permissions and specific quota. In this way, the SM can check the availability of the computing resources (e.g., reachable/unreachable) and its capacity (e.g., saturated). In Figure 3-1 it is shown the workflow followed to register computing resources into the SM.

   In addition, the SM can register network resources associated with a VIM, which are then provided to slices in terms of chunks. For this purpose, the SM via its SBI communicates with the VIM to create the required provider networks. The network resource registration is implemented in terms of IP pools, VLAN tag ranges and bandwidths.

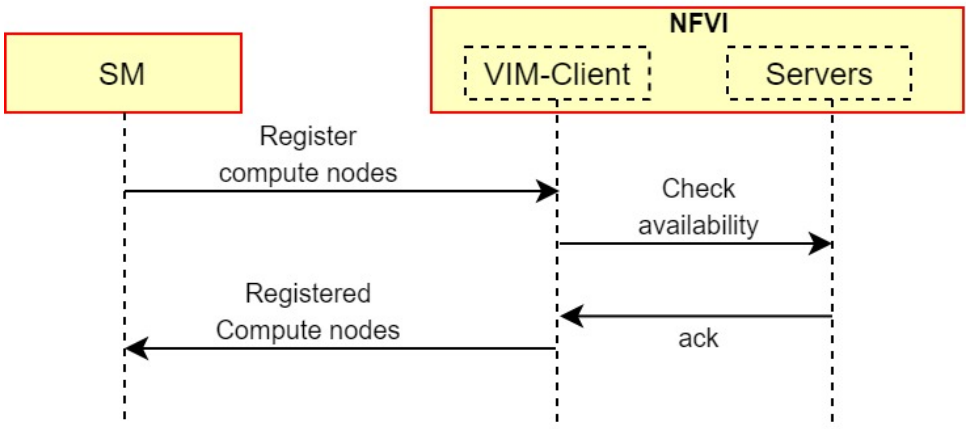


Figure 3-1 - Compute node registration workflow.

2. On the other hand, SM not only handles computing resources, but also RAN resources. These resources are not directly managed by SM, but by a dedicated RAN controller that is aware of

the radio resources of an infrastructure by exposing these resources to the SM, it can generate slices that support services with user connectivity.

The registration procedure done by the RAN Controller can be proactive and reactive. This process depends on the radio vendor and how it is integrated with the RAN Controller. Once a device is registered, typical management protocols, such as SNMP and NETCONF are used to communicate with the radio devices and apply settings. As such, through its SBI, the RAN Controller has access to the configuration of the RAN (physical parameters, services) and can perform basic management functions on the radio devices, such as registering, configuring, deleting, and retrieving information. The abstraction of the radio infrastructure provided by RAN-Controller through its NBI, allows higher layer modules such as the SM to be aware of the radio resources to then configure and manage them. In this sense, SM through its SBI can perform the basic management operations listed above.

Following, in Figure 3-2 an example of the RAN device registration workflow is presented.

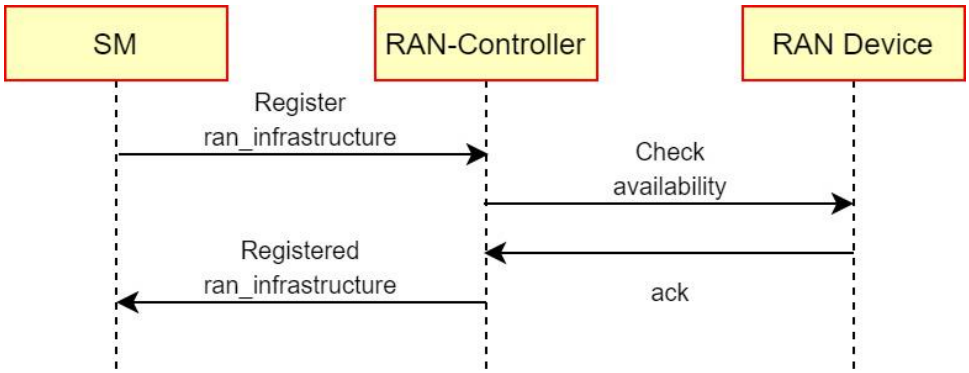


Figure 3-2 - RAN device registration workflow.

3.1.1.2. NetApp Controller & MECO and VIM: Node registration and provisioning

One of the main functionalities of the platform regarding infrastructure management is the ability to register and provision compute nodes. While node registration adds a node and makes sure that the nodes is configured by running the specific agents required by the Smart5Grid platform, node provisioning extends this concept by also configuring all the required software stacks, from the OS to the Smart5Grid agents.

The interfaces used to perform node onboarding and provisioning depend on the VIM used in each use case. Thus, it is up to the NetApp Controller and MECO to have the right adapters in place for the VIM selected (cf. Section 2.2.) In any case, one can find the functional and technical specification of node onboarding and provisioning in Section 4.4.2.1 of [2].

3.1.2. Resource monitoring

As detailed in Section 4.4.2.1.3 of D2.2, the NetApp Controller & MECO constantly monitors the metrics exposed by a NetApp to guarantee the Service Level Objective (SLO) of every NetApp instance. These service metrics are combined with the metrics regarding the network and infrastructure to take decision on a NetApp lifecycle. The part of the infrastructure that are constantly monitor are the compute nodes, the 5GC and the RAN.

For what concerns the infrastructure metrics, like outlined in Section 4.4.2.1.2 of D2.2 once a node is onboarded, the installed agents are used to extract information regarding the node itself. This information includes but is not limited to node configuration (e.g., IP addresses, OS, software stack), resource utilizations (e.g., CPU and DRAM utilization), network latency and bandwidth. While some of this information is static, other is continuously changing and it is stored in time series databases like Prometheus. The usage of time series data bases able to evaluate expressions that aggregate data over time allows the creation of alerts and alarm, upon which components like the NetApp Controller & MECO can take decision or signal issues. When a node is ready and its agents running this is the flow of the data regarding resource monitoring:

1. Each agent periodically collects its data and publishes it to an HTTP(S) endpoint (e.g., agent-name.node-uuid.com:9090/metrics)
2. The time series DB periodically visits every agent endpoint and scrapes their published metrics.
3. A separate thread of the time series DB periodically checks every defined alarm and if they trigger. If an alarm does trigger, the DB publishes a message on a queue so that other components can read it and react as needed.

