

780315 — SEMIOTICS — H2020-IOT-2016-2017/H2020-IOT-2017





SEMIoTICS

Deliverable D5.10 Demonstration and validation of SARA-Health Use Case (Cycle 2)

Deliverable release date	29.12.2020	
Authors	 Domenico Presenza, Philip Wright (ENG) Emmanouil Michalodimitrakis, Nikolaos Petroulakis, Thodoris Giakoumakis, Dimitris Mavroeidhs (FORTH) Felix Klement, Korbinian Spielvogel, Henrich C. Pöhls (UP), Bartłomiej Lipa, Michal Rubaj, Jakub Rola, Mateusz Kamiński (BS), Jordi Serra (CTTC) 	
Responsible person	Domenico Presenza (ENG)	
Reviewed by	Felix Klement (UP), Nikolaos Petroulakis (FORTH)	
Approved by	 PTC Members (Vivek Kulkarni, Nikolaos Petroulakis, Ermin Sakic, Mirko Falchetto, Domenico Presenza, Verikoukis Christos) PCC Members (Vivek Kulkarni, Nikolaos Petroulakis, Verikoukis Christos, Georgios Spanoudakis, Domenico Presenza, Danilo Pau, Joachim Posegga, Darek Dober, Kostas Ramantas, Ulrich Hansen) 	
Status of the Document	Final	
Version	1.0	
Dissemination level	Public	



Table of Contents

1.	Introd	uction	6
	1.1.	Methodology and document structure	7
2.	Use C	ase Description	8
	2.1.	Generic storylines	9
	2.1.1.	Fall Management Assistive Task	9
	2.1.2.	Remote Gait Analysis Assistive Task	10
	2.2.	Challenges and objectives	11
	2.3.	SEMIoTICS sub use cases	12
3.	Sub U	lse Case 1: SEMIoTICS-enabled Gait Analysis Web App	14
	3.1.	Scope and Objectives	14
	3.2.	Interaction with SEMIoTICS framework	14
	3.3.	Setup testbed and integration	19
4.	Sub U	se Case 2: Automated, trustworthy healthcare connectivity	23
	4.1.	Scope and Objectives	23
	4.2.	Interaction with SEMIoTICS framework	25
	4.2.1.	Interaction with NFV	25
	4.2.2.	NFV Orchestrator and VNF manager	25
	4.2.3.	Virtual Infrastructure Management	26
	4.2.4.	Pattern Orchestrator	26
	4.2.5.	Pattern Engine	26
	4.2.6.	Thing Directory	26
	4.2.7.	Backend Semantic Validator	26
	4.2.8.	GUI	26
	4.3.	Setup testbed and integration	27
	4.3.1.	NFV testbed	27
	4.3.2.	Service Function Chaining Patterns Validation	30
	4.3.3.	Backend Semantic Validator and Thing Directory Validation	40
5. se	Sub U Insitive c	se Case 3: Enforcement and Monitoring of GDPR-compliant access Control to confidential and lata	44
	5.1.	Scope and Objectives	44
	5.2.	Interaction with SEMIoTICS framework	45
	5.2.1.	Authorization Policy Management	45
	5.2.2.	Authentication Management	46
	5.3.	Setup testbed and integration	47
	5.3.1.	Setup steps	48
	5.3.2.	Enforcing access control	48



5.3.3	3. Dynamic Policy Adaptation	
5.3.4	4. Monitoring compliance	
5.3.5	5. Attribute-based encryption	
6. Sub	Use Case 4: Controlling bulbs and robots	53
6.1.	Scope and Objectives	53
6.2.	Interaction with SEMIoTICS framework	54
6.3.	Setup testbed and integration	54
6.3.1	1. Virtualization of The Robotic Rollator	54
6.3.2	2. Virtualization of ZigBEE Light bulbs	57
6.3.2	2. Virtualization of SoftBank's Pepper Robot	57
7. Sub	Use Case 5: Monitoring Activities of Daily Life	60
7.1.	Scope and Objectives	60
7.2.	Interaction with SEMIoTICS framework	61
7.3.	Setup testbed and integration	62
8. Over	rall KPIs and Requirements (ALL)	67
8.1.	Related KPIs	67
8.2.	Related framework requirements	
9. Con	clusion	92
9.1.	Observed Obstacles	92
9.2.	Concluding Remarks	92
10. Refe	erences	93
Appendix	A - Network Infrastructure Validation	94



Acronyms Table			
Acronym	Definition		
AE	Auto Encoder		
AI	Artificial Intelligence		
AWS	Amazon Web Services		
BPTT	Back Propagation Through Time		
CEP	Complex Event Processor		
CGNN	Causal Generative Neural Network		
DoA	Description of Action		
eBPF	extended Berkley Packet Filter		
ETC	Event Triggered Causality		
FPR	False Positive Rate		
GAN	Generative Adversarial Network		
GUI	Graphical User Interface		
ΙΙοΤ	Industrial Internet of Things		
ют	Internet of Things		
КРІ	Key Performance Indicator		
LLDP	Link Layer Discovery Protocol		
LOOCV	leave-out-one-device cross validation		
LSTM	Long-Short-Term-Memory network		
МС	Monitoring Component		
MCU	Micro Controller Unit		
ML	Machine Learning		
NMS	Network Management System		
NSGI	Next Generation Service Interface		
ODL	OpenDayLight		
OF	OpenFlow		
ОМР	Orthogonal Matching Pursuit algorithm		
PaaS	Platform-as-a-Service		



РО	Pattern Orchestrator
QoS	Quality of Service
RC	Recipe Cooker
RNN	Recurrent Neural Network
SDK	Software Development Kit
SDN	Software Defined Network
SNMP	Simple Network Management Protocol
SNS	Simple Notification Service
SPDI	Security, Privacy, Dependability, Interoperability
SR	Sparse Representation
SVDD	Support Vector Data Description
TPR	True Positive Rate
VM	Virtual Machine
WoT	Web of Thing



1. INTRODUCTION

This deliverable provides the validation of the SEMIoTICS technologies in the context of the development of a solution in the domain of Ambient assisted living.

The aims of SARA, this is the name of the solution, is to integrate a number of assistive technologies to address the diverse problems that older people face when living independently with the support of their community.

In particular, the SARA solution is being developed around the concept of Assistive Task (AT). An Assistive Task is a coherent set of SARA functionalities addressing a specific risk related to the independent living of elderly people affected by Mild Cognitive Impairment (MIC) or mild Alzheimer's disease (AD).

The Assistive Tasks being developed in SARA address risks related to four areas: physical decline, cognitive decline, medication management, and psychological needs.

We selected two assistive tasks to validate how and to what extent the SEMIoTICS technologies can facilitate the implementation of a system like SARA. The selected ATs are the Fall Management and the Remote Gait Analysis. The *Falls Management Assistive Task* aims to reduce the consequences of fall events of elderly living at home. The *Gait Analysis Assistive Task* aims to support remote gait quality assessment by means of specific exercises executed by the patient using a Robotic Rollator and under the remote supervision of a Doctor.

The implementation of each SARA Assistive Task requires the integration of multiple mobile (mobotic) and static (domotic, or home automation) robotic elements working in tandem with wearable health monitors, personal smart devices (phones and tablets) and various (proprietary and third party) computational and information services. All these devices that are not specifically designed to work together.

This integration effort requires to cope with a number of challenges arising in each of the main areas addressed by the SEMIOTICS project:

- security, privacy, dependability and safety: all stored personal data must be securely encrypted with
 protected access only by authorized and authenticated individuals and/or system processes; all
 communications with personal data as content must be securely encrypted with authenticated endpoints;
 the treatment of patient health data must comply with healthcare domain standards and national and
 international regulations concerning security and privacy; all actuator control signals (whether originating
 'on device 'or 'over the network') must be protected from malicious interference;
- semantic interoperability: the different devices consume and produce different kinds and quantities of data, in diverse formats, and support different modes of network communication (Bluetooth, Wi-Fi, wired ethernet, can-bus, etc.). Moreover, for reliability, critical data is collated from multiple redundant sources (several components monitoring the same features) and shared through multiple redundant channels;
- embedded intelligence: to ensure timely responses of robotic elements, AI intense computations are executed concurrently throughout the system, with operations dynamically dispatched to the nearest available computing resources (including edge devices);
- trustworthy network management: system components and technologies must be configured on a
 reliable network to avoid service interruptions; high-priority communications, e.g. incident alerts, must be
 transmitted by reliable means (e.g. multiple identical transmissions over multiple channels) and given
 network priority; network configurations are not static: mobile devices move around and may temporarily
 go out of range of Wi-Fi; mains powered devices (e.g. Wi-Fi base stations, domotic servers) are lost
 during power outages, while battery powered devices (e.g. robots and phones) will need periodic
 recharging; and wearable health monitors can be removed (e.g. when taking a bath).

In this document, the validation of the benefits brought by the SEMIoTICS technologies to the development and operation of SARA is presented by means of five sub use cases. Each sub use case focuses a subset of the challenges in one the areas presented and represents a different view on the very same system. The five sub use cases are:



- Moving Intelligence to the Edge demonstrating how SEMIoTICS Local Embedded Intelligence can be deployed on the SARA Robotic Rollator and utilized in the implementation of the SARA Remote Gait Analysis Assistive Task.
- Automated, trustworthy healthcare connectivity show how NFV along with the pattern related components, offer a flexible solution to deal with the heterogeneous traffic flows of SARA. Also, how they reduce the manual intervention in the networking configuration and management, thus yielding a network management automation solution.
- Enforcement and Monitoring of GDPR-compliant access Control to confidential and sensitive data presenting how SEMIOTICS can be used to address security, privacy, dependability and safety challenges.
- Controlling bulbs and robots demonstrates how the SEMIoTICS Semantic Interoperability technologies can help the developers of the SARA solution to interface heterogeneous devices ranging from light bulbs to semi-autonomous robots.
- **Monitoring Activities of Daily Life** show how the SEMIoTICS Monitoring, Prediction and Diagnosis Mechanisms (MPD) component developed in the context of Task 4.2, can be used to support the Human-Robot Interaction in the context of the SARA Remote Gait Analysis Assistive Task.

1.1. Methodology and document structure

The overall development of SARA in SEMIoTICS proceeded in three phases:

- the initial phase took place during the first year of the project and concerned the establishment of both business requirements and requirements of SARA towards the SEMIoTICS framework.
- the second phase, during the second year of the project, saw the set-up of the initial infrastructure, the selection of the ATs to be used for the evaluation, and the development of a first version of the Fall Management AT without the use of SEMIoTICS technologies (still under development at that time). This development served to establish a baseline for the subsequent developments based on the SEMIoTICS technologies. This first implementation was shown during the first formal review of the project.
- the third phase started with the third year of the project and is concerning both the re-engineering of Fall Management AT and the development of the new Remote Gait Analysis AT. Both this development are based on the adoption of the SEMIoTICS technologies as illustrated by means of four sub use cases presented in the present deliverable.

This deliverable D5.10 accounts for the intermediate results achieved during the third phase of the development and is organised as follows: section 2 presents the Assistive Tasks selected for the evaluation of the SEMIoTICS technologies (section 2.1), the challenges subsumed by the implementation of selected ATs (section 2.2), and introduces the aspects addressed by each of the five sub use cases (section 2.3). Sections 3, 4, 5, 6 and 7 go in the details of each sub use case. Finally, the KPIs and requirement related to this use case are presented in Section 8.

This deliverable D5.10 is the result of the joint effort by all contributing partners during the development of the SARA prototype. At the beginning of the third phase bi-weekly meetings were run to keep aligned each partner about the work of the others, share results of experimentations and take design decisions. Later on the meetings were run weekly.



2. USE CASE DESCRIPTION

As described in Deliverable 2.2, the use case 2 is from the domain of Ambient Assisted Living and focuses on the development of the SARA distributed application running on the CLOE-IoT platform.

The global trend for increased life expectancy is expected to be accompanied by a rapid rise in the number of people affected by Mild Cognitive Impairment (MIC) or mild Alzheimer's disease (AD). Aging and dementia lead to a significant chronic loss in physical/cognitive capacities and a decline in functional ability, resulting in increased dependency and need for caregiving, with substantial medical, social, psychological, and financial burdens placed on patients, their families, and their communities. Indeed, the kind of continuous, holistic and integrated care required by people with dementia cannot be satisfied by the current 'specialist' model, and the gap between the number of people in need of care and those able to provide it is expected to grow. People also prefer to live in their own homes for as long as possible rather than being institutionalized in sheltered/nursery homes when age-related problems appear. Leading, for example, to propose "Aging in place" - allowing older adults to age in the least restrictive environment of their choice - as a more desirable and viable care model.

Home automation and robotics systems can help relieve the caregiving burden and make 'ageing in place' a reality by providing physical and cognitive assistance to the elderly in the common tasks of daily home living, helping them maintain their autonomy longer and delay the need for social/health service interventions. Advances in artificial intelligence (AI) and IoT technologies are rapidly converging to make such solutions both technically and economically feasible. In this light, Engineering is extending its current commercial AREAS® suite - an Enterprise Resource Planning solution for the Health sector - with SARA (Socially Assistive Robotic Solution for Ambient assisted living), a robotic system aimed at providing automated health/incident monitoring and home assistance to elderly people with mild cognitive impairments.

As its name indicates, SARA is a Socially Assistive Robotics (SAR) solution. SAR solutions resides at the intersection of Assistive Robotics (AR) - typically robots that provide physical assistance (e.g. rehabilitation robots, wheelchair robots, mobility aids, manipulator arms) - and Socially Interactive Robotics (SIR) - whose main task is some form of social interaction with humans (e.g. engaging in conversation, understanding gestures, role taking, task collaboration). SAR aims to assist humans through social interaction – an ambitious goal given that social interaction is arguably the most complex of all human behaviours. A socially competent robot, for example, should be able to: communicate with high-level dialog; use natural cues (gaze, gesture, etc.); express and/or perceive emotions; learn/recognise models of other agents; exhibit distinctive personality and character; establish and maintain social relationships; (possibly) learn/develop social competences.

Achieving human level competence in these areas is well beyond even the best of contemporary AI technologies, so with SARA our goal is more pragmatic. We aim to develop an expandable platform that initially exhibits only trivial social capabilities (well within current technological limits), while yet providing a solid foundation for future expansion to more sophisticated human-like interactions.