On the 5GC and RAN side, the monitoring system described in Sections 2.6 includes software and hardware tools that can track various aspects of a network and its operation, such generic information about the usage of the radio up and downlinks (e.g., bitrate, transmitted packets, average MCS used, average delay, etc.), but can be also more specific, such as exposing connectivity information for UEs connected to the 5G network. Depending on the network solution (vendor, software framework that controls the devices), these metrics may differ and be exposed in a different way, e.g., via a database, using Prometheus, or any other solution. Often, these metrics can be exposed graphically, e.g., via a dashboard that pulls the metrics from where they are stored, allowing to review the behavior of the 5G network connectivity. Software agents running in the Smart5Grid platform as part of the monitoring system can pull and surveil relevant pieces of this information to assure that SLAs are met or to trigger alarms otherwise. When detecting failures or lacks performance, the monitoring system can alert administrators and deliver reports using network analytics. To perform this network monitoring, different protocols can be used including:

- SNMP (Simple Network Management Protocol), that is an application-layer protocol that uses a call-and-response system to check the status of many types of devices, from switches to printers. SNMP can be used to monitor system status and configuration.
- ICMP (Internet Control Message Protocol) can send IP-operations information and to generate error messages in the event of device failures.

3.2. NetApp lifecycle

This section describes the Smart5Grid NetApp lifecycle, once a NetApp has been developed and packaged.

At this point, the lifecycle of the NetApp is managed by the M&O Framework by the NetApp Controller & MECO component. To this effect, the NetApp Controller & MECO offers a Northbound API for allowing OSS/BSS elements to control this lifecycle (cf. Section 2.2). As described in the following

subsection, the NetApp Controller & MECO delegates the management of certain components of the NetApp to other entities of the M&O framework, such as the Network Services lifecycle to the NFVO and the Slices lifecycle to the Slice Manager.

As shown in Figure 3-3, a NetApp lifecycle is divided in four parts: Onboarding, Instantiation, Operation, and Termination. During the Onboarding phase, the NetApp package is decomposed and the Network Services that form it are onboarded to the respective NFVO. At the Instantiation phase, placement requirements are addressed, resources are allocated, and the service is deployed over these resources. The completion of the previous phase triggers the start of the Operation stage. At this stage the NetApp metrics are monitored and constantly evaluated to trigger scaling and migration operations. Finally, upon request, a NetApp can be terminated, implying the shutting down of services and liberation of reserved resources.

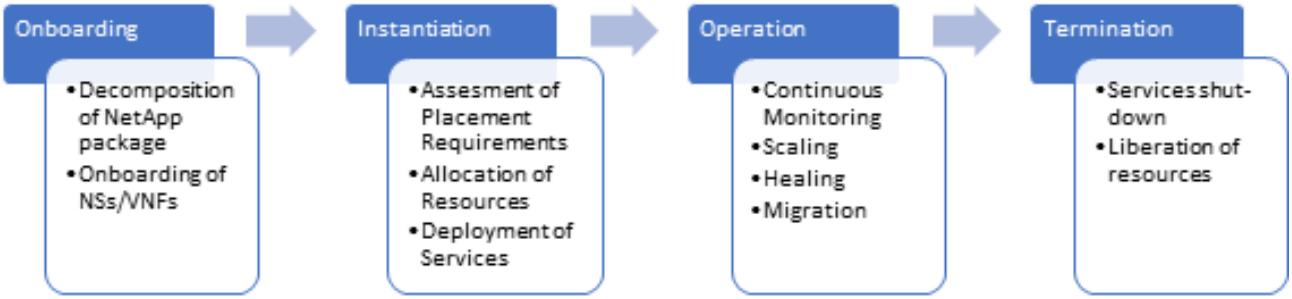


Figure 3-3 - NetApp lifecycle.

3.2.1. NetApp onboarding

The NetApp Controller & MECO receives NetApp onboarding requests from its front end, usually either from the V&V framework or from the user. Every request to onboard a NetApp comes with a NetApp Descriptor (see Section 4.6 of D2.2) that completely defines the NetApp and its characteristics, including all its NS and VNF. Upon an onboarding request the NetApp Controller & MECO, the first step is to validate the descriptor syntax. If this is correct, the onboarding process can proceed by storing and adding the NetApp to the NetApp Controller & MECO.

Once the NetApp is onboarded in the NetApp Controller & MECO, the controller can proceed to onboard the NetApp NSs and VNFs in the NFVO. Note that, depending on the NetApp Controller & MECO implementation, the NS and VNF onboarding in the NFVO can be done in a lazy fashion. That is, the onboarding is only done when deploying the NetApp in an NFVO instance for the first time. This allows to save storage space on the edge nodes and onboards the NetApps only in the NFVOs of the sites where they are really needed.

The onboarding of NSs and VNFs packages or descriptors consists in loading these packages into the NFVO so that they can be registered in its catalogue. There is a set of steps required for onboarding these packages defined by ETSI in Release 3 of Network Functions Virtualization (NFV) [11][10]. However, as ETSI is still studying the onboarding and instantiation feature in Release 4, see ETSI GR NFV-REL 011 V1.1.1 [11], and this report has not yet been finalized in [12], this specification can serve as a reference guide of the steps required. To perform the onboarding, the order of uploading the packages must be,

first upload the containerized VNFs packages and, once these are uploaded, upload the NS packages (the descriptor of the Network Service refers to the VNF Descriptors (VNFDs) and therefore it must be able to see them in NFVO before performing the onboarding). The main steps to incorporate the containerized VNFs packages are:

1. The NFVO receives a request to incorporate a containerized VNF package.
2. The NFVO checks that the VNFD contains the mandatory elements and validates the completeness and authenticity of the VNFD through a manifest file. After everything is correct, the NFVO notifies the catalogue.
3. If the containerized VNF package contains any images in the package, the NFVO loads these images into the Container Image Registry (CIR).
4. The NFVO loads the declarative descriptor and configuration files into the Managed Container Infrastructure Object Package (MCIOP).

Once all VNFs of a NS are onboarded to the NFVO, the incorporation of the NS can be performed following the next steps:

1. The NS package is uploaded to the NFVO using the Network Service Descriptor interface, performing the NSD on-boarding operation.
2. The NFVO checks that the Containerized VNF packages that are part of that NS are in the orchestrator, checks that the NSD contains the mandatory fields as well as the presence of the necessary external interfaces that the Network Service has to provide in each VNFD and validates the integrity and authenticity of this service.
3. The NFVO shall notify the catalogue of the addition of this network service with its version and inform the sender of the successful addition of this service.

3.2.2. NetApp instantiation

To instantiate a NetApp, some preliminary steps must be taken to prepare the network infrastructure. For example, once the NetApp Controller & MECO has read the requirements of a NetApp, it first sends a request to create a network slice to SM. That is, the NetApp Controller & MECO requires to the SM to reserve logical resources on the computing nodes and radio devices that are associated with a particular slice. The workflow of the slice creation procedure is depicted in Figure 3-4.

First, the NetApp Controller & MECO sends the request to the Slice Manager (Step 1). The SM verifies that it has resources registered with the characteristics requested by the NetApp Controller & MECO. Once the resources have been verified, SM interacts with both the VIM and the RAN Controller to create and ensure the logical resources for a specific slice (step 2). On the one hand, by attacking the VIM northbound interface, the SM secures reserves a specific number of computing resources such as CPU, RAM and Storage for the network slice associated with the user. Also, into the same VIM, the SM requests network capabilities for the network slice, such as IP's ranges, VLAN's tags, and quotas. On the other hand, the SM requests to the RAN Controller the radio resources that will be used to enable the RAN connectivity for users (step 2.1). Within these reserved resources, the NetApps will be able to

execute their required virtual network functions or applications according to the SLOs defined in their descriptors.

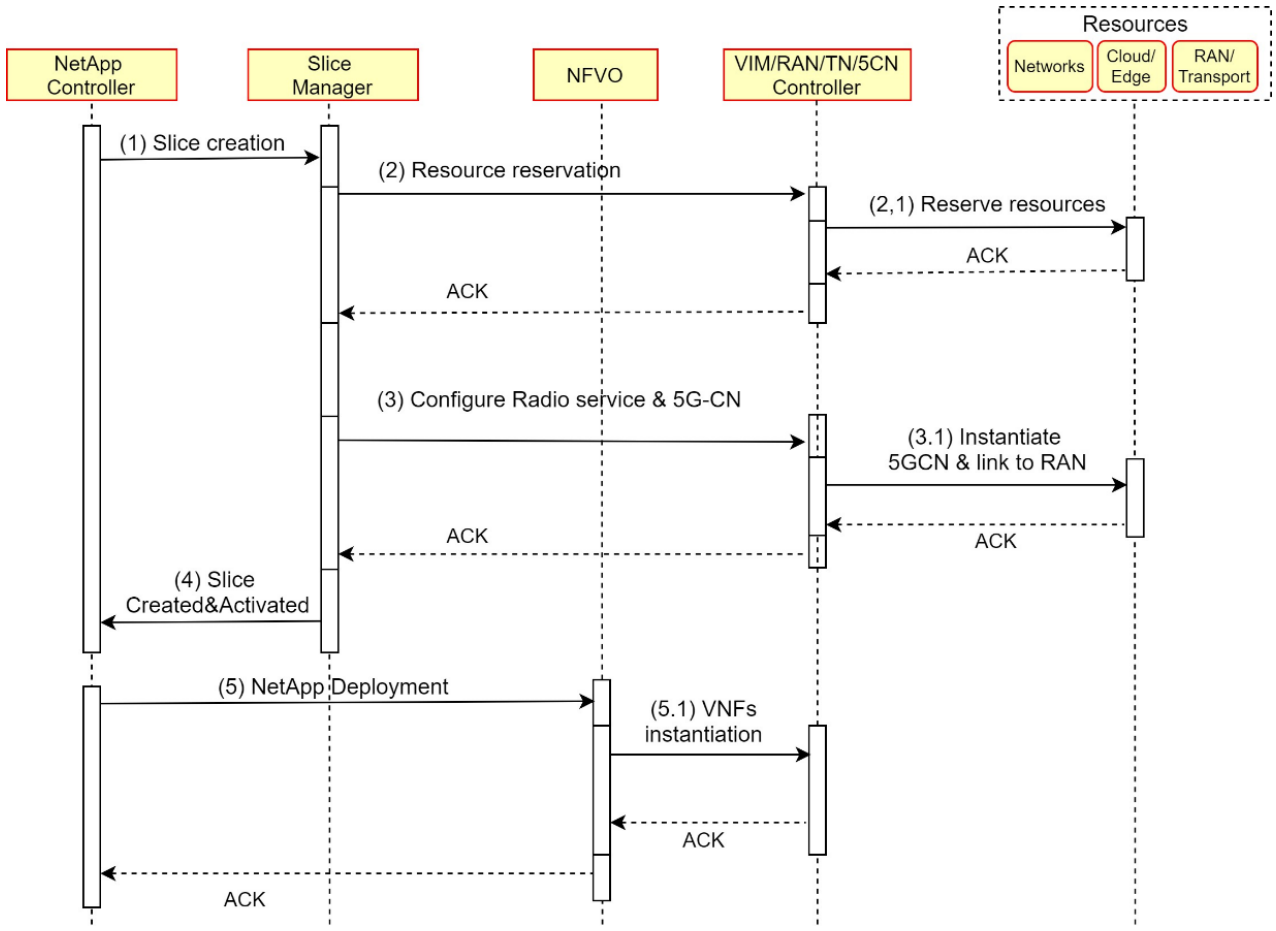


Figure 3-4 - Network preparation and resource allocation.

Once the SM resources have been reserved, if required, it creates a radio service where it prepares the deployment of the 5G CN and links to the radio devices associated with the network slice based on the Public Land Mobile Network (PLMN) ID information. The SM interacts with the VIM to deploy a 5G CN within the reserved computational resources (step 3). Besides, it applies traffic configurations and binds it to the radio devices reserved for the slice (step 3.1). At this point, SM returns an ACK to the NetApp Controller & MECO, informing that the creation of the network slice has been completed providing some relevant information, such as: name, slice_id, user_id, resources_ids (compute, network, radio), and activation_status (step 4).