The design of SARA integrates a number of assistive technologies to address the diverse problems that older people face when living independently with the support of their community. More specifically, it comprises multiple mobile (mobotic) and static (domotic, or home automation) robotic elements working in tandem with wearable health monitors, personal smart devices (phones and tablets) and various (proprietary and third party) computational and information services, to continuously monitor and assist elderly subjects (and their caregivers) in their normal daily activities in and around the home. The technical components of SARA are as follows:

- backend AREAS Cloud Services (ACS): serving primarily as a repository of medical records and an offline store for monitored bio-medical data;
- a Body Area Network (BAN) comprising wearable health monitors (e.g. for heart-rate, blood pressure, breathing rate, stress levels, balance and fall detection) worn by the elderly subject and communicating via Bluetooth to their personal device (phone or tablet);
- a **Robotic Rollator** (RR): a smart wheeled walking frame providing physical support for standing, sitting and walking, and capable of monitoring its user's posture and gait, and of (limited) autonomous navigation;



- a Robotic Assistant (RA): a mobile humanoid robot specifically Softbank's humanoid "Pepper" robot¹
 capable of autonomous navigation, face recognition, scripted dialog and adopting life-like human poses (e.g. to perform arm/hand gestures);
- various **AI Services**: computational resources providing machine learning (ML), mapping, route planning and reasoning capabilities (among others);
- a Smart Home: a home incorporating intelligent objects, utilizing wired or wireless networks and having the capacity to analyse patterns of activities to manage appliances in anticipation of human needs or to provide adaptive cues for human occupants.



FIGURE 1: SARA MAIN TECHNICAL COMPONENTS AND COMMUNICATION SYSTEM

With the exception of the ACS and RR (both developed in house by Engineering) these components are all offthe-shelf devices that are not specifically designed to work together. The different devices consume and produce different kinds and quantities of data, in diverse formats, and support different modes of network communication (Bluetooth, Wi-Fi, wired ethernet, can-bus, etc.). Moreover: for reliability, critical data is collated from multiple redundant sources (several components monitoring the same features) and shared through multiple redundant channels; to ensure timely responses of robotic elements, Al intense computations are executed concurrently throughout the system, with operations dynamically dispatched to the nearest available computing resources (including edge devices); network configurations are not static: mobile devices move around and may temporarily go out of range of Wi-Fi; mains powered devices (e.g. Wi-Fi base stations, domotic servers) are lost during power outages, while battery powered devices (e.g. robots and phones) will need periodic recharging; and wearable health monitors (BAN devices) can be removed (e.g. when taking a bath).

The SARA solution is built on top of on the CloE-IoT platform. The CloE - IoT platform² aims to simplify the integration of highly distributed, complex and robust IoT solutions exploiting computational resources both in the cloud and at the edge.

2.1. Generic storylines

The SEMIOTICS technologies are validated in the context of a re-engineering process of the SARA prototype being developed by ENG. This re-engineering process concerns both the business logic layer (i.e. the assistive tasks provided by SARA to its end users) and the middleware layer (i.e. the CloE-IoT services).

At the business logic layer two assistive tasks are chosen: Fall Management and Gait Analysis.

2.1.1. FALL MANAGEMENT ASSISTIVE TASK

¹ <u>https://www.softbankrobotics.com/emea/en/robots/pepper</u>

² Staring from January 2020 the CloE-IoT platform is part of the Digital Enabler ecosystem. <u>https://www.eng.it/en/our-platforms-solutions/digital-enabler</u>



The Fall Management Assistive Task is one of the Assistive Task contributing to the fulfilment of requirements of the SARA Use case and described in deliverable D2.2 - "SEMIOTICS usage scenarios and requirements".

- The SARA system MUST (via SARA Hub devices) continuously monitor the Patient to detect, and raise an alert in case of, incidents (i.e. critical medical events such as falls, heart attacks, etc.).
- The SARA system MUST reliably and rapidly notify assigned Caregivers of Patient incidents.
- The SARA system MUST (for reasons of privacy) notify the Patient whenever a telepresence session has started/ended, and also inform the Patient of the identity of the remote operator.
- The SARA system MUST reliably and rapidly respond appropriately to health-critical events (in particular incident alerts).

The *Falls Management Assistive Task* aims to reduce the consequences of fall events of elderly living at home. In particular:

- 1. SARA continuously monitors the patient by means of the devices (e.g. smart phone/smart watch) within her Body Area Network;
- 2. whenever the fall detection software hosted by the BAN detects a possible fall, it raises (both via cellular and Internet connectivity) an early warning towards a remote assistance Call Centre. This warning message carries also information about the location/area where the event was detected. The fall detection software identifies the location of the event using iBeacon technology.
- 3. using the local Wi-Fi network the warning message is also forwarded to the Robotic Assistant (i.e. a Pepper robot in the current prototype).
- 4. as soon as the Robotic Assistant receives a fall event message it starts to navigate toward the location indicated in the message. The Robotic Assistant relies on the navigation and localization cloud services of the SARA solution to autonomously reach the location of the event.
- 5. once reached the location of the event the Robotic Assistant may interact with the Smart Environment to better prepare the environment for the telepresence session initiated by the Call Centre operator later on (see next steps in this scenario). As an example the Robotic Assistant might decide to turn on the lights or open the shades if on board sensors report a low level of enlightenment in the room.
- 6. the call distributor software operating at the Call Centre dispatches the message to one of the operators logged into the system. The call distributor software forwards to an operator only those messages originating from a certified device. Moreover the call distributor also filters out duplicated messages related to the same event.
- 7. the Operator Console web application retrieves from the Electronic Patient Healthcare record service of the AREAS® platform the relevant clinical data of the patient indicated in the warning message. The Operator Console web application presents, in a coherent way, both the information carried by the warning message and that retrieved from the Electronic Patient Health Record.
- 8. once all the relevant information has been presented, the Call Centre operator can assess the actual situation at patient's home starting a telepresence session with the Robotic Assistant using the cameras on board of it.
- 9. if the position reached autonomously by the Robotic Assistant is not satisfactory for an assessment of the scene of the event, the Call Centre operator can also initiate a teleoperation session to move the Robotic Assistant to a better point of view.

2.1.2. REMOTE GAIT ANALYSIS ASSISTIVE TASK

The Remote Gait Analysis Assistive Task is one of the Assistive Task contributing to the fulfilment of requirements of the SARA Use case and described in deliverable D2.2 - "SEMIOTICS usage scenarios and requirements"



- The SARA system MUST (via the SARA Hubs) continuously monitor the Patient's daily activities.
- The SARA system MUST assist the Patient in performing a variety of physical rehabilitation exercises.
- The SARA system MUST provide (via the SARA Hubs) the GP remote access to real-time (Hub monitored) Patient biomedical data.
- The SARA system SHOULD facilitate the integration of legacy and new IoT Devices (Hubs, Sensors & Actuators) e.g. through standardised low-level API conformance.

The *Gait Analysis Assistive Task* aims to support remote gait quality assessment by means of specific exercises executed by the patient using the Robotic Rollator and under the remote supervision of a Doctor. In particular:

- 1. The Doctor schedules the timeframe for the execution of the assessment session using the SARA Agenda service. As an example a Doctor could schedule a periodic assessment session to be executed during the third week of every month.
- 2. At the start of the period scheduled by the Doctor (e.g. the third week of the month) the SARA Agenda service notifies a remainder to the Robotic Assistant.
- 3. Once received the notification the Robotic Assistant autonomously navigates to the location where the patient is. For the navigation and localization the Robotic Assistant relies on the navigation and localization cloud services of the SARA solution (see also previous Fall Management scenario).
- 4. Once reached the patient the Robotic Assistant initiates a dialog with the patient to engage him/her in the execution of the gait assessment session. For the execution of the dialog the Robotic Assistant relies on the Dialog Manager cloud service of SARA and other third party AI services (e.g. speech to text).
- 5. If the patient accepts to do the gait assessment session a notification is sent to the Doctor. If the patient refuses, the session is rescheduled.
- 6. Upon the reception of the acceptance by the patient the Doctor starts a remote supervision session using the Gait Analysis web application from the SARA solution.
- 7. At the beginning of the session the Doctor starts a telepresence session using the camera on board of the Robotic Assistant. Using this session the Doctor can do a visual assessment of the gait of the patient. The doctor can use the Gait Analysis web application also to teleoperate the Robotic Assistant and move it to a better pint of view if the current one does not allow a clear view of the scene. The Doctor may also decide to keep a video recording of the execution of the exercises for subsequent analysis.
- 8. The patient executes the exercises as requested by the Doctor. During the execution of the Gait Encoder software hosted by the controller of the Robotic Rollator segments and encode the time series generated by various sensors embedded within the Robotic Rollator (e.g. IMU, Lidar, handlebars).
- 9. As the encoded time series are produced by Gait Encoder they are sent to the Gait Clustering cloud service part of the SARA solution.
- 10. The Gait Clustering service clusters the time series received by the Gait Encoder and presents the resulting clusters to the Doctor using the Gait Analysis web application. The Doctor uses this clusters to make his/her assessment about the condition of the patient. The Gait Clustering service also attaches the result of the clustering to the Health record of the patient.

2.2. Challenges and objectives

The smart end-to-end IoT interoperability, connectivity and security technologies developed by the SEMIoTICS project can play a critical role in addressing the key challenges entailed in the development of SARA.



In particular, already the implementation of the two assistive tasks described in the section 2.1 requires to address a number of challenges arising in the area of:

- End-to-end security, privacy, dependability and safety: the development and operation of SARA need means to accurately assess the impact of varying system configurations on the security, privacy, dependability and safety properties of the system: to verifiably ascertain, for example, if using a particular device for a particular purpose, or a particular configuration of devices, exposes the system to an unacceptable level of risk.
- IoT Semantics Interoperability: SARA system relies on a distributed network of sensors and actuators, • employing diverse wired and wireless communications technologies. The nodes in the network are highly heterogenous, ranging from basic inertial measurement units (IMU - e.g. for balance and fall detection), presence sensors (e.g. cameras equipped with motion detection and face/object recognition), to sophisticated robotic devices with their own internal sensor/actuator systems and onboard computers. The sensor/actuator network also communicates with backend cloud services to access and store events (e.g. 'fall detected', 'telepresence session initiated') and monitored health data (e.g. detailed timesequenced measurements from BAN devices). Accordingly, the system must deal with a wide range of device semantics (different kinds of devices with different functions), data formats (syntactic representations), measurement unit conventions (for sensor readings), and communications protocols (e.g. Wi-Fi, ZigBee, Bluetooth). Various key aspects of SARA functionality, moreover, require that data from multiple sources is collated, aggregated and/or analyzed in a coherent collective fashion. The reliable detection of 'fall incidents', for example, may involve the continuous comparative evaluation of data from wearable IMU devices, RR handle-mounted pressure sensors and RA video cameras (among others).
- Embedded Intelligence and local analytics: SARA is an application of Socially Assistive Robotics (SAR), and is thus fundamentally an artificial intelligence system. SARA requires the continuous operation of a range of perceptual systems. This means multiple AI/ML processes working concurrently to extract different features from the sensory input in a timely fashion - i.e. without any significant network latency. Thus there is a strong need for an IoT infrastructure that supports computationally intensive, distributed AI processing at the edge of the IoT network.
- Trustworthy Network Management: SARA is a distributed application where different computing entities continuously communicate and synchronize their activities to collectively generate the overall functionality of the system. Many of these distributed computations, due to their near real-time nature, are subject to strong time constraints which can only be met by a network infrastructure capable of guaranteeing trustworthy, low latency, reliable communications between all participating components. The SARA robotic nodes, moreover, have (quasi)autonomous behaviours driven to a large extent by uncertain and unpredictable events in the environment with a corresponding uncertainty and unpredictability in the generated computational load. Such variability prohibits the advance determination of optimal configurations of networking resources for trust guarantees. Indeed, dealing with such dynamic run-time variability entails adopting correspondingly dynamic and flexible mechanisms for trustworthy network management.

2.3. SEMIoTICS sub use cases

In this document the validation of the benefits brought by the SEMIoTICS technologies to the development and operation of SARA is presented by means of five sub use cases. Each sub use case focuses a subset of the challenges in one the areas presented in section 2.2 and represents a different view on the very same system. The five use cases are:

- Moving Intelligence to the Edge demonstrates how the time series clustering algorithm part of the SEMIoTICS Local Embedded Intelligence component developed in the context of Task 4.3, can be deployed on the SARA Robotic Rollator and utilized in the implementation of the SARA Remote Gait Analysis Assistive Task.
- Automated, trustworthy healthcare connectivity shows how NFV along with the pattern related components, offer a flexible solution to deal with the heterogeneous traffic flows of SARA. Also, how they



reduce the manual intervention in the networking configuration and management, thus yielding a network management automation solution.

- Enforcement and Monitoring of GDPR-compliant Access Control to Confidential and Sensitive Data shows how the interaction of the SEMIoTICS Security & Privacy components is able to address, realize and monitor the enforcement of all security- and privacy-relevant properties of user-related data. Especially in this sub use case, the data relates to the medical conditions and thus is privacy sensitive data. SEMIoTICS enforces the GDPR-compliant controlled access to that data by enforcing and monitoring access control through policies as well as through encryption to enable GDPR-compliant data handling also during transfer and storage even beyond the system's boundary.
- Controlling bulbs and robots demonstrates how the SEMIoTICS Semantic Interoperability technologies developed in the context of Task 3.3 can help the developers of the SARA solution to interface heterogeneous devices ranging from light bulbs to semi-autonomous robots while keeping the application logic independent from the underlying hardware and transport protocols as much as possible.
- **Monitoring Activities of Daily Life** shows how the SEMIoTICS Monitoring, Prediction and Diagnosis Mechanisms (MPD) component developed in the context of Task 4.2, can be used to support the Human-Robot Interaction in the context of the SARA Remote Gait Analysis Assistive Task.

The reader can find in Appendix A the mapping between the sub use cases introduced by this section and the requirements posed by the SARA-Health use case towards the SEMIoTICS framework. This set of requirement was initially presented in deliverable D2.3 - "Requirements specification of SEMIoTICS framework".



3. SUB USE CASE 1: SEMIOTICS-ENABLED GAIT ANALYSIS WEB APP

3.1. Scope and Objectives

This sub use case aims to demonstrate how various SEMIoTICS technologies developed in the context of WP3 and WP4 can help to implement one of the SARA functionalities: the Gait Analysis Assistive Task described in section 2.1.2. This sub use case represents an evolution of the sub use case 1 that was named "Moving intelligence to the edge".

In particular this sub use case aims to demonstrate how the SEMIoTICS Local Embedded Intelligence technologies (more specifically the time series clustering algorithm) developed in the context of Task 4.3 can help to reduce the amount of raw data required to be transmitted from field devices to cloud services.

The SARA Gait Analysis Web Application is the application utilized by a Doctor to remotely supervise a Gait Analysis examination session performed by a patient at home. The SARA Gait Analysis Web Application allows a Doctor to schedule gait analysis sessions, to run a telepresence session using a Pepper Robot, receive and store gait data collected by means of Robotic Rollator and, to review data after the end of a session.

The Figure 2 shows the page offered by the GA Web App to supervise a gait analysis session. The widget at the center of the page shows the video stream generated by the camera on board of robotic assistant (i.e. the Pepper robot). A Doctor can select the point of observation using the controls on the right part of the page: the buttons allow positioning the robot and fixing the posse of its head. The controls on the bottom part of the page allow start/stop recording the gait data collected by means of the Robotic Rollator.



FIGURE 2: THE SARA INTERFACE SUPPORTING THE SUPERVISION OF A REMOTE GA SESSION

3.2. Interaction with SEMIoTICS framework

Figure 3 shows the SEMIoTICS components involved during the runtime for the execution of the Gait Analysis AT.





FIGURE 3: SEMIOTICS COMPONENTS USED BY DOCTOR WEB APP

The Doctor Web App is a web application residing on the SARA Server. A Doctor can access the Doctor Web App across the Internet using a standard Web Browser.

The Doctor Web App uses the SEMIoTICS Security Manager to authenticate a user. The Doctor Web App relies on the Electronic Health Record from AREAS® to retrieve/store information about patients.

The SARA Gait Analysis Server offers to the Doctor Web App the operations to manage a Gait Analysis session.

Whenever a Doctor requests to start the recording of data, the Gait Analysis Server communicates with the Gait Sampler application hosted by the Robotic Rollator assigned to the patient.



The Gait Sampler uses the Rollator Servient to obtain the time series representing the distances of patient's legs from the back of the rollator (Figure 4). This operation is done putting an observation on the resources legR and legL exposed by the rollator servient (see also section 6.3.1). The Rollator Servient is the SARA component that virtualizes the Robotic Rollator as a "Thing" in the sense defined by the WoT standard.



FIGURE 4: AN EXAMPLE OF TIME SERIES RECEIVED BY THE GAIT SAMPLER

The Gait Sampler extracts the time series representing a single gait from the time series received from the servient (Figure 5). Each time series accounting for a single gait is encoded using the SEMIOTICS Time series encoder. The encoded time series are sent back to the Gait Analysis server via the TS Listener interface.



FIGURE 5: EXAMPLE OF TIME SERIES SENT TO THE TS ENCODER

Whenever a Doctor requests to stop the recording of data, the Gait Analysis Server uses the SEMIoTICS Timeseries clustering component to assign each buffered time series to cluster. The **TS Classifier** allows Gait Analysis Server to compute the probability that a gait cycle belongs to one of the possible predefined cluster (Figure 6)





FIGURE 6: AN EXAMPLE OF ASSIGNMENT PRODUCED BY THE TS CLASSIFIER

The result of the clustering is sent back to the Doctor Web App that presents it in the web interface of the Doctor.

At the end of a session the SARA Gait Analysis Server persists an encrypted version of the results of the clustering: The encryption is done using the SEMIOTICS Attribute-Based Encryption component.

The Figure 7 shows more precisely the interactions occurring between components during the execution of Gait Analysis Task.



FIGURE 7: INTERACTIONS BETWEEN COMPONENTS

In particular:

- 1. The Doctor Web Application requests the address of the patient's rollator to the Electronic Health Record (EHR).
- 2. The EHR return the rollator's address to the Doctor Web Application.
- 3. The Doctor Web Application requests to the Gait Analysis Server to start the sampling of the patient's gaits.
- 4. The Gait Analysis Server requests to the Gait Sampler to start sampling process.
- 5. The Gait Sampler requests to the Local Thing Directory (a SEMIOTICS component) the descriptor of the Rollator.
- 6. The Local Thing Directory returns the descriptor of the Rollator to the Gait Sampler component.
- 7. The Gait Sampler requests to the Rollator Servient to notify periodically the distance of the right leg.
- 8. The Gait Sampler requests to the Rollator Servient to notify periodically the distance of the left leg. Once both observations (left leg and right leg) are in place the Gait Sampler repeats continuously the steps 9-13 described below:
- 9. The Gait Sampler receives (asynchronously) from the Rollator Servient the distance of the left leg, or
- 10. the Gait Sampler receives (asynchronously) from the Rollator Servient the distance of the right leg.



- 11. The Gait Sampler, whenever detects that a gait cycle is complete, requests to the TS Encoder (a SEMIoTICS component) the encoding of the two times series accounting for the distance of the left and right legs from the rollator.
- 12. The Gait Sampler receives from the TS Encoder the encoded versions (latent representations) of the two time series.
- 13. The Gait Sampler forwards to the Gait Analysis Server the time series received by the TS Encoder.
- 14. The Gait Analysis Server requests to the TS Clustering component (a SEMIoTICS component) to assign to each time series the probability to be in one of the predefined gait clusters.

3.3. Setup testbed and integration

The tested of sub use case 1 utilizes three hosts (see also Figure 3):

- Doctor's workstation implemented by a standard PC running Window 10 OS.
- SARA server's software components are deployed both on a standard personal computer and on a virtual machine (running Linux OS) available from the FiwareLab³ node operated by ENG in its data center located in Vicenza (Italy).
- **Robotic Rollator Hub** implemented by a Raspberry Pi 3 running Raspbian OS. This host is the same host used by sub use case 4 (see section 6). The Robotic Rollator Hub is realized using a Raspberry Pi 3 which is a clone of the one on board of the actual Robotic Rollator. During the second half of the year it was required to rely on a clone to cope with the impossibility to access the actual Robotic Rollator because of the Covid-19 pandemics.

The components presented in Figure 3 are implemented as follows:

- **Doctor Frontend** is realized by a Chrome Web browser executing the JavaScript served by the Doctor Web App hosted by the SARA server. This script implements the GUI of the Doctor Web Application.
- Doctor Web App is implemented by a web application written in Angular 8 and Typescript running on a Tomcat web server and uses the angular-oauth2-oidc⁴ support to interact with the SEMIoTICS Security Manager instance operated by University of Passau⁵ (Figure 8).

³ https://www.fiware.org/developers/fiware-lab/

⁴ https://www.npmjs.com/package/angular-oauth2-oidc

⁵ http://semiotics-security.sec.uni-passau.de:3002/





FIGURE 8: THE AUTHENTICATION PAGE PRESENTED BY THE SEMIOTICS SECURITY MANAGER

- Electronic Health Record is realized by a database from AREAS®. The Electronic Health Record from the AREAS® is a development instance that does not contain any sensible data related to any real individual.
- **TS Clustering** is realized by the Keras library and the SEMIoTICS time series clustering model described in deliverable D4.10 and trained using data collected by means of the Robotic Rollator before the Covid-19 lockdown (i.e. same data used by sub use case 4 described in section 6).
- Gait Analysis Server is realized by a REST service developed in Python 3.8 making use of Flask framework⁶. The Gait Analysis Server uses the Keras 2.4.3 and Tensorflow 2.3.0 to load and execute the SEMIOTICS Clustering Model. The SEMIOTICS Clustering Model is deployed on the SARA server as an HDF5 (.h5) file.
- TS Encoder is realized by the Keras library and the SEMIoTICS time series encoder model described in deliverable D4.10 and trained using data collected by means of the Robotic Rollator before the Covid-19 lockdown (i.e. same data used by sub use case 4 described in section 6). Similarly to the Gait Analysis Server the time series encoder is rest service developed in Python 3.8 making use of Flask framework⁷. The Gait Sampler uses Tensorflow Lite runtime to exploit the Timeseries encoder model. The Robotic Rollator Hub hosts a Tensorflow Lite version of the regular Timeseries encoder model.

⁶ https://flask.palletsprojects.com/en/1.1.x/

⁷ https://flask.palletsprojects.com/en/1.1.x/





- **Rollator Servient** is the WoT servient virtualizing the Robotic Rollator as a "Thing" in the sense of the WoT standard and is described more in details in section 6).
- **Gait Sampler** is realized by a REST service implemented in the Java programming language. Figure 10 shows a snapshot of the window used to debug the Gait Sampler component.

4



FIGURE 10: GAIT SAMPLER: INPUT FROM THE SERVIENT (TOP) AND EXTRACTED GAIT TIME SERIES (BOTTOM)



4. SUB USE CASE 2: AUTOMATED, TRUSTWORTHY HEALTHCARE CONNECTIVITY

4.1. Scope and Objectives

The SARA system generates traffic with heterogeneous features, stemming from the diversity of IoT devices embedded in the different SARA components such as the robotic rollator, the robot or the Body Area Network (BAN). Namely, the traffic will have different requirements in terms of priority:

- Low priority. For instance, traffic from appliances within the smart home during ordinary activities.
- High priority. Traffic generated by those Assistive Tasks like the Fall Management AT described in section 2.1.1, which are related to potential health problems of the elderly people under control, e.g. abnormal change in his breathing, in his heart rate or in his gait.

Moreover, the SARA use case involves data related to a health application. Thereby, it involves several trust levels that have to be considered for their processing at the networking layer:

- Low trust. For instance, the traffic generated by the mobile phone.
- Medium trust. For instance, the traffic from sensors at the smart home.
- High trust. For instance, the traffic generated by the robot assistant.

The heterogeneity in the traffic requires a flexible network that is capable to process the traffic accordingly. Moreover, the heterogeneity of the traffic leads to an increased network complexity, as it can lead to more configurations to adapt the network to the traffic features. In order to cope with the traffic heterogeneity and with the increased network complexity we adopt herein the Network Function Virtualization (NFV) framework. NFV provides a flexible network, as it relies on the virtualization of the computing, storage and networking resources yielding the so-called Network Function Virtualization Infrastructure (NFVI). Thereby, the network functions can be deployed on top of this NVFI in the form of the so-called Virtual Network Functions (VNF). This provides a flexible networking approach as the virtual resources can be allocated easily according to the network services ' needs. Moreover, the VNFs can be chained to form the so-called Service Function Chains (SFC). For instance, one can form an SFC consisting of a Firewall (FW) and a Load Balancer (LB). Another example is an SFC consisting of a Firewall (FW) and a Load Balancer (LB). Another example is an SFC consisting of a FW, a Deep Packet Inspection (DPI) and Intrusion Detection System (IDS). Thereby, NFV provides flexibility and the proper means to deal with traffic heterogeneity. Last, but not least, the configuration and lifecycle management of the VNFs along with the virtualized infrastructure management, are automated thanks to the NFV Management and Orchestration (NFV MANO) sub block within the NFV framework.

Considering the above, there is significant motivation to leverage the flexibility provided by SFC to define specific service chains for each type of traffic. By applying the previous described procedure of chain instantiation, the SARA solution can support traffic forwarding through specific service functions. That includes the traffic forwarding for the different type of traffic exchanged between the different actors as following:

- Chain 1 Mobile Phone: Firewall -> DPI -> IDS -> Output.
- Chain 2 Robotic Rollator: Firewall -> IDS -> Load Balancer -> Output.
- Chain 3 Smart Home: Firewall -> IDS -> Output.
- Chain 4 Robotic Assistant: Firewall -> Load Balancer -> Output.
- Chain 5 Malicious: Firewall -> Honeypot.

The above scenario sketches a complex environment, requiring support for integration of heterogeneous devices and communication protocols, high degrees of interoperability, and support for distributed services and applications (each with its own set of intrinsic requirements), while guaranteeing the safety of the patient and the security and privacy of her patient data. This use case is visualized in Figure 11, which depicts the various types of devices, their interactions, and the involved communication technologies.





FIGURE 11: SARA-HEALTH SCENARIO AND TRAFFIC CLASSIFICATION

The instantiation of a sequence of functions can constitute a service chain. Similar to the insertion of service functions in the SFC manager through the exposed service function REST interface, service chains can be inserted. In the list of the service functions, the firewall, the DPI, the IDS and the Load Balancer have been defined as the most crucial ones to enable the SPDI properties required by each chain to guarantee. Each VNF has a unique IP address which is required for the configuration and integration with the other functions interacting also with the use case devices and apps. Pattern Orchestrator is responsible to forward the SFC request to the Pattern Engine in order to verify or instantiate SFC requests.

By leveraging the NFV framework, herein several SFC will be deployed on top of the virtual resources exposed by the NFVI. Namely, first a traffic classifier splits the incoming traffic from the SARA IoT field devices into several data flows. Then, these data flows are processed by the most proper SFC according to the traffic features.

The main objectives of this sub use case are summarized as follows:

- Show how NFV along with the pattern related components, offer a flexible solution to deal with the heterogeneous traffic flows of SARA. Also, how they reduce the manual intervention in the networking configuration and management, thus yielding a network management automation solution.
- Provide an NFV testbed to process, at the networking level, the different data flows of the SARA use case, according to their heterogeneous requirements.
- Provide the NFV MANO components that control the lifecycle management of the VNF along with the management of the virtualized infrastructure to deploy the VNFs. Thus, the NFV MANO leads to the automation of the virtual resources and the network functions management.



4.2. Interaction with SEMIoTICS framework

The components of the SEMIoTICS 'framework that are used in this sub use case are highlighted in Figure 12 and their interaction is detailed in the following subsections.