Finally, to instantiate NetApps the NetApp Controller & MECO uses the NFVO's NBI (cf. Section 0). It requests the deployment of specific NSDs and VNFDs by pointing their identifiers information described on the NetApp Descriptor. Then, the NFVO starts the deployment procedure by using VIMs previously registered. Therefore, the basic network service instance lifecycle management is performed by the NFVO (step 5). Nevertheless, in some use cases the NetApp deployment procedure could be performed via the Slice Manager by taking advantage of some functionalities implemented in the components involved. In this case, the NetApp Controller & MECO could make calls to the SM pointing the NSD

identifier to deploy the NetApps. As Containerized VNF instantiation is defined by ETSI [13] this operation can be requested in several ways. In the case of Smart5Grid the request to instantiate VNFs is carried out through a Network Service that will perform the interconnection between VNFs if this is necessary once they are instantiated.

To implement containerised VNFs, the following cases must be followed:

1. When the NS requests the instantiation of containerised VNFs the NFVO prepares and activates the VNF Manager (VNFM) for containerised instantiation.
2. The VNFM requests the Container Infrastructure Service Management (CISM) to perform the instantiation, as it is responsible for the deployment, monitoring, and lifecycle of the containerized VNFs.
3. The CISM accesses the CIR to obtain the image defined in the descriptors, the CISM will use the information from the image(s) to perform the instantiation.
4. VNFM completes the configuration of the VNFs when all resources have been allocated by the CISM and communicates the completion of the instantiation.

Once all NS of a NetApp are successfully deployed, all traffic steering policies and rules of the NS needs to be applied to make sure that UEs can reach the NSs. To do this, polices and rules defined in the NetApp Descriptor are forwarded to the Policy Control Function (PCF) of the 5G Core and will eventually reach the UPF – component that will implement them (cf. the 5GC network function description of Section2.5. The communication with the PCF can be initiated by two components: an AF of the 5G CN Controller or by an AF of the NetApp Controller & MECO. In the former case, since the AF belongs to the 5G CN Controller, it is to be considered as trusted and hence the communication with the PCF is direct. In the second case, instead, since the communication is instantiated by an AF embedded in the NetApp Controller & MECO - which is an external component of the 5G CN - the AF is to be considered as not trusted. Hence, the communication will have to be indirect and to flow via the 5GC NEF before to reach the PCF. Both options are valid, and in the scope of Smart5Grid, each use case will decide which option to use considering the available AF functionality offered by the used NetApp Controller & MECOs and 5G CNs.

3.2.3. NetApp operation

To manage the lifecycle of a NetApp instance, the NetApp Controller & MECO combines the infrastructure metrics with the NetApp metrics to guarantee liveness and the Service Level Objective defined in the NetApp descriptor. To guarantee this, the possible actions are:

- NetApp Scaling: This action allows to scale one or more NSs of a NetApp based to meet current demands and given SLOs. Scaling requests may be manual or automatic.
- NetApp Healing: This action allows to heal a NetApp instance when its liveness probes fail to respond or report failures.
- NetApp Migration: This action allows to migrate a NetApp instance based on some criteria like network traffic, latency to the UEs etc cetera.

Note that from a practical point of view, these actions might require the re-deployment of a new instance of the NetApp (or of some of its NSs). When this happens, it is the responsibility of the provider of the NetApps (and their clients) to support session migration at the application level.

3.2.4. NetApp termination and decommissioning

The final aspect of managing a NetApp and its life cycle is its termination and decommissioning. This action includes decommissioning of non-shared constituents if required and removing the network slice instance-specific configuration from the shared constituents. After the decommissioning phase, the network slice instance is terminated and does not exist anymore.

The main steps for VNF instance termination are:

1. NFVO receives a request to terminate an existing VNF instance using the operation Terminate VNF of the VNF Lifecycle Management interface.
2. NFVO validates the request. It verifies the validity of the request (including sender's authorization) and verifies that the VNF instance exists.
3. NFVO calls VNF Manager to terminate the VNF using the operation Terminate VNF of the VNF Lifecycle Management interface.
4. VNF Manager terminates the VNF. This step may include a graceful shutdown of the VNF possibly in coordination with other management entities or the VNF itself.
5. Once the VNF is terminated, VNF Manager acknowledges the termination back to the NFVO.
6. NFVO requests the deletion of resources (compute, storage, and network) used by the VNF instance.
7. VIM deletes the internal connectivity network.
8. VIM deletes the compute and storage resources of the VNF instance.
9. NFVO acknowledges the completion of the VNF instance termination.

4. Integration guidelines

4.1. Introduction

Software and hardware development can be a very complex task. In order to ensure code quality, to test the developed product and to track the changes made by different developers, Integration Guidelines (IG) are needed.

Example of these guidelines could be a standard to be followed, in order to ensure that the code is readable and understandable by different developers involved in the process.

For example, how to enable in an easy way track changes in the software and repositories where these software are used and where the different versions committed by the developers are stored. In the following sections, guidelines for code development are discussed.

4.2. Software

4.2.1. Conventions

Smart5Grid proposes "behavioural" guidelines/conventions to be respected during both the development and integration phases. In the following, design patterns and code comments are proposed for facilitating communication among the developing teams.

4.2.2. Development phase

Since software development will be conducted by geographically dispersed distributed teams of the project Consortium, efficient communication among them as well as comprehension of affected or affecting code branches are vital for optimizing integration in terms of time and effectiveness.

With this aim, development guidelines constitute a set of rules, which will ensure consistent and standardized coding practices.

4.2.3. Patterns

Design patterns have been derived as general reusable solutions to commonly take care of problems within a given context. The design pattern concept was introduced by the architect Christopher Alexander in the field of architecture. However, the concept has been adapted and applied in other fields as well, including software design. Software design patterns do not imply designs which can be directly transformed into code. Rather, they constitute a formal way of documenting the solution to a known specific problem.

Developers within Smart5Grid are encouraged to use design patterns, where applicable. Through design patterns developers can have a common standard terminology for a common problem, facilitating communication among them. Moreover, specified as common solutions, design patterns have been evolved by developers over time, so that they describe best practices in common diagnosed problems.

4.2.4. Code comments

Code documentation is highly recommended because it is beneficial for both the developer developing a specific piece of software, but also for the ones revising or enhancing it or wishing to interface with it. Since this process will typically take place significantly later after the time of development, even the initial

developers often find difficulty in instantly identifying their own code functionality, often costing significant time and effort to recover memories.