FIGURE 12: SEMIOTICS' COMPONENTS USED IN SUB USE CASE 2

4.2.1. INTERACTION WITH NFV

The SEMIoTICS 'NFV component is a fundamental part of the sub use case presented in this section. The NFV component allows to virtualize the computing, the networking and the storage resources at the cloud level yielding the so-called NFVI. Thereby, the VNFs that compose the SFC can be deployed as VMs that leverage those virtual resources. Moreover, the NFV component has the role to manage the whole lifecycle of the VNFs, through the NFV MANO subcomponent, see Figure 13. Thereby, it controls their onboarding within the NFV framework. Also, it provides configuration files that allow to automatize the instantiation of the VNFs, e.g. by defining the features and configuration of the VMs that hold the VNFs. Moreover, it allows the termination of the VMs that hold the VNFs. Next, more details on the role of the NFV component are given.

4.2.2. NFV ORCHESTRATOR AND VNF MANAGER

These are the sub blocks of the NFV component that manage the lifecycle of the VNF. Namely, the NFV orchestrator provides templates of configuration files to allow the end-user to describe the features and functionalities of the Network Services (NS) and their associated VNFs. These configuration files are so-called



VNF descriptors and NS descriptors. Note that this explanation is assuming that we use an OSM implementation of the NFV MANO, thereby a VNF is always within the context of a NS. Also, a cloud init file can be included to indicate the initial behavior, configuration and software installations of the VM that will hold the VNF. Once these configuration files are edited, we can onboard to the library of the OSM the VNF and NS packages that define the VNFs and the NS. Then, one can trigger the instantiation of the NS and their associated VNFs. This provokes that the NFV orchestrator communicates internally with the VNF manager, note that both blocks are implemented by the OSM. Then, the VNF Manager communicates internally with the VIM to trigger the instantiation or termination of the VNF. Thereby, it triggers the management of the virtual resources in the VIM to deploy the VMs that hold the VNFs.

4.2.3. VIRTUAL INFRASTRUCTURE MANAGEMENT

This block is the responsible for managing the NFVI, i.e. the virtual computing, storage and networking resources that allow to deploy the VNFs at the cloud level. The VNF manager and the NFV orchestrator manage the communication with the VIM internally. In other words, the end-users or applications that are external to the NFV framework do not need to communicate with the VIM, as the NFV orchestrator provides a high level interface through the configuration files mentioned above and through REST API interfaces. Thereby, as an example, the VIM creates or terminates the VM that holds the VNF, when the instantiation or termination of the VNF is triggered at the NFV orchestrator. The instantiation or terminate the allocation or release of the virtual resources from the NFVI to create or terminate the VM. Note that herein the VIM and NFVI are implemented by means of the OpenStack software, see section 4.4.

4.2.4. PATTERN ORCHESTRATOR

Pattern Orchestrator is responsible for automated configuration, coordination, and management of different patterns and their deployment. Moreover, it is used for processing a received requirement in order to translate it to Drools facts. The result of the said processing enables the Pattern Orchestrator to choose which Pattern Engine should receive the corresponding Drools facts in order to reason with them.

4.2.5. PATTERN ENGINE

Pattern Engine at the application layer, includes the pattern rules in the form of Drools rules, responsible for verifying and instantiating VNFs and SFCs. With the said rules the Pattern Engine is able to reason whether a new VNF is needed to be instantiated in order to complete a specific SFC. Moreover, if an SFC is not instantiated at all, the rules allow the instantiation of the SFC after having gathered all the necessary VNFs. Due to the fact that the Pattern Engine at the application layer, has a global view of the pattern facts across all layers, it makes it appropriate for providing up to date information on a graphical interface designed specifically for SFC management in a higher level. This interface is called SFC GUI and extracts all the information depicted directly from the Pattern Engine.

4.2.6. THING DIRECTORY

Thing Directory at the application layer represents a global repository for all the registered Things and is used to browse Things based on their Thing Description. The information extracted from Thing Directory is used from the Backend Semantic Validator to verify the interoperability between two Things. Moreover, it is used from the SFC GUI to populate the available nodes that can be used as source and destination of a flow.

4.2.7. BACKEND SEMANTIC VALIDATOR

The Backend Semantic Validator component is responsible for semantic validation mechanisms at the backend layer. It receives from the Pattern Engine the source and destination of a flow in order to verify the interoperability between them. When the interoperability between source and destination is not present, The Backend Semantic Validator informs the Pattern Engine so that the latter can take appropriate actions.

4.2.8. GUI

The GUI component is responsible for the visualization of SPDI pattern monitoring and pattern details. It receives from the Pattern Orchestrator information regarding the source and destination of a chain as well as the current state of the SPDI properties for the said chain.



4.3. Setup testbed and integration

4.3.1. NFV TESTBED

Herein, we leverage CTTC's NFV testbed for several purposes. First, to host the VNFs that process the traffic stemming from the SARA's IoT GW. Second, those VNFs are chained yielding the so-called Service Function Chains (SFC), which pave the way to support the trustworthy healthcare service. Third, a Virtual Private Network (VPN) has been configured at CTTC to let other partners access the NFV testbed from outside CTTC. Fourth, as it is detailed below, the NFV testbed exposes a northbound interface (NBI). That interface allows external entities, such as the pattern orchestrator or the SFC GUI, to control remotely the whole lifecycle of the SFC, NS or VNFs and to enforce the proper patterns dynamically. In the next Figure 13, we show a block diagram of the NFV testbed along with its interaction with the external entities through the NBI.



FIGURE 13: NFV TESTBED AT CTTC AND ITS INTERACTION WITH THE SFC GUI AND THE PATTERN ORCHESTRATOR

More specifically, this testbed consists of the next functional blocks according to the ETSI's NFV standard [1]. First, the VNFs are deployed on top of a virtualized infrastructure that exposes virtual computing, networking and storage resources. This infrastructure is so-called Network Function Virtualization Infrastructure (NFVI) and is one of the fundamental blocks of the NFV framework. Second, the whole NFV framework is managed by the so-called NFV Management and Orchestration (NFV MANO). More specifically, the NFV MANO is responsible for the whole lifecycle of the NS, the VNFs and the SFC, i.e. their creation, deployment and termination. The NFV MANO consists of three sub blocks. The NFV Orchestrator (NFVO), the VNF Manager (VNFM) and the Virtualized Infrastructure Manager (VIM). The virtualized resources and the whole NFVI are controlled by the VIM. The whole lifecycle of the VNF is managed by the NFVO and the VNFM, which have the responsibility to provide northbound API interfaces to interact with the end-user. This allows the end-user to edit the VNF within the NFV



framework, i.e. as part of the library of available VNFs. Moreover, the NFVO/VNFM blocks let the end-user to trigger the VNF instantiation or termination on top of the NFVI. In this regard, it is worth mentioning that the NFVO communicates downwards with the VNFM and the VNFM in turn communicates downwards with the VIM. Similarly, it is the NFVO that controls the whole lifecycle of the NS and the SFC and communicates downwards with the required operations of the lifecycle such as instantiation or termination. In fact, note that a NS contains or embeds one or more VNFs. More insights on NFV for the SEMIOTICS purposes are given in the SEMIOTICS 'deliverable D3.8 [2].

The above-mentioned NFV blocks are implemented in CTTC's NFV testbed by using the next software:

- OSM (Open Source MANO) [3]. It implements the NFVO along with the VNFM blocks.
- OpenStack [4]. The OpenStack controller implements the VIM, the OpenStack compute node implement the NFVI.

The NFV testbed, mentioned above, has the topology described in Figure 14. More specifically, the NFV MANO consists of two functional blocks. Thus, on the one hand, the OSM implements the NFVO and VNFM. On the other hand, the NFV MANO contains another sub block, the OpenStack controller node, which implements the VIM. The NFVI is composed of one functional block, an OpenStack compute node that exposes its virtualized resources to instantiate the VNFs. A switch is leveraged for the communications between all the functional blocks mentioned above.



FIGURE 14: TOPOLOGY OF THE CTTC'S NFV TESTBED.

In order to implement the NFVI and NFV MANO functional blocks described in Figure 14 we have deployed VMs within the servers available at the CTTC premises. Namely, each functional block of Figure 14 is a VM. Moreover, the computing, storage and RAM memory of the VM have been set up according to the requirements of each of the software components. Next, we describe the features of each VM and the switch:

- VM containing the OSM.
 - Storage of 98 GB.
 - RAM of 8 GB.
 - 4 CPU cores.
- Operative system is the Ubuntu 18.04.4 LTS.
- VM containing the OpenStack controller node.



- Storage of 98 GB.
- RAM of 32 GB.
- 6 CPU cores.
- Operative system is the Ubuntu 18.04.4 LTS.
- VM containing the OpenStack compute node.
- Storage of 10 GB.
- RAM of 8 GB.
- 4 CPU cores.
- Operative system is the Ubuntu 18.04.4 LTS.
- The switch is a 10GBASE-T programmable switch.

Moreover, it is important to note that OSM version 7 has been installed, following the guidelines in [5]. Regarding the OpenStack, the Train version has been installed and set up in our NFV testbed. To this end, a devOps approach has been adopted. More specifically, OpenStack Ansible has been leveraged to ease and to automate the installation of the application and the configuration process of the underlying infrastructure that will support the OpenStack. We have followed the OpenStack Ansible guidelines in [6] to install and configure OpenStack Train in our NFV testbed. It is worth noting that Ansible is a type of configuration-management software, which follows an infrastructure as code approach. Thereby, it relies on an inventory, i.e. a set of configuration files that describe the nodes that can be accessed by Ansible. Moreover, Ansible relies on a set of files that permit to carry out operations on the managed nodes, thus these files are so-called Playbooks. Note that each Playbook maps a group of hosts to a set of roles.

Regarding the NFV testbed, it is also important to mention that the capability to support SFC in OpenStack has been enabled. To this end, we have installed the "networking-SFC" plugin for the OpenStack Neutron sub-block⁸. This plugin provides APIs to support SFC directly from OpenStack Networking (i.e. Neutron). Moreover, a set of constraints must be taken into account to enable SFC in OpenStack:

- First, each compute node of the NFVI must reserve a dedicated network interface for connecting VNFs to VLAN provider networks.
- Second, VXLAN are the only supported network technology for SFC data planes.
- Third, the data plane interconnecting VNFs in the SFC must support OpenFlow.

Additionally in FORTH's premises, the Pattern Orchestrator, Pattern Engine, Backend Semantic Validator and SFC GUI are deployed also in VMs. An Intel NUC (Figure 15) is used with 32GB RAM, 500GB storage and a CPU i7-6770HQ 2.60GHz with 8 cores. Proxmox Virtual Environment is installed in the Intel NUC which hosts the aforementioned VMs

- VM containing the Pattern Orchestrator.
 - Storage of 10 GB.
 - RAM of 4 GB.
 - 4 CPU cores.
 - Operative system is the Ubuntu 18.04.4 LTS.
- VM containing the Pattern Engine.
 - Storage of 10 GB.
 - RAM of 4 GB.

⁸ "Service Function Chaining Extension for OpenStack Networking.," [Online]. Available: https://docs.openstack.org/networking-sfc/latest/. [Accessed December 2020].



- 4 CPU cores.
- Operative system is the Ubuntu 18.04.4 LTS.
- VM containing the Backend Semantic Validator.
 - Storage of 15 GB.
 - RAM of 2 GB.
 - 2 CPU cores.
 - Operative system is the Ubuntu 18.04.5 LTS.
 - VM containing the SFC GUI.
 - Storage of 20 GB.
 - RAM of 2 GB.
 - 4 CPU cores.
 - Operative system is the Ubuntu 18.04.4 LTS



FIGURE 15: INTEL NUC AT FORTH TESTBED

Last, but not least, it is important to expose that the interface between the NFV and the SFC GUI-Pattern orchestrator is done through REST APIs and http requests, according to the ETSI GS NFV-SOL 005⁹. In fact, this document describes the standardized implementation for the Os-Ma-NFVO reference point, which is ETSI-NFV compliant [1] and defines the interface between the NFV MANO and external Operational Support Systems (OSS). Thereby, in our case, the SFC GUI-Pattern Orchestrator are considered as OSS and are fully compliant with the ETSI-NFV framework.

The detailed procedure used to validate the NFV testbed described in this section can be found in Appendix A.

4.3.2. SERVICE FUNCTION CHAINING PATTERNS VALIDATION

Service Function Chaining Patterns provide the ability to define an ordered list of security network services (e.g. firewalls, DPIs, IDS) for security in network infrastructures by creating chains at design and by updating functions in chains based on available ones at runtime. SFC patterns should cover the placement, security and scalability

⁹ ETSI, "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point," ETSI GS NFV-SOL 005, 2018.



aspect of SEMIoTICS network infrastructures. The deployment of network service functions can be used to guarantee specific network security and dependability properties. However, stitched them into chains can satisfy more than one SPD properties. One of the innovative approaches supported by this work, is the dynamic instantiation of SFCs based on the predefined SFC patterns. When there is a request for an SFC instantiation containing service functions, the depicted in Figure 16 procedure should be followed. If the SFC does not exist, the instantiation of the respective SFC is deployed through the identification of the requested VNFs. If the VNFs exist in the service nodes, the SFC is updated including these VNFs. If the VNFs do not exist, the service node with the available resources is requested to instantiate the respective VNFs. The procedure is ended when all the requested VNFs are included in the SFC.



FIGURE 16: VNF INSTANTIATION BASED ON SFC REQUEST

To proceed to the above description, SFC patterns are developed to achieve the requirement for end to end guarantees by the traffic forwarding through different security service functions. The patterns can be expressed as Drools rules¹⁰ to enforce the following requirements:

- Verify service function chains on chain requests.
- Instantiate service function chains, if the required functions have been already instantiated.
- Verify functions to insert them in the request chain.
- Instantiate not defined service functions to satisfy service function chain requests.

The ultimate goal of the SFC GUI is to enable the Administrators of the SARA solution to interact with the system that provides a secure networking infrastructure, via the associated proactive and reactive security mechanisms, such as the deployment of network security services and the continuous monitoring and intrusion detection. In Figure 17 the SFC GUI is presented which gathers in one screen all the necessary information.

¹⁰ Drools Business Rules Management System https://www.drools.org





FIGURE 17: SFC GUI

All information depicted in SFC GUI is gathered from Backend Pattern Engine with a simple http-GET request (Figure 18).

- Backend Pattern Engine gathers information from:
 - Thing Directory (Nodes)
 - OSM (Function List and Function Instances)
 - SFC Requests (Chains)
 - Pattern Requirement (user input)

GET	/sfcGUI	sfcGUI
Curl		
curl	-X GETheader 'Accept: application/json' 'https://139.91.182.125:9443/sfcGUI'	
Reque	est URL	
http:	s://139.91.182.125:9443/sfcGUI	
Reque	est Headers	
{ }	<pre>ccept": "application/json"</pre>	
Respo	onse Body	
{	<pre>"requirementList": [], "NodeList": [], "functionDescriptionList": [], "chainList": [], "functionInstancesList": []</pre>	

FIGURE 18: PATTERN ENGINE REST FOR SFC GUI



A user can request for specific VNFs to be applied in a flow of a specific source and destination such as the end hosts such the robot, patient or backend services such as AREAS, AI services. The list of sources and destination will be populated on the next cycle based on the information retrieved from the Thing Directory or manually in case of backend services. The VNFs that the user can choose from, is dynamically populated with information provided by the OSM.

Upon clicking on the submit button a new Pattern Requirement is created that is sent directly to the Pattern Engine and is consumed by the addSFCReq endpoint (Figure 19 and Figure 20).

Response Class (Status 200) string Parameters Parameter Value Description Parameter Data String [*] sfc:e:::Docctor', [*] scc:::::Docctor', [*] scc:::::Docctor', [*] scc::::::Docctor', [*] scc::::::Docctor', [*] scc::::::Docctor', [*] scc::::::::Docctor', [*] scc:::::::::::::::::::::::::::::::::::	POST	/addSFCReq2		add	SFCReq2
Response Content Type application/json* Parameters Parameters factString	Respon string	nse Class (Status 200)			
Parameters Description Parameter Parameter <td>Respon</td> <td>se Content Type application/json 🗸</td> <td></td> <td></td> <td></td>	Respon	se Content Type application/json 🗸			
Parameter Value Description Parameter Dype factString [*sfcReqID*:8, "string" body string body string "string" "chain":1 "name": "firewall", "instantiated": "false"}, "string" body string "string" "string" body string body string Parameter "chain":1 "name": "firewall", "instantiated": "false"}, "string" string body string "string" "string" body string body string body string Parameter "chain":1 "name": "firewall", "instantiated": "false"}, "string" body string string "string" "string" body string string string string string Parameter contant tealers tealers <td>Param</td> <td>eters</td> <td></td> <td></td> <td></td>	Param	eters			
FactString { "sfcReqID":0, "sfc."; "bottor, "sfc."; "sfc."; "sfc."; "sfc."; "sfc."; "sfc."; "sfc."; "bottor, "sfc	Parame	ter Value	Description	Parameter Type	Data Type
Parameter content type: application/json Response Messages HTTP Status Code Reason Readers 201 Created 401 403 Forbidden 404 Not Found Try it out! Htde Response Curl - X POSTheader 'Content-Type: application/json'header 'Accept: application/json' -d '{*sfcReqID*:0, *src*:*D Request URL https://139.91.68.107:9443/add5FCReq2 Request Headers { { Response Body Response Code	factS	<pre>tring { "sfcReqID":0, "src":"Doctor", "dst":"AREAS", "chain":[{"name":"firewall","type":"firewall","instantiated":"false"}, {"name":"dpi","type":"dpi","instantiated":"false"} "satisfied":"false" }</pre>	factString	body	string
201 Created 401 Unauthorized 403 Forbidden 404 Not Found Try it out! Hide Response Curl Curl -X POSTheader 'Content-Type: application/json'header 'Accept: application/json' -d '{"sfcReqID":0, "src":"D Request URL https://139.91.68.107:9443/addSFCReq2 Request Headers { * Accept": "application/json" Response Body no content Response Code 200	Respon	Parameter content type: application/json nse Messages status Code Reason Response Model		Headers	
401 Unauthorized 403 Forbidden 404 Not Found Try it out Hide Response Curl curl -X POSTheader 'Content-Type: application/json'header 'Accept: application/json' -d '{"sfcReqID":0, "src":"D Request URL https://139.91.68.107:9443/addSFCReq2 Request Headers { Response Body no content Response Code 200	201	Created			
403 Forbidden 404 Not Found Try it out! Hide Response Curl curl -X POSTheader 'Content-Type: application/json'header 'Accept: application/json' -d '{"sfcReqID":0,"src":"D Request URL https://139.91.68.107:9443/addSFCReq2 Request Headers { "Accept": "application/json" Response Body no content Response Code 200	401	Unauthorized			
484 Not Found Try it out! Hide Response Curl curl -X POSTheader 'Content-Type: application/json'header 'Accept: application/json' -d '{"sfcReqID":0, "src":"D Request URL https://139.91.68.107:9443/addSFCReq2 Request Headers { "Accept": "application/json" ? "Accept": "application/json" Response Body no content Response Code 200	403	Forbidden			
<pre>curl -X POSTheader 'Content-Type: application/json'header 'Accept: application/json' -d '{"sfcReqID":0,"src":"D</pre>	404 Try it c	Not Found ut! Hide Response			
<pre> Request URL https://139.91.68.107:9443/addSFCReq2 Request Headers { "Accept": "application/json" Response Body no content Response Code 200 </pre>	curl	-X POSTheader 'Content-Type: application/json'header 'Accept: application/json'	-d '{"sfcRe	qID":0,"sı	c":"Doc
Request URL https://139.91.68.107:9443/addSFCReq2 Request Headers { "Accept": "application/json" Response Body no content Response Code	4				۲
https://139.91.68.107:9443/addSFCReq2 Request Headers { "Accept": "application/json" Response Body no content Response Code 200	Reque	st URL			
Request Headers { "Accept": "application/json" Response Body no content Response Code 200	https	:://139.91.68.107:9443/addSFCReq2			
<pre>{ *Accept*: *application/json* Response Body no content Response Code 200</pre>	Reque	st Headers			
Response Body no content Response Code	{ } "Ac	ccept": "application/json"			
no content Response Code	Respo	nse Body			
Response Code	no	content			
200	Respo	nse Code			
	200				

FIGURE 19: PATTERN ENGINE REST FOR SUBMIT BUTTON OF SFC GUI



```
2020-12-15 02:28:13.835 INFO 46195 --- [nio-9080-exec-3] o.d.c.k.bui
2020-12-15 02:28:13.945 INFO 46195 --- [nio-9080-exec-3] o.d.c.k.bui
2020-12-15 02:28:14.168 WARN 46195 --- [nio-9080-exec-3] o.a.maven.i
Chain Instance Not Exists :Time: 2020-12-15 02:28:21.079
Chain Descriptor Not Exists :Time: 2020-12-15 02:28:21.079
Chain Descriptor Begin Creation :Time: 2020-12-15 02:28:21.08
Chain Descriptor Creation Completed :Time: 2020-12-15 02:28:29.373
Chain Descriptor Verified :Time: 2020-12-15 02:28:29.375
Chain Begin Instantiation :Time: 2020-12-15 02:28:29.376
Chain Instantiation Completed :Time: 2020-12-15 02:28:31.539
Chain Instance Verified :Time: 2020-12-15 02:28:31.54
8 rules fired
Fact count is 25
```

FIGURE 20: PATTERN ENGINE FIRING SFC RELATED RULES

After the new requirement has been consumed by the Pattern Engine, the pattern rules (Figure 21 to Figure 25) are fired causing the verification and instantiation of VNFs and SFCs.

```
    rule "Chain Verification Pattern Rule"
    salience 10
    when
    $chain: Chain($functions: functions, instantiated==true)
    $PR: Req($src:src,$dst:dst, pro.chain.functions==$functions, satisfied==false)
    then
    System.out.println("Chain Verified");
    modify($PR){satisfied=true};
    end
```

FIGURE 21: SERVICE FUNCTION CHAIN VERIFICATION



1. rule "Chain Instantiation Pattern" 2. salience 20 3. when 4. \$chain: Chain(\$functions: functions, instantiated==false) 5. \$PR: Req(\$src:src, \$dst:dst, pro.chain.functions== \$functions, satisfied==false) 6. then 7. Orch orch=new Orch(); orch.instantiateChain(\$chain.id, \$chain.name); 8. modify(\$chain){instantiated=true}; 9. 10. System.out.println("Chain Instantiated"); 11. end FIGURE 22: SERVICE FUNCTION CHAIN INSTANTIATION

1. rule "Chain Creation Pattern" 2. salience 30 3. when 4. \$PR: Req(\$src:=src, \$dst:=dst, \$pro: pro, satisfied== false) **not** Chain(\$functions: functions, \$functions:= \$PR.pro.chain.functions, 5. instantiated==false) 6. then 7. Orch = new Orch(); String idd = orch.createChain(\$PR.pro.chain.functions, "chain", \$src.address, \$src.port, \$dst.address, \$dst.port); Chain chain = new Chain(idd, "chain", \$PR.pro.chain.functions, false); 9. 10. insert(chain); 11. System.out.println("Chain Descriptor Created"); 12. end

FIGURE 23: SERVICE FUNCTION CHAIN CREATION



```
1. rule "Virtual Service Network Function Verification"
2. ruleflow-group "SFC"
3. salience 40
4. when
5.
         $function: Function($type:=type, instantiated==true)
         $chain: Chain($functions: functions, $functions not contains)
6.
   $function, instantiated==false)
7.
         $PR: Property($src:=src, $dst:=dst,
   $reqFunctions:=property.chain.functions, satisfied== false)
         Function($type==type) from $reqFunctions
8.
9. then
         System.out.println("Verification of Function");
10.
11.
         $chain.addFunction($function);
        update($chain);
12.
13 end
```

FIGURE 24: SERVICE FUNCTION VERIFICATION

```
1. rule "Virtual Service Network Function Instantiation"
2. ruleflow-group "SFC"
3. salience 50
4. when
       $function: Function($type:=type, instantiated==false)
5.
6.
       not Function($type:=type, instantiated==true)
       $chain: Chain($functions: functions, $functions not contains)
7.
   $function, instantiated==false)
8.
       $PR: Property($src:=src, $dst:=dst,
   $reqFunctions:=property.chain.functions, satisfied== false)
9.
       Function($type==type) from $reqFunctions
10. then
11.
       System.out.println("Instantiation of Function");
12.
       Function function = new Function($function.type);
13.
       modify($function){instantiated=true};
14. end
```

FIGURE 25: SERVICE FUNCTION INSTANTIATION

In use case 2, the main two applications are the Gait analysis and Fall management. More specifically, for the gait analysis the requirement include the instantiation of a chain for the traffic forwarding from the patient to the doctor including a firewall, a DPI and an IDS. Upon submitting the requirement, the appropriate rules are fired at the pattern engine, forcing the instantiation of the required functions in OpenStack and the formation of the chain in the OSM. As soon as all rules have finished, all information is available in the SFC GUI Figure 26.


GuiHub									
≡ Toggle Sidebar ∰ Cockpit	SEMĨelics	Home							FORTH
✿ ⁰ Smart Things →	Pattern Requi	rements						Topology	
SPDI Patterns -	Application	Source	Destination	Service Fu	nctions	Satisfied		🖉 Edit	()
a Monitoring →	GAIT Analysis	patient	doctor	ids firewa	all dpi	True	Remove	Doctor	Call Center
≪ Recipe Cooker ↓ SFC GUI	Select Application: GAIT Analysis V	Choose patient	a Source: Choose	a Destination:	Choose Servi Functions: Dioad-balanc dids firewall	ce Submit			Al Services actend
	Nodes		Function	Instance	es	Chains			
	Node Name	IP Address	Name	Туре	IP	Chain Name	Service Functions	Mobile Access Po	int 35
	patient	13.0.1.175	ids-VNF	ids		Chain1	Show Functions	Patient Home Galeway	Robot (日)
	robot	13.0.1.214	firewall-VNF	firewall				686 1	
	doctor	13.0.1.237	dpi-VNF	dpi				Smarth	ome
	call_center	13.0.1.170							
		े	This project has received for Horizon 2020 research and agreement number 780315	unding from the Eu I innovation program 5 Conditions of Use	ropean Union's mme under gran Privacy Policy	SEMI	Made by Blues Copyright © 20 All Rights Rese	ioft sp. z o.o.)19 irved.	



The procedure includes the verification of a chain based on the described patterns. If the chain is not instantiated, the pattern will trigger the instantiation of a new chain based on an existing chain descriptor as can be seen in Figure 27 and Figure 28. If the chain descriptor exists, the patterns are able to trigger the creation of the chain before its instantiation of the required functions in the OpenStack, as can be seen in and Figure 29.

O MANO	=								5%	≥ adm	in *	e admin
MAIN NAUGADON										*	Home	NS Instances
🛠 Home	NO											d Marrison
	INS I	istances										W New NS
Overview	Show	10 \$	entries						Searc	hc		
Packages	• Nan	e 🏦 Iden	tifier	11	Nsd name	Operational Status	п	Config Status II	Detailed Status	Acti	ons	
 NS Packages VNF Packages 	chai	n 0095 be53	523c7-f5b3-4150- 3-cb9699b2f609	c	chain- ids-	•			Stage 2/5: deployment	i	÷	•
S NetSlice Templates				firewa dpi	firewall- dpi				of KDUs, VMs and	Ac	tions -	
A Instances									execution environments.			
NS Instances O VNF Instances									ur1. viw: ({'VMs': '0/3', 'networks': '0/2'})			

FIGURE 27 INSTANTIATION OF SERVICE CHAIN IN OSM



Instances

	INSTANCE II) = -			FI	LTER	٥.	AUNCH INSTA	NCE	💼 DELE	TE INSTANCI	ES MORE ACTIONS -
Disp	aying 7 items											
	Instance Name	lmage Name	IP Address	Flavor	Key Pair	Status		Availability Zone	Task	Power State	Age	Actions
	chain-gait-ids-fire wall-dpi-3-dpi-VN F-1	ubuntu1 8-minim al	externalNet 172.113.40.38 sfc 13.0.1.153	firewall- VM-flv	Isanabria	Active	Φ	nova	None	Running	9 minutes	CREATE SNAPSHOT -
	chain-gait-ids-fire wall-dpi-2-firewall- VNF-1	ubuntu1 8-minim al	sfc 13.0.1.152 externalNet 172.113.40.48	firewall- VM-flv	Isanabria	Active	¢.	nova	None	Running	9 minutes	CREATE SNAPSHOT +
	chain-gait-ids-fire wall-dpi-1-ids-VNF -1	ubuntu1 8-minim al	sfc 13.0.1.151 externalNet 172.113.40.181	firewall- VM-flv	Isanabria	Active	¢	nova	None	Running	9 minutes	CREATE SNAPSHOT +

FIGURE 28 INSTANTIATION OF SERVICE FUNCTIONS IN OPENSTACK

	=	🗠 admin 👻 😫 adm
MAIN NAVIGATION	NS Packages	🅷 Home > Projects > admin > ns Packag
🖀 Home		Compose a new NS
		gr compose a new ro
Overview	Show 10 + entries	Search:
Packages	Short Name	11 Description 11 Vendo
NS Packages	4b8264c5-577c-42	207-a3d1-
To VNF Packages		
	=	🗈 admin 👻 😫 admin
MAIN NAVIGATION	NS Packages	👫 Home > Projects > admin > ns Packages
🖨 Home		10
		Compose a new NS
Overview	Show 10 ¢ entries	Search:
Packages	Short Name	1 Description 1 Vendor
NS Packages	chain-ids-firewall-dpi 4b8264c5-577c-420 ed0667980c6b	07-a3d1- Chain OSM
VNF Packages		

FIGURE 29 CREATION OF SERVICE CHAIN DESCRIPTOR IN OSM



To evaluate the developed solution, different experiments for forwarding traffic through the different chains such as for the Gait Analysis application and the Fall management. The patient sends data to the doctor via the instantiated chain including a firewall a dpi and an ids service functions. As can be seen from the tcp dump data capture in FIGURE 30, each service function forwards the traffic to the next node, offering different security guarantees in each chain.