This bottleneck can be overcome by including short comments throughout the code explaining functionalities. Developers are encouraged to take into account the following principles while documenting their code:

- Comments should precede the code they refer to. During the course of development, additions and modifications on the existing code could result in displacement of comments with regard to the referred code. Developers should take care of their code structure and their comments placement, after code updates.
- Comments should be consistent with the code they refer to. As code gets modified, comments are often not updated accordingly, so that they could describe different functionalities or parameters than the ones eventually developed or used. Thus developers are encouraged to provide consistent documentation of their code.
- Comments should be short and concise, so that their effectiveness and readability does not get compromised
- Comments should consist of the author name and date, so that potential incompatibilities can be reported to the original developers.

4.3. Conventions related to integration

In this section, guidelines facilitating integration activities are outlined.

4.3.1. Respect over Interfaces and Data Models

A major principle for effective code integration relates to interfaces and data models. Data models define classes, parameters and allowed methods, while the same classes can be used by several developers during the development of a system framework, like Smart5Grid. On the other hand, interfaces specify the communication among software components, defining the information flow, the interacting entities, their role in this interaction and the way this interaction is realized. In the same way, well-defined data model and interfaces specifications may facilitate and actually enact components interaction, while not respected specifications may result in significant incompatibilities and lack of interaction. As Smart5Grid progress will be realized in an evolutionary manner, enhancements and modifications over existing code will necessarily take place. As a result, data models and interfaces may need to be modified as well. Developers are required to respect specified interactions, data models and processes, to the extent possible. In the case of mandatory changes over specified processes developers are required to concretely specify the changes required, communicate them to the developing teams and include them in the revised version of the Smart5Grid data model or architecture, if there is one to be delivered.

4.3.2. Source Commits

When software development takes place in geographically dispersed places by different developing teams, a common code repository can be of great help, as a means of collecting different pieces of code and gradually integrating individual software parts. This way, distant teams can be updated with the work conducted with other teams, identify points and potential problems of interaction and thus have a better picture of the code they develop as a whole. Although source control appears as a greatly helpful solution for distributed software development, its benefits may be burdened by improper use of the

characteristics offered by the source code management platform. First, serious integration problems may arise, when broken code is committed. Interaction with other pieces of software requires that the already committed code is working and functional.

Thus, Smart5Grid developers should test their code before uploading it in the common repository.

Moreover, effective communication among different developing teams is achieved only if code commits take place on a frequent and systematic basis. Small pieces of code can be easily reviewed and integrated, while big pieces of code committed at once are harder to manipulate.

As a result, Smart5Grid developers should commit their code on a frequent and meaningful basis, providing the other developing teams with basic ideas and interactions of their solution. Software commits should start from basic API skeletons and gradually construct full-functional code parts.

4.4. CI/CD

CI/CD are two acronyms, where CI stands for Continuous Integration (CI), and the CD part again refers to Continuous Delivery and Continuous Deployment. CI/CD is relatively a new concept and has

gained a lot of popularity in the past few years.

CI/CD is a way or method of developing software which enable to deliver software applications introducing by design automation into the stages of software development.

Generally speaking CI/CD is about streamlined automation. So it can be said that implementing CI/CD we try to automate every possible stage of a software development cycle, stages like integrating the code, testing, building, deploying. This approach pushes the project that adopt CI/CD to minimizing human intervention so to minimize risk of human errors, which eventually increases the overall development speed.

Developers can make a code change live in production and the application will be smoothly deployed. CI/CD is not some tool or particular technology rather it is a methodology and operating principles that need to be adopted to automate and speed up the development and deployment cycle.

CI/CD is just a collection of standards and practices that needs to be followed by teams

that enables them to deliver code changes more frequently and reliably.

4.4.1. Continuous Integration in Smart5Grid

Continuous Integration (CI) refers to having a framework allowing developers to work on both differentiated branches of code and on a core (master) branch, effectively referring to the integration of code into a known or working code base. All source-code related actions and control operations (e.g. branching requests, review requests, merge requests, unit testing procedures, software package building etc.) are related to CI, as well. CI functionality are generally exposed by advanced source code management tools like Git, Subversion, Mercurial etc, Git being the most prominent and widely used of all. However, nowadays, there exist multiple platforms offering added value services on top of these source code management tools, such as Github, GitLab, Bitbucket etc. Continuous Delivery (CD) refers to the automated process of delivering a complete software package or service to an environment, be it integrated testing or production. In other words, it refers to controlling the deployment of the software

package or service to designated target application containers or servers for immediate consumption. Most of the aforementioned source code management platforms also provide the ability to integrate CD technologies either as third party (e.g. use Github combined with Jenkins or Travis) or integrated (e.g. GitLab with GitLab-CI) services.

Based on the above definitions, CI/CD refers to integrating the entire source code management and service delivery processes into one single pipeline that would cater for the deployment of a software package or service whose code has been mutually accepted by either automated routines (e.g. related to testing) or human-based activities (e.g. code review requests, branch merge requests etc.)

In the above context, testing plays a critical role as to the quality of the service being delivered in the end of the process. Three main types of testing are defined in general:

- Unit testing refers to testing the behaviour of the smallest part of software design, like functions. The goal is to identify code-related issues, pertaining to the syntactical correctness of the code and the rationality of the produced outcomes.
- Integration testing combines several software components (already unit-tested), such as libraries and modules, and test them as a whole. It ensures certain subsets of the system work as expected, from an interworking perspective
- System testing focuses on the user's or customer's perspective, checking whether the overall system meets the given specification.