FIGURE 30 GAIT ANALYSIS SCENARIO (2)

Similarly, for the fall management, when there is an emergency, a connection between the Call center and the robot will be opened to exchange data and guidelines. Figure 31 and Figure 32 present the traffic forwarding from call center to robot through the instantiated chain including a load balancer and a firewall.





FIGURE 31 FALL MANAGEMENT SCENARIO (1)



FIGURE 32 FALL MANAGEMENT SCENARIO (2)

Finally, ac can be seen during the evaluation of the two scenario, both experiments can prove the flexibility and applicability of our developed solution to offer traffic forwarding to enable end to end SPD properties.

4.3.3. BACKEND SEMANTIC VALIDATOR AND THING DIRECTORY VALIDATION



Backend Semantic Validator (BSV) is able to verify whether two endpoints, a.k.a. source and destination such as end hosts (robot, patient, smart home) or backend services (doctor, AREAS, AI Services), are semantically interoperable. Figure 33 shows how the Backend Semantic Validator is involved in the flow and Figure 34 show a closer look to that process. The main idea here is to feed the BSV with a pair of end nodes that represent the source and destination. Their Thing Description is looked up in the Thing Directory and based on the result the BSV is capable to respond whether interoperability exists between them.



FIGURE 33: BSV INTEGRATION



Pattern Engine	BSV Thing Directory
Answer: Interoperability: True/False	THINGWED Thing Directory Register (v) Verification of the second of t

FIGURE 34: BSV AND THING DIRECTORY INTERACTION

All the above are possible by the REST endpoint "validateData" (Figure 35) of the BSV which receives the source and destination in order to lookup their Thing Description in Thing Directory

POST	/validate	Data			validateData
Respo string	nse Class	(Status 200)			
Respor	nse Conten	t Type application/json 🗸			
Param	eters				
Param	eter Val	ue	Description	Parameter Type	Data Type
IdsUC	2 I J Par	<pre>{ "idDestination": "string", "idSource": "string" ameter content type: pplication/json </pre>	IdsUC2	body	<pre>Model Example Value [{</pre>
Respo	nse Messa	ages			
HTTP S	Status Code	Reason	Response Model		Headers
201		Created			
401		Unauthorized			
403		Forbidden			
404		Not Found			
Try it	t out!				
Respon	nse Body tesponse": "Tar	n Backend Semantic Validator and I receive	your Data. There is Interoperability in two Things"		

FIGURE 35: BSV "VALIDATEDATA" ENDPOINT



The result of the interaction between the Backend Pattern Engine and the Backend Semantic Validator is visualized in the SPDI monitoring view, where the end to end Interoperability property becomes green as can be seen in Figure 36.

GuiHub			(*
≡ Toggle Sidebar	HOME / SPDI PATTERNS / SPDI PLATFORM	I VIEW	
🚳 Cockpit	SPDI pattern monitori	ng	Show Table
📽 Smart Things 🛛 👻			
SPDI Patterns -		BACKEND	
SPDI platform view		(0/0)	
SPDI recipe view		S P D I P	
🖚 Monitoring 🗾 👻		(0/0) (0/0) (0/0) (0/0)	
< Recipe Cooker		D	
🖵 SFC GUI		(0/0)	(0/1)
		(0/0)	(0/1)
		S P D I (0/0) (0/0) (0/0) (0/0)	
		(0/0)	(0/1)
		GATEWAY (0/0)	(1/1)
			A
		S P D I (0/0)	
		(0/0) (0/0) (0/0)	
		(0/0)	
		end to end Interc	perability
			perability
	1	This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant SENTER SENTERS ■ Made by BlueSoft sp. z o.o. Copyright © 2019	

The evaluation of the SFC approach in the SARA use case was presented in this section. The SDN/NFV-enabled test-bed setup for service function chaining was tested in the NFV-enabled architecture for the dynamic VNF instantiation enabling the interaction between the pattern engine to instantiate VNF through the OpenSource MANO (OSM) and OpenStack.



5. SUB USE CASE 3: ENFORCEMENT AND MONITORING OF GDPR-COMPLIANT ACCESS CONTROL TO CONFIDENTIAL AND SENSITIVE DATA

5.1. Scope and Objectives

We would like to highlight a very important topic of security and privacy: the protection of personal data. Not only with the advent of the GDPR, the protection of confidentiality and also integrity of personal data has become of paramount importance. Especially in a medical use case, such as UC2. However, we would like to note that when we speak of confidentiality protection for GDPR-compliance we can as well map the technical protection mechanisms of SEMIoTICS to protect the confidentiality of trade-secrets. Following the definition in TRIPS from the World Trade Organization, trade secrets describe information that is considered to have a commercial value due to its secrecy and that its secrecy must be protected against third parties. So, when we can prohibit that information is made available or disclosed to unauthorized individuals, which is the more technical definition of confidentiality from ISO 27000, then we can protect both personal data to comply with the GDPR in the title.

The challenges we face are that the GDPR requires you to limit the amount of data you collect to what you actually need and further limit the access to the information already collected to the authorized entities.

The access to data needs to be managed in a very generic fashion to make it usable at large scale but also at very fine-grained levels in order to allow the SEMIoTICS framework to be adapted to many different usage scenarios. To fulfil this objective the access control is attribute-based which allows Role-based access control (RBAC), e.g. you allow or deny access based on an attribute that specifies a role, e.g. like entity with the attribute "doctor" which shall be given to those with the role of a medical doctor in our hospital use case. Using attributes enables fulfilling two GDPR-requirements, first and most obvious they allow to specify policies which control and thus allow to restrict access to the sensitive data. Second, it allows that selected individuals, which have the attribute, could gain access without having to reveal more than their attribute, i.e. a more privacy-friendly form of access control can be implemented. So instead of having to tell the system that one has to state that one is the doctor "John Doe" one would only need to reveal to the system that one has the correct set of attributes, e.g. being a "doctor". This limits the data being collected.

Another challenge was to ensure the confidentiality of medical data even when that data becomes stored or transported outside of SEMIoTICS perimeter. To address this challenging objective, SEMIoTICS facilitates a cryptographic mechanism, known as attribute-based encryption (ABE). This is like normal cryptographic encryption, e.g., we encrypt data and unless you have the correct decryption key you cannot read the data. However, in many applications it is not a-priori known who the authorized recipient will be. Instead, one might only know certain attributes of authorized entities and all those entities who possess the right attributes would form a group. This means that encryption can happen, e.g., data should only be read by entities that are doctors, without knowing nor without needing to know who a doctor at the time of retrieval is. SEMIoTICS achieves this loose coupling and still enforces confidentiality for data even if it is stored on systems outside the realm of SEMIoTICS.

To enforce one of the most sought-after features of GDPR-compliance, that of retention times, the attributebased encryption capabilities of SEMIoTICS are combinable with a time-variant attribute. That means that the patient's system might generate date, e.g., a video stream or a location, which is encrypted not only for doctors (see our previous example), i.e., using an attribute "doctor" during encryption, but with an additional attribute that encodes that this data can only be accessed for a certain time frame, e.g., 48h is an often-used retention period for videos. The technical details have been discussed in D4.12.

Last but not least, the challenge is to monitor that the system correctly enforces the intended behaviour, for this we describe how SEMIoTICS monitoring is used as a solution for periodical self-audit capabilities by monitoring for pattern-compliance.



5.2. Interaction with SEMIoTICS framework

5.2.1. AUTHORIZATION POLICY MANAGEMENT

Figure 37 shows the workflow of the Remote Gait Analysis Assistive Task (see section 2.1.2): the workflow starts with the scheduling of the Gait Analysis session. The data shared by devices can only be provided to systems or users who have the appropriate privileges at the moment, specifically, the patient's data can only be accessible by his or her doctor during the specified period (e.g., during the exercises). Changing policies for accessing specific devices will take place before and after exercise. The design shown in Figure 37 fulfils data protection principles like purpose limitation – the data is processed only for the legitimate purposes specified explicitly to the patient, the data processing is done in such a way which ensures appropriate security, integrity and confidentiality.

The exercise can be either postponed or started. If the patient postpones the exercise, it is rescheduled (i.e., for the next day) and it's one end of the flow. If the patient chooses otherwise, SARA sends an HTTP request to Pattern Orchestrator informing that the exercises have already started, and the doctor should get permission to get the patient's data for the time of exercises. Pattern Orchestrator transforms the request and sends it to Backend Pattern Engine. Then, using the Security Manager's API Backend Pattern Engine updates the policies to give the doctor access to the data. As soon as the patient finishes exercises, the flow is repeated but the only difference is the update of policies takes away privileges to inspect the patient's data instead of giving them.



FIGURE 37: REMOTE GAIT ANALYSIS WORKFLOW



- SARA: The component is responsible for interaction with the patient. SARA's main role in the use case
 will be to trigger the request to update the doctor's policies after the patient starts exercises and as soon
 as they are finished.
- Security Manager (backend): The component will be used in the sub-use case both as the security
 provider and policy management point. It will serve to generate an authentication token using SARA's
 credentials and later on, to verify entities based on their token. The second role of Security Manager will
 be to alter doctor's permission, either to allow him to get access to the patient's data during the exercises
 or to take away privileges after the exercises are finished.
- **Pattern Orchestrator:** Pattern Orchestrator receives the HTTP request from SARA and transforms it into facts that will later send to the Backend Pattern Engine.
- Backend Pattern Engine: Backend Pattern Engine reasons based on the received facts and sends HTTP request to Security Manager to change the doctor's privileges.

5.2.2. AUTHENTICATION MANAGEMENT

To authenticate SARA, to every HTTP request an authorization token is added. The Token is generated based on SARA's unique client identification number and client secret. All incoming request to the device will be intercepted by Policy Enforcement Point which verifies whether the owner of the request has sufficient permission to get access to the data. If it does, the request is forwarded to its original destination, if not an error is thrown due to lack of privileges (according to standard RFC 7519).



FIGURE 38: AUTHENTICATION MANAGEMENT PROCESS

- AEP (Authentication Enforcement Point) Authentication Enforcement Point is a stand-alone application
 which generates an authentication token that adds it to every HTTP request that is coming out of SARA.
 To apply a valid token, AEP uses Security Manager's OAuth 2.0 Clients Credentials Grant flow using
 SARA's client-id and client secret. Using these values, Security Manager can generate a unique token
 for the component, which is added to the HTTP request header in order to not be rejected by Policy
 Enforcement Point. This component supports also the automatic addition of token to all outgoing HTTP
 requests and it is supported when it runs in the backend layer.
- PEP (Policy Enforcement Point) In order to secure access to the components' resources, the APIs of a
 device is protected by Policy Enforcement Point. PEP provides security in two ways. Firstly, it is deployed
 as a sidecar along with the primary application. They both share the same network allowing them to



communicate through localhost. The main application is only accessible from within local network, because it does not expose its port outside the device, only PEP does that, meaning that in order to get the resource of the main application all HTTP traffic must go through PEP. Secondly, the HTTP request must bear a valid authorization token and the client which makes an HTTP call has to be authorized to do so. To evaluate client authorization, PEP uses the Security Manager's Policy Decision Point. After the evaluation of the client's policy, HTTP call is either allowed to reach the main application or it is rejected due to the insufficient permission.

• **Device** The device is a generic component which is compliant to the WoT standard and has an API. The device shares the data through its API.

As stated before, PEP is a generic component that can be used in every setup which involves guarding REST API. In the following scenario, PEP is used to secure communication between GUI and Monitoring Component. PEP was deployed along with Monitoring Component, as its sidecar whereas AEP was deployed with GUI. Four possible outcomes might be distinguished:

• HTTP request does not bear any authorization token

<pre>INFO 1 [nio-8050-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]</pre>	: Initializing Spring DispatcherServlet 'dispatcherServlet'
INFO 1 [nio-8050-exec-1] o.s.web.servlet.DispatcherServlet	: Initializing Servlet 'dispatcherServlet'
INFO 1 [nio-8050-exec-1] o.s.web.servlet.DispatcherServlet	: Completed initialization in 294 ms
<pre>INFO 1 [nio-8050-exec-1] com.bluesoft.pep.service.EntityService</pre>	: HTTP request is DENIED: Authorization token is missing

• HTTP request does have authorization token though it is invalid

2020-12-17 14:03:26.081 INFO 1 --- [nio-8050-exec-3] com.bluesoft.pep.service.EntityService : DENIED The request body to Security Manager is: {"actions":[{"entityId":"user!@!local","entityType":"user","method":"write","field":"user"}]} The response from Security Manager is: Unauthorized

• HTTP request has a valid authorization token and the entity is authorized to get the resource

2020-12-17 08:23:13.997 INF0 1 --- [nio-8050-exec-3] com.bluesoft.pep.service.EntityService : APPROVED The request body to Security Manager is: {"actions":[{"entityId":"admin!@!local","entityIype":"user","method":"write","field":"user"}]} The response from Security Manager is: {"result":[true]} 2020-12-17 08:23:28.604 INF0 1 --- [nio-8050-exec-3] com.bluesoft.pep.service.EntityService : The HTTP methods is: PUT The request body/narams is: {}

The request body (harams is: () The response is: <200,Submitted! [Content-Type:"text/plain;charset=UTF-8", Content-Length:"10", Date:"Thu, 17 Dec 2020 08:23:28 GMT", Ke on:"keep-altve"]>

 HTTP request has a valid authorization token but the entity does not have required permissions to get the resource

2020-12-17 13:12:25.199 INFO 1 --- [nio-8050-exec-1] o.s.web.servlet.DispatcherServlet Completed initialization in 287 ms 2020-12-17 13:12:41.395 INFO 1 --- [nio-8050-exec-1] com.bluesoft.pep.service.EntityService DENIED The request body to Security Manager is: {"result":[false]}

5.3. Setup testbed and integration

There are 4 main steps for enforcing and monitoring that are taking place in several components spread over all layers of the SEMIoTICS architecture:

- First, setup tasks, as well as later management tasks, establish the policies and users as well as their attributes.
- Second, the actual enforcement of access control.
- Third, we dynamically adapt the access control to react, e.g., to emergency situations like in the fall assistance task.
- Forth, monitoring for compliance, e.g., GDPR-conformance.

In the remaining text we will detail how these main steps are happening within the generic storyline of UC2 we will highlight which components' interactions are facilitated to reach our objectives.



5.3.1. SETUP STEPS

SEMIOTICS is built upon the concept that each participating entity has their own identity and their own set of attributes. If you think about human users, the name would be "John Doe" and an attribute could be "doctor". To ensure that only particular entities are able to obtain certain attributes, the identity management must initially setup administrative users who can change those attributes or create new users, and the system could also gradually enforce administrative restrictions, by enforcing access control on the ability to modify (create, change, delete) attributes or new entities. Further, there are policies which govern which access is allowed and which access is restricted. This follows the general principle of access control policies, e.g., an entity is given the right to access an object in a certain manner, e.g., an entity with the attribute doctor is allowed to read Hans Meier's medical records. These policies are created initially as part of the system's setup and the use of patterns allows to create those even faster. We explained the usage and technical backgrounds of attribute-based policies as well as entity related policies in D4.12 Section 2.2.4.1. These policies are defined individually as required by the respective application or use case. In order to provide a more user-friendly way to define the policies defined by config files, there is the possibility to set the policies through the currently deployed graphical user interface (GUI). Our GUI is implemented as a node is based application, which facilitates the Security Manager API and can also be used to showcase how to interact with the Security Manager. All necessary functionality of the Security Manager's API is made available via the GUI and can be accessed via the interface. This removes the hurdle of having to write complicated configuration files to initially setup the Security Manager.

Furthermore, one of the initial steps during the system setup is to define the patient's privacy settings. As each patient may have other privacy concerns, we can individually enforce different privacy settings for each patient. This can range from defining how long each private and sensitive data should be accessible, by whom it should be accessible to defining several characteristics of the person that can access the data (e.g., gender, age, language, country). These settings can, of course, be later changed manually by the patient him-/herself.

5.3.2. ENFORCING ACCESS CONTROL

The SEMIoTICS framework allows us to enforce access control on privacy sensitive data in two different ways:

- **By means of access control**: When an entity with the role "doctor" tries to access the patient's data, the request is intercepted by the Policy Enforcement Proxy (PEP) (see Section 5.2.6 and D4.12 Section 3.2.4.1 for detailed information) which then forwards this request to the Security Manager. In this context the Security Manager functions as a Policy Decision Point (PDP), that then either allows or rejects the access to the patient's data, based on the entity's attributes (see Section 5.2.6 and D4.12 Section 3.2.1.1). For Use-Case 2, this mechanism is especially used for step 7 of the Remote Gait Analysis Assistive Task.
- **By means of encryption**: During an emergency case, the patient's data is accessible by anyone of the role "doctor" (enforced by the Security Manager and PEP). However, the SEMIoTICS framework is also able to enforce the patient's previously defined privacy settings. This is done by encrypting the patient's data with the help of Attribute Based Encryption (see D4.12 Section 3.2.1.4). Due to the encryption, the privacy sensitive data is now only accessible by an entity that fulfils all conditions that the patient wants to be enforced (e.g., gender of the doctor, language, age). Additionally, the maximum availability of the data can also be enforced by this technique. E.g., the patient can control how long the data can be decrypted, by defining a maximum lifetime of the data which is then part of the encryption. When this time-window expires nobody will be able to decrypt this data anymore. For Use-Case 2, this mechanism will be especially helpful for step 7 of the Fall Management Assistive Task.

5.3.3. DYNAMIC POLICY ADAPTATION

We assume the following case:

Alice is a Call Centre Operator who normally is not able to get Bob's location. In an emergency (e.g. during the execution of Fall Management AT), however, Alice should be able to retrieve it. The SM offers the possibility to dynamically adjust access rights very quickly through the policies. If Bob falls, the application changes the policy via the API of the Security Manager dynamically. Now it is possible for Alice to get the location (see Figure 39).





5.3.4. MONITORING COMPLIANCE

It is important that in complex systems we not only set policies and enforce them, but that we also –at leastperiodically monitor that the system is still conforming to the intended behaviour; this is especially true in a system where the policies will get dynamically updated to adapt to changes in the environment as supported by SEMIOTICS. In UC2 this dynamicity happens to adapt to medical emergency situations, in which case access is granted to aid in such emergency situations.

As an example, let us explain a bit more details about the Fall Management Assistive task of UC2. Here we allow logging of and access to a patient's geo location, in order to correctly assist the patient, but only in case the patient fell down (dynamic policy update).

In order to ensure compliance SEMIoTICS is capable, and we take advantage of this functionality in UC2, to check actively if the intended policies are enforced. This allows to detect suspicious access capabilities or changes to the policy which were not reverted. This self-audit capability is based on checking periodically the current access capabilities and compare them with expected values, and this checking is based on the SPDI patterns, here especially the security (S) and privacy (P) related patterns are of concern. To stay within the general storyline, there is a pattern on privacy that states that only patients normally have access to their location. Thus, in SEMIoTICS there is a periodic question to the security manager asking to list all entities who are not patients which have access to location data, and if this answer is not an empty list, then the privacy pattern is violated, and an alarm is raised. Of course, in emergency cases, like the fall of a patient, this default privacy setting is dynamically overruled. But, only for those patients that fell and only for the duration of the emergency. Thus, the monitoring component should not have to constantly alert the data protection officers.

So, by creating patterns that implement GDPR-compliant behaviour SEMIoTICS allows the systems administrators to enable the system to self-check itself.

5.3.5. ATTRIBUTE-BASED ENCRYPTION

As described in D4.12, the SEMIoTICS framework supports the use of Attribute-Based Encryption (ABE). The ABE is a relatively recent cryptographic approach that reconsiders the concept of public-key cryptography. For instance, in common public-key cryptography, a message is encrypted for a specific receiver using the receiver's public-key. Identity-based cryptography and in particular identity-based encryption (IBE) modified the established structure of public-key cryptography by changing the public-key in an arbitrary string, the email address of the receiver. The ABE goes one-step further and determines the identity not as an individual but as a set of attributes (user roles). Due to that, the encryption of the messages is achieved by using the subsets of attributes –key policy ABE (KP- ABE) – or the policies, which are defined over a set of attributes –ciphertext policy ABE (CP- ABE). In comparison with the IBE, the ABE has meaningful benefit, because it offers flexible one-to-many encryption instead of one-to-one; it is considered as a promising method to manipulate the issue of secure and fine-grained data sharing and decentralized access control.



In UC2, we applied this technique to the Gait Analysis Scenario, by encrypting the privacy sensitive data of the patient when sending it to the doctor/monitoring person during the exercise session of the patient. With the help of SEMIoTICS' Attribute-Based Encryption, the patient can determine his/her privacy requirements before the session starts. Those privacy requirements will then be translated into specific attributes, which will be used for encrypting the patient-related data.

If the doctor/monitoring person wants to access the patient's data, he/she first has to create a user-specific decryption key with the help of 1) the SEMIOTICS Security Manager (hosted at the premises of University of Passau), which also provides an Attribute-Based Encryption endpoint or 2) the locally deployed Attribute-Based-Encryption component, which is included in the SEMIOTICS framework as a dockerized component.

A decryption is possible if and only if the monitoring person's decryption key satisfies all privacy requirements that were previously defined by the patient, i.e. their attributes match, otherwise, no data will be revealed.



FIGURE 40: DEMONSTRATION OF ATTRIBUTE-BASED-ENCRYPTION IMPLEMENTATION ON THE ROLLATOR

For this specific use-case scenario, we decided to deploy the Attribute-Based Encryption component directly onto the rollator, which also determines the patient-related gait analysis data. When the rollator starts recording the patient's data (left side of Figure 40), the first step is to forward the data to the locally deployed Attribute-Based Encryption component. The ABE component then uses the previously defined privacy requirements of the patient to encrypt the data (right side of Figure 40) and transfer it back to the rollator, which then makes the data available for the monitoring person.



FIGURE 41: SUCCESSFUL DECRYPTION OF PATIENT-RELATED DATA



FIGURE 42: UNSUCCESSFUL DECRYPTION OF PATIENT-RELATED DATA



Afterwards, the monitoring person, can use the SEMIoTICS Security Manager or his/her own locally deployed Attribute-Based Encryption component to decrypt the data. We demonstrate this functionality with two fictional examples:

- 1. In Figure 41, we have Alice, a Doctor, 36 years old. She's been working at the Hospital in Rome for 5 years. On Monday at 8:30 am, during Bob's exercise session, she receives the encrypted data of Bob. With her user-specific Key, which she previously generated with the help of SEMIoTICS' Attribute-Based Encryption API, she tries to decrypt the received data, by sending a decryption request to the Backend Security Manager. The Security Manager then checks whether Alice's key satisfies all privacy requirements of Bob. As she's accessing the data on the same date, the requirement that the data should only be accessible for 7 days, is fulfilled. Furthermore, Alice is older than 30 years old and has been working at the hospital for more than a year. And the time of access is during Bob's exercise session. Therefore, the security manager can successfully decrypt Bob's data with Alice's key and forwards the data back to her. Alice can now work with Bob's data.
- 2. In Figure 42, we have Eve, a doctor, who is 29 years old and has been working at the hospital of Passau for roughly 6 months. On Monday at 8:30 am, during Bob's exercise session, she receives the encrypted data of Bob. As Alice did in the first scenario, she sends her key and the encrypted data of Bob to the Backend Security Manager. Even though she's accessing Bob's data during his exercise session, her personal key does not satisfy the privacy requirements of Bob. He clearly stated, that the doctor must be older than 30 AND must have been working at the hospital for at least a year. Therefore, the Security Manager fails at decrypting Bob's data and sends an error message to Eve. She's not able to access Bob's data.

A demonstration and further explanation of the software components and their deployment can be found in D4.12 or in the video about sub-use case 3 as part of the overall Security & Privacy functionality of SEMIOTICS.



6. SUB USE CASE 4: CONTROLLING BULBS AND ROBOTS

6.1. Scope and Objectives

This sub use case aims to demonstrate how the SEMIoTICS Semantic Interoperability technologies developed in the context of Task 3.3 can help the developers of the SARA solution to interface heterogeneous devices ranging from light bulbs to semi-autonomous robots.

As should be evident from the description of the Assistive Tasks given in section 2.1, in general, a SARA Assistive Task is realized through the interaction of various devices. As an example the Fall Management AT requires that the Robotic Assistant interacts with the lighting system of the Smart Home whilst the Gait Analysis AT requires the Doctor Web Application, interacts across the Internet, with a remote Robotic Rollator (Figure 43).



FIGURE 43: THE ROBOTIC ROLLATOR BEING USED BY THE SARA PROTOTYPE

In the Fall Management AT the Robotic Assistant needs to discover which are the lights available in the room where the possible fall event occurred and then request their activation to illuminate the scene. Once the available lights are discovered the interaction between the robot and the lights bulbs may occur using different protocols (e.g. ZigBee, Bluetooth, Z-wave). This need is reflected in the SARA requirements R.UC2,5 and R.UC2.11.

In the development of SARA there is the general requirement to keep the application logic independent from the underlying hardware as much as possible. As an example, in the context of the Gait Analysis AT, we wish to keep the application logic deployed on board of the Robotic Rollator independent from the specificities of the rollator used for the prototyping. This general requirement is reflected in the requirement R.UC2.6.



6.2. Interaction with SEMIoTICS framework

In relation to interconnection between different devices, and the requirements previously described, we followed the KPI-2.1, that it is the semantic descriptions for different types of smart objects, that we described on the D5.1. The semantic descriptions for all the types of smart objects are based on W3C Web of Things (WoT) standard; in particular we used the WoT Thing Description standardized format for describing IoT things. Each sensor, actuator or thing from all SEMIoTICS use cases were identified and for each smart object one Thing Description (TD) was provided. We used iotschema.org to semantically annotate each TD.

The goal was to enable smart objects to become interoperable. For the UC2 we focused on the sensors/actuators on board of the Robotic Rollator (RR), such as the Handlebar, Inertial Measurement Unit (IMU), LiDAR, Range Sensors (obstacle sensors) and Motorised Hub Wheels.

Figure 44 shows the SEMIOTICS Semantic Interoperability technologies involved in the implementation of the SARA solution.