Figure 4-1 : CI/CD provides an overview of an integrated CI/CD framework:

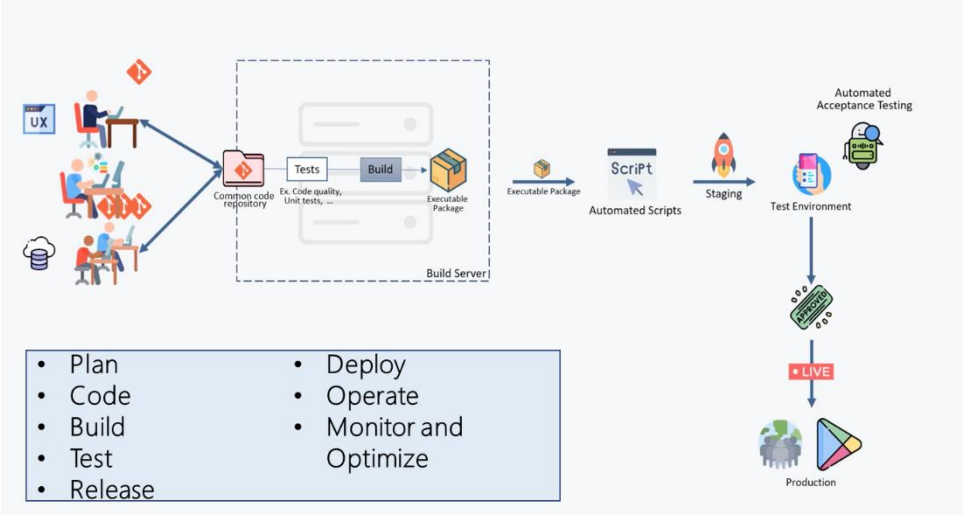


Figure 4-1 : CI/CD

Note that the CI part practically ends on the Automated Acceptance tests; thereafter, the CD part takes place. Completing the automated testing and quality enforcement status quo of the CI/CD framework, one may apply source code inspection through automation frameworks such as for example SonarCube[15]. Although code quality is not vital for the successful testing of a system or even its deployment, it can largely affect its efficiency in terms of resource allocation and/or performance, hence

[15] https://www.sonarqube.org/

it is, in general, recommended. Overall, the above approach could highly benefit our work as automated testing, though cumbersome and very time consuming, can assure that development over heterogeneous infrastructure and of different scope remain working and inter-working at any time. Under a CI/CD approach and considering the project orientation towards using OSM as base platform for NFV, the CI/CD approach would imply the following steps.

- The developer implements a certain piece of software that wishes to integrate to the core system functionality, irrespectively of whether this piece of code corresponds to the development of a VNF or a platform component as well as a NetApp.
- The developer introduces (or should have already introduced) a test case that check the relevant piece of code at the level of rationality checking and overall consistency.
- The developer commits the code to a dedicated development branch defined in the CI infrastructure.
- The developer initiates a request for code merging.
- The CI platform performs the determined unit testing.
- If the unit testing is successful, the code is subject to further integrated system testing.
    - If the integrated system testing is successful, then the merge request is accepted and the newly committed source code gets part of the code master branch.
    - Otherwise, the merge request is rejected.
- If either the unit or integrated system testing fails, the pull request is rejected. In this case, the developer should consider either a code rewrite to satisfy the unit testing procedures, or the unit testing rewrite (to avoid situations of erroneous unit testing. The workflow should be re-initiated from the beginning.
- The source code management platform move to the CD part, building the package and deploying it.

After successful CD, the new code is running on the deployment infrastructure and is subject to user testing/production.

4.5. Testing and Integration of the different components

4.5.1. Integration testing

Integration testing occurs after unit testing. The integration testing takes several modules or units, combines them into larger groups, sometimes called aggregates. The testers have already tested the individual components and the main purpose of the integration testing is to test the groups of units to see whether they properly interwork. The interfaces between units are tested without neglecting integration of the components and the hardware, the components and the operation system, or the components and the interface of any external system or application is tested as well. There are four approaches towards effective integration testing:

- Bottom-up integration testing
- Top-down integration testing
- Big-bang integration testing
- Sandwich integration testing

Big-bang testing: This approach is usually employed when the target of the test is the system as a whole; most of the components, if not all them, are tested at once. It is time-saving method in integration testing but there could stay some errors or defects untracked since not all the interfaces between the components are tested. The executed test cases and their results and outputs must be recorded properly to make the integration testing process clean and tabular.

Top-Down testing: This testing approach tests the highest-level modules first and then lower level modules are tested step by step, until the tests hit the smallest possible division. This approach is usually chosen when it comes after top-down development of the system. Stubs are important in the top-down approach; a stub is a program or a method that simulates some functionality of lower-level modules. Since in top-down integration testing a higher-level is tested first, sometimes they need functionality of the lower-level modules which are not prepared for testing or which are not developed yet. The stub can provide an expected response to a request, or simulate some activity which is requested by the higher-level module.

Bottom-up testing: This is an approach to integration testing where the lowest-level components are tested first, then they are combined and used for the testing of the higher-level components. The process is repeated until the component at the top of the hierarchy is tested. This approach helps in determining the completeness of software development processes, especially the percentage progress. When all the modules from the lower layer are ready, they get integrated and tested. To efficiently apply this testing approach, all components should be ready and prepared for the integration approximately at the same time. In such cases, a driver is necessary. Similar to a "reverse" stub, the driver is a method that passes some test cases to the lower-level modules or subsystems. This means that the driver simulates the higher integration layer that might not be implemented

Sandwich testing: This approach combines top-down testing with bottom-up testing. There are no strict rules where the developers should start with development. This creates a system with missing components at various levels. Stubs and drivers are used where needed in order to allow test execution. Usually, less stubs and drivers development is required compared to pure top-down and bottom-up testing methods.

The different test cases that will be documented in the next documents will use a template sheet as shown in the Figure 4-2:

Name	The test case code which is unique to the project	Locations	The places where the test will be executed
Component under test	The component under the test	Responsible	The partners responsible
Product Requirement		The requirement validated by the integration or test case	
Test environment		List of elements needed fort the test execution	
Feature under test		List of the features to be integrated	
Preparation		Pre conditions that the test environment has to meet before test execution	
Dependencies		Eventual dependencies need to be verified before the test	
Steps		Test procedures list of action needed.	
Pass Criteria		Expected measurable results, allowing to judge if the test is passed or not	

Figure 4-2 : Integration template

4.6. Smart5Grid platform integration

4.6.1. Interaction between OSR and V&V

This interaction concerns the OSR requesting the V&V to perform a validation or verification request and retrieving the test result. Both the requests (validation and verification) require the same steps to take place.

Data flow	Connection type	API protocol	Data type	Comments
OSR -> V&V	TCP/IP	HTTPS REST	JSON	The OSR requests V&V test
OSR -> V&V	TCP/IP	HTTPS REST	JSON	The OSR requests V&V test result

Figure 4-3 - Data flow

The testing agent asks the OSR to initiate a validation or verification of a predefined NetApp. The OSR sends a POST request to the V&V containing in the body of the request the necessary NetApp information in JSON format. The V&V generates a test instance ID and stores it along with the test status as "pending". Then the V&V initiates the process requested and sends back to the OSR the test ID in JSON format and a successful HTTP response code (200) if the test was successfully initiated or if failed, an error code (5##) according to the error encountered by the V&V. The OSR stores the test ID as the last performed test linked to the NetApp. Then in an asynchronous way, the V&V continues to wait for the test to complete and when it is finished the V&V internally stores the results in the Results Manager component.

As a second step, the testing agent asks the OSR to retrieve the test results using the test ID generated in the previous step. The OSR sends a GET request to the V&V sending the test id as a URL parameter. The V&V returns the test status and result in JSON format. If the test is not yet completed the result field is

empty. In an automated testing process, this step can be placed in a loop with the exit condition requiring the V&V test status to be in a different state than "pending" or a certain time threshold is reached.

Installation details

The integration testing process requires both platforms and a testing agent to be deployed. The OSR and the V&V must be reachable via a network with each other and the OSR must be reachable by the testing agent. This test case does not require a use case environment to be present, so the V&V test results can be artificially generated to cover all the possible expected outcomes. At least two predefined NetApps per V&V methods (validation, verification) must be registered in the OSR prior to the execution of this test, one should pass the V&V test and the other should fail it. The testing agent should accept the definition of tests. The test definitions shall include the expected outcome of the test (e.g. "failed" for the NetApp destined to fail the V&V test) and the NetApp ID in the OSR.

In the table below a road map for the integrations test is presented

Table 4-1 : Time schedule

Action	Kind of test	Start Date	End date
First integration round	Manual testing of API interactions involved	M18	M19
End-to-end automated testing	Automated testing with the use of a testing agent to verify the expected results on all possible interactions	M19	M24

5. Pre-piloting /Hardware in the loop /Energy Infrastructure

The pre-piloting testing phase refers to the testing of the use-case specific NetApps from the point of view of ensuring the necessary quality and functionality of the energy services they were designed for. Specifically, in this pre-piloting testing phase, the minimum quality and communication network requirements of the NetApps developed specifically for Use Case 3 and Use Case 4, will be investigated. The validation tests will follow in detail the full list of business specifications (energy related service(s)) that the use-case specific NetApps shall satisfy. In this framework, a sensitivity analysis for different parameters of the communication network infrastructure that could impact the performance of the NetApps and consequently affect the operation of the energy infrastructure will be performed. As an example, if the E2E latency requested by a NetApp and/or VNF implementing a virtual Phasor Data Concentrator (vPDC) energy-specific service, as extracted from the specified NetApp (or VNF) descriptor, is 40 ms (the baseline scenario), then different E2E latency scenarios will be executed ranging from 10 ms to 400 ms. It should be noted that the base-line network specific KPIs for the NetApps that will be tested in the pre-piloting phase could be retrieved from the V&V Results database as specified in Section 2.1.3, or can be simply specified by the telco operators involved in the use case implementation.

The implementation of the pre-piloting tests will be done by using a Real-Time Hardware-In-the-Loop (RT-HIL) infrastructure for power systems, where a hardware network emulator will be properly integrated. This infrastructure is demonstrated in Figure 5-1 and consists of the following components:

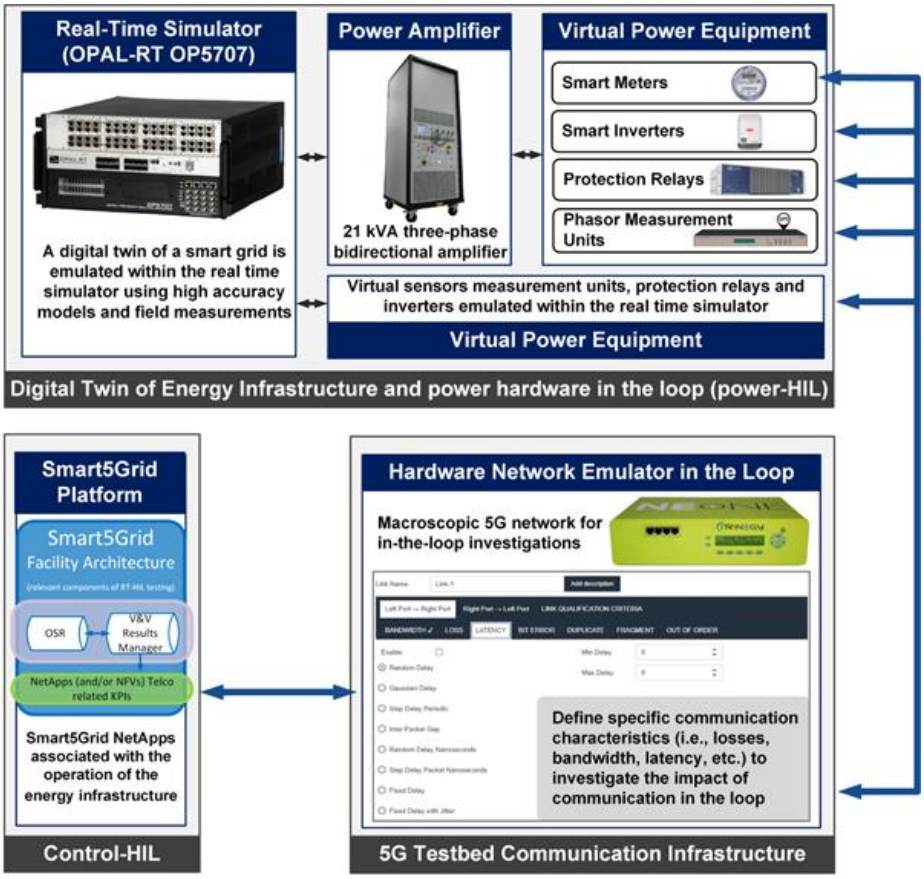


Figure 5-1 - Pre-piloting testbed facility.

Digital twin of the energy infrastructure and power-hardware-in-the-loop (power-HIL). In the context of the NetApps pre-piloting tests, the role of this block is to develop and implement accurate models of the relevant section of the power grid to be monitored or controlled by the energy service provided by the use-case specific NetApps. As an example, for the UC#4, where Phasor Measurement Units (PMUs) are involved to monitor a section of a cross-border transmission grid, a testbed power system can be implemented in the real time simulator for replicating a similar power system topology of the specific Use Case. It should be noted that the actual parameters and topology of the cross-border transmission line are confidential data and it is difficult to be provided by the operators for the pre-piloting phase. This is the main reason that testbed systems will be used for the testing of the different NetApps that are related to the energy infrastructure. In this use case, the actual PMUs will be connected with the digital twin of the power testbed through a power amplifier, integrating a power hardware in the loop (power-HIL) framework, and then synchronized measurements will be sent to the use-case specific NetApp using industrial communication protocol (I.e., IEEE C37.118 [14]). In an alternative way, virtual PMUs can also be implemented within the real time simulator, emulating the actual PMUs, and the measurements will be sent using the exact same protocol. For the case of the UC#3, this module will implement accurate models of the wind turbine(s) to be monitored and for which the predictive-maintenance services will be integrated in their use-case specific NetApps. Within this testing framework, a power test system will be used to create the digital twin of the energy infrastructure where the inverters of wind turbine systems can be either virtualized within the real time simulator or replicated through a commercial inverter (with lower power ratings in the range of 5-15 kVA) connected in the loop with the digital twin through a power amplifier (power-HIL). In general, through the digital twin and hardware in the loop framework different kind of virtual or actual equipment (used withing the smart grid context) can be integrated and tested in realistic conditions.

Control-Hardware In the Loop (Control-HIL). In the context of the NetApps testing from the energy service perspective, the role of this block is to offer a development space to integrate and test the use-case specific NetApps that are related to the energy infrastructure operation in a realistic and relevant environment. The control-HIL framework allows the real-time interaction between the NetApps of the Smart5Grid platform and the digital twin of a power system. In this context, specific NetApps can be integrated and validate their control performance and evaluate their impact on the operation of the energy infrastructure.

Hardware network emulator. This is in principle a software-defined hardware-based network emulator (NE) which mimics the effects of real-world networks, including 5G technology. The major functionality of the NE in the case of RT-HIL pre-piloting testing is to provide the macroscopic environment for the 5G telco infrastructure as it was defined in the Smart5Grid Architecture Facility. The NE will be integrated in the loop between the digital twin of a smart grid (where virtual or actual power equipment is connected through P-HIL) and the Smart5Grid platform (where the NetApps will be integrated using control-HIL) in order to incorporate the 5G communication characteristics in the validation environment. The pre-piloting testing environment aims to investigate how the performance of the communication infrastructure (in terms of macroscopic operational characteristics such as latency, bandwidth, losses, bit-error, etc.) might affect the quality of the energy services enabled by the use-case specific NetApps, following a sensitivity analysis in a macroscopic approach. In example, if the end-to-end (E2E) latency of a use-case specific NetApp is 40ms (e.g. the vPDC NetApp of UC#4, after it was validated by the V&V

framework, and stored in the Results Database KPIs – see Section 2.2.3), then several other E2E latency scenarios will be implemented through the NE (e.g. 50ms, 100ms, 200ms etc) to investigate the latency impact in the operation of the power system. . Furthermore, other communication scenarios will also be investigated, based on realistic operation macroscopic characteristics to be decided based on expert-knowledge models (elaborated using historical 5G test results, e.g., from OTE in Greece) obtained from other network services requesting similar KPIs with the specific NetApp under test

6. Conclusion and next steps

In this deliverable, the design and the specification of the different modules and components that compose the Smart5Grid platform and more in general the entire Smart5Grid ecosystem has been provided. In particular has been defined the peculiarities of the V&V framework and the Open Service Repositories that support and allow the operation phases of the NetApp E2E service.

This first deliverable of the WP3 in fact documents the interfaces needed to communicate between the Open Service Repositories and the V&V framework and how the NetApp Controller and the Slice Manager interact each other. Another important topic presented is the CI/CD strategy and how this could be exploited to the main goal to provide the seamless integration of the different components and modules of the Smart5Grid platform.

The next steps to be accomplished will see the coverage of the technical implementation of the final version of the OSR as well as the testing activities of HIL so to allow the global and final report about the work done on T3.1 & T3.2

7. References

[1] Smart5Grid Project (2021). Smart5Grid deliverable D2.1 "Elaboration of Use Cases and System Requirements". Available at: https://smart5grid.eu/wp-content/uploads/2021/07/Smart5Grid_D2.1_Elaboration-of-UCs-and-System-Requirements-Analysis_V1.0.pdf

[2] Smart5Grid deliverable D2.2 ""Overall architecture, design, technical specifications and technology enablers" Available at: https://smart5grid.eu/wp-content/uploads/2021/11/Smart5Grid_WP2_D2.2_V1.0.pdf

[3] ETSI GS NFV 002 V1.2.1 (2014-12) - Network Functions Virtualisation (NFV); Architectural Framework Available online: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf

[4] 3GPP, "Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); Stage 2 (Release 15), 3GPP TS 23.501 V15.12.0," 2020.

[5] https://www.5gradio.com/5g-networks/5g-gnodeb-base-station/

[6] https://www.nokia.com/about-us/newsroom/articles/open-ran-explained/

[7] https://www.zyxel.com/products_services/5G-New-Radio-Outdoor-Router-NR7101/overview

[8] https://www.winncom.com/docs/ericsson/Ericsson_Radio_4408_Datasheet.pdf

[9] https://docs.confluent.io/platform/current/platform.html

[10] https://www.etsi.org/deliver/etsi_gs/nfv-man/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf

[11] https://www.etsi.org/deliver/etsi_gr/NFV-IFA/001_099/029/03.03.01_60/gr_NFV-IFA029v030301p.pdf

[12] https://www.etsi.org/deliver/etsi_gr/NFV-REL/001_099/011/01.01.01_60/gr_NFV-REL011v010101p.pdf

[13] https://www.etsi.org/deliver/etsi_gr/NFV-IFA/001_099/029/03.03.01_60/gr_NFV-IFA029v030301p.pdf

[14] https://standards.ieee.org/standard/C37_118_1-2011.html