FIGURE 44: SEMANTICS INTEROPERABILITY COMPONENTS IN SARA FIELD DEVICES

6.3. Setup testbed and integration

6.3.1. VIRTUALIZATION OF THE ROBOTIC ROLLATOR

The first step that we did to make the Robotic Rollator available as a "Thing" (in the sense of the WoT standard) it was to expose all the sensors onboard of the RR, through the CoAP protocol. This because CoAP is one of the protocol supported from the WoT (as described in details in D3.9). To do that, we implemented a CoAP server running on the Raspberry Pi that it is the device acting as Robotic Rollator Hub, i.e. in charge of controlling all the sensors/actuators. The Raspberry Pi used by the Robotic Rollator is a Raspberry Pi 3b with 1Gb RAM, 16GB SSD and Raspian OS.



pi@sara:~/CoAPthon \$ python coapserverJSONlegL.py
CoAP server started on 0.0.0.0:5555
['/obstacleC', '/obstacleL', '/', '/motorL', '/obstacleR', '/legL']

Following an example of REQUEST to the CoAP server previously shown:

```
pi@sara:~/CoAPthon $ python coapclient.py -o GET -p coap://127.0.0.1:5555/obstacleR
Source: ('127.0.0.1', 5555)
Destination: None
Type: ACK
MID: 5706
Code: CONTENT
Token: Qq
Payload:
value: 3.23 [m]
```

FIGURE 45: A SENSOR VALUE RETURNED BY THE COAP SERVER FO THE ROBOTIC ROLLATOR

After that, we created the node-wot servient; this one allows to show the RR as a Thing. To do that we utilized the ThingWeb's node-wot implementation of the WoT standard¹¹.

We create a TD for the RR, linking all the sensors to the related CoAP connection; so that, if we want the value of the obstacleR (obstacle sensor put on the right front on the RR), the node-wot servient provide a GET REQUEST to the related CoAP link, in this case "coap://127.0.0.1:5555/obstacleR".

To summarise we developed two main programs running on the RR:

- the CoAP server, that exposed all the sensors compliant with the CoAP protocol and
- the node-wot servient, that made the RR as a Thing in the WoT, for this reason we named it as "WoT_RR_SARA".

If both those programs are running properly on the RR, it is possible to open a browser in a PC connected on the same networks, pointing to the RR's ip address followed by the port 8080 (where the WoT Servient is listening for incoming requests) and all the sensors are been shown as a property of the RR (Figure 46).

	192.168.1.158:8080/WoT_RR_SARA/ - Chromium
	× +
$\leftarrow \rightarrow \mathbf{C}$ (i) Not secure	192.168.1.158:8080/WoT_RR_SARA/
{"@context ":"https://www.w3.org/2019/ A","security":["psk_sec"],	/td/v1","id":"urn:wotrrsara","title":"WoT_RR_SARA",' "securityDefinitions":{"psk_sec":{"scheme":"psk"}},'

```
{"obstacleL":
   {"description":"obstacleL","type":"application/json","readOnly":false,"writeOn
   e":false,"forms":
```

```
[{"href":"http://192.168.1.158:8080/WoT_RR_SARA/properties/obstacleL","content
son","op":["readproperty","writeproperty"]},
```

FIGURE 46: THE TD OF ROBOTIC ROLLATOR AS RETURNED BY THE WOT_RR_SARA SERVIENT

As it is possible to see from Figure 46, the RR is described as a Thing and exposed all the sensors following the iotschema.org, that it was our scope.

¹¹ <u>https://github.com/eclipse/thingweb.node-wot/</u>



If we wish to get the data of a single sensor, we need to add "/properties/obstaleR" to the completed link previously explained as following:

192.168.1.158:8080/Wo × + O Not secure | 192.168.1.158:8080/WoT_RR_SARA/properties/obstacleR С

"value: 3.08 [m]"

FIGURE 47: A SENSOR VALUE RETURNED BY THE WOT_RR_SARA SERVIENT

In addition to that, we deployed the Semantic-IoT-Gateway developed by Siemens in the context of the SEMIoTICS project. Using this component, it is possible to make the Robotic Rollator available as a node within a Node-RED instance running on the Robotic Rollator hub. This enable the possibility to use the Node-RED graphical environment to develop node.js applications that make use of the vitalized sensors and actuators of the Robotic Rollator.

In the Figure 48 it is shown a simple Node-RED flow that visualizes the value of the obstacleR sensor every 10 seconds by means of a GET request.

Rode-RED	× +			
$\left(\leftarrow ight) ightarrow $ C' \bigtriangleup	🗊 i 127.0.0.1:1880/#fla	pw/7ab22c18.fe0e8c	♥ ☆ 💷 🖽	۳ :
Node-RED			Deploy 🗸	
Q filter nodes	Flow 1	+	i ji debu i ji Q	, Lui
			▼ all nod	es 💼
 UndeFINed 			"value: 1.47 [m]"	
obstacleL obstacleC	obstacleR	msg.payload	<pre>7/3/2020, 11:19:19 AM node: b0430016.10f6c8 msg.payload : string[15] "value: 3.46 [m]"</pre>	
bobstacleR			7/3/2020, 11:19:29 AM node: b0430016.10f6c8 msg.payload : string[15] "value: 3.82 [m]"	
acceiz			7/3/2020, 11:19:39 AM node: b0430016.10f6c8 msg.payload : string[15] "value: 1.16 [m]"	¥
Wot ThingDescript ionGenerator FIGURE 48: A	SIMPLE NODE-RED FLO	W USING A SENSOR O	7/3/2020, 11:19:49 AM node: b0430016.10f6c8 msg.payload : string[15] "value: 0.88 [m]"	



To the date, due to the impossibility to access the actual Robotic Rollator because of the Covid-19 pandemics, all the steps previously listed in detail have been carried out with a Raspberry Pi simulator. All the validations steps will be repeated with the actual physical device as soon as it will be possible to access again the ENG premise where the Robotic Rollator is.

The data used for this first validation round comes from traces recorded with the actual rollator before the Covid-19 lock down.

6.3.2. VIRTUALIZATION OF ZIGBEE LIGHT BULBS

The Smart Home Hub is realized using a Raspberry 4. The connectivity with ZigBee devices enabled by an XBee module connected to the Raspberry via USB. The light bulb used for the validation is a Philips Hue lamp (Figure 49).



FIGURE 49: THE HARDWARE USED FOR THE VALIDATION OF THE SMART HOME HUB

The WoT Servient is implemented using the SANE Web of Things Servient¹². The SARA ZigBee Servient uses DigiKey's library¹³ to control the XBee module.

The SARA ZigBee Servient creates an http binding for the Hue hence it is possible to control the light bulb using standard REST call over http for turning the light on (e.g. http://10.0.1.29:8080/ZigBeeBulb/actions/on) and off (e.g. http://10.0.1.29:8080/ZigBeeBulb/actions/off).

6.3.2. VIRTUALIZATION OF SOFTBANK'S PEPPER ROBOT

The humanoid robot used for the validation is a SoftBank's Pepper Robot (Figure 50). A CoAP server enables the communication between the robot and the other components of SARA. The CoAP server is implemented using the Californium¹⁴ library. Differently from other components (e.g. Robotic Rollator) in the system the Robotic Assistant Hub does not host any of the SEMIoTICS Field components (e.g. Local Thing Directory, IoT Gateway). This design decision is motivated by the need to reduce as much as possible the drain of computational resources from the robot's processor. This requirements stems from the fact that the robot's processor is in charge for the execution of all the processes controlling the mechanics of the robot. An overload of the processor with extra process could have an impact on the responsiveness of the robot and the fluidity of its movements and, hence, in a worsen User Experience (UX).

¹² https://github.com/sane-city/wot-servient

¹³ https://www.digi.com/resources/documentation/digidocs/90001438/Default.htm

¹⁴ https://www.eclipse.org/californium/



The SARA Pepper CoAP server creates a CoAP binding for a Pepper robot and hence it is possible to observe current status robot's resources (e.g. speed of front wheel via the resource coap://192.168.1.10:5683/wheelFL/speed/sensor/value) and to activate/deactivate robot's behaviours (e.g. activate navigation behavior coap://192.168.1.10:5683/behaviour/navigation/start).



FIGURE 50: THE PEPPER ROBOT USED FOR THE VALIDATION OF THE SMART ASSISTANT HUB

The robot is not able to serve its own Thing Descriptor since it does not host a WoT compliant servient. Therefore in a typical SARA field configuration the TD of the robot is available from the Local Thing Directory deployed on the Smart Home Gateway. Below is shown a snippet of the TD describing the robot used for the SARA validation:

```
{
   "id": "urn:robotic-assistant-0",
   "title": "SARA Robotic Assistant",
   "@context": [
        "https://www.w3.org/2019/wot/td/v1",
        {
        "iot": "http://iotschema.org/",
          "schema": "http://schema.org/",
          "sara":"http://www.eng.it/sara/"
        },
        {
            "@language": "en"
        }
   ],
   "@type": "Thing",
   "security": [
```



```
"nosec sc"
],
"properties": {...}
"actions":{
    "startNavigation":{
        "@type":"StartNavigation",
        "iot:capability":"sara:MobileBase",
        "description":"Start navigation of robot",
        "idempotent": false,
        "safe": false,
        "forms":[{
                "href": "coap://192.168.1.10:5683/behaviour/navigation/start",
                "contentType": "application/json",
                "op": [
                     "invokeaction"
                ]
            }
        ]
    },
    "stopNavigation":{"@type":"StartNavigation",
        "iot:capability":"sara:MobileBase",
        "description":"Start navigation of robot",
        "idempotent": false,
        "safe": false,
        "forms":[{
                "href": "coap://192.168.1.10:5683/behaviour/navigation/stop",
                "contentType": "application/json",
                "op": [
                     "invokeaction"
                ]
            }
        ]
    }
}
```

}



7. SUB USE CASE 5: MONITORING ACTIVITIES OF DAILY LIFE

7.1. Scope and Objectives

This sub use case aims to demonstrate how the SEMIoTICS Monitoring, Prediction and Diagnosis component (MPD) can be used for the monitoring of Activities of Daily Life (ADL) performed by the elderly while at home. Examples of ADL include cooking, eating, sleeping, relaxing etc.. For the SARA solution understanding which activity is being performed by the patient is a key capability. As an example this information can be used to drive the interaction between the humanoid (Pepper) robot and the patient. In particular it can be used as indicator of the availability of the patient to interact and hence to reduce the risk that the patient refuses the social interaction.

In the context of the Gait Analysis Task this need occurs during the engagement of the patient (steps 2-4 described in section 2.1.2).

A Doctor uses the SARA Gait Analysis Web Application (introduced in the context of sub use case 1) to schedule a gait analysis session. The Figure 51 shows the page offered by the GA Web App to schedule a remote gait analysis session.

SARA - Doctor Con	sole			Welcome	ma
Home 🕇	Patient	Details			
YOUR PROFILE	Mario Rossi	Age:75			
Patients 🔒	< Gene	ral Info Contacts	Notes	Add Examinations	
Manage Examinations 🏚	Schedule N	ledical Examinations			
Account 💿	Type GAE - Gait Ana	lysis Examinat			
	From 01/10/2020	то 31/10/2020			
	Notes Observation of	how the			

FIGURE 51: THE SARA INTERFACE SUPPORTING THE SCHEDULING OF A REMOTE GA SESSION

It is important to note that, also as evidenced by the figure, at this stage the Doctor does not fix a precise date to run the session. Instead the Doctor fixes a range of dates (typically one week) when the examination should occur and the robotic assistant is delegated to engage the patient and remind him/her to fix a precise date.

In order to reduce the risk that patient refuses the interaction it is important for the robotic assistant to understand when it is the "right" way and time to successfully engage the user.

A basic awareness of the type of activity being carried on by the patient can help the robotic assistant's software to decide whether to start a successful social interaction. In fact we can expect that, depending on the patient, there will be activities that he/she is more prone to interrupt in order to interact with the robot.

This awareness can be created by using the SEMIoTICS Monitoring component to infer the kind of activity being carried on from the stream of events generated by the appliances within the smart home.



7.2. Interaction with SEMIoTICS framework

Figure 52 shows the SEMIOTICS components utilized for the scheduling of a Remote Gait Analysis AT.



FIGURE 52: SEMIOTICS COMPONENTS SUPPORTING THE SCHEDULING OF A REMOTE GA SESSION

The Doctor Web Application (already introduced for the sub use case 1 described in chapter 3) relies on the Human Task Manager to create and assign tasks to human agents. In the particular in the context of the Gait Analysis the Doctor Web Application uses the Human Task Manager to assign to the patient the task to fix date and time of the examination. The start deadline and the completion deadline of the task are set to the dates specified by the Doctor by means of the interface presented in Figure 52.

At the start date set for the task the Human Task Manager creates a notification for the patient. The notification is realized by writing a corresponding tuple within the Coordination Server deployed on the hub of patient's Smart Home.

The Coordination Server notifies the potentially interested clients about the availability of a new tuple representing the goal to finalize the examination scheduling. The potential clients could be the patient's smartphone, the caregiver's smartphone or the Robotic Assistant Agent. Each of the notified clients can decide to remove the tuple from the server indicating with this action that it is going to take care of the achievement of the goal. For the purposes of this section we consider only the case of the tuple goal being removed by the Robotic Assistant Agent.

The Robotic Assistant Agent is the software component controlling the activation/deactivation of the social behaviours of the robot. For the evaluation of the precondition of the user engagement behavior the Robotic Assistant agent relies on the Human-Robot Interaction (HRI) Manager residing within the SARA backend server. The RA Agent requests to the HRI Manager to be notified when the patient can be approached and the social interaction can start.



The HRI Manager uses the SEMIoTICS Monitoring component to infer the ADL being performed by the patient and, based on this, to decide whether the patient can be potentially interrupted by the robot interaction (i.e. the robot is likely to succeed with user engagement).

More specifically the HRI Manager submits to the Monitoring component a collection of queries each defining an ADL as a pattern over the stream of events generated by the Smart Home Servient (described in section 6.3.2).

The HRI Manager requests to the robot to approach the patient whenever the HRI Manager receives the notification that the patient is performing a potentially interruptible activity (i.e. whenever the Monitoring component matches one of the patterns defining an ADL). The HRI Manager makes the request towards the robot by writing a specific tuple within the Coordination Server.

The writing in the Coordination Server of a tuple representing the goal to approach the user, results in the Robotic Assistant Agent requesting the robot to move towards the user. The RA Agent starts the navigation towards the patient by activating the navigation behavior of the robot via the Robot CoAP Server.

7.3. Setup testbed and integration

The tested of sub use case 5 utilizes three hosts (see also Figure 52):

- **SARA server:** as in sub use 1 was the software components part of the SARA server and are deployed both on a standard personal computer and on a virtual machine (running Linux OS) available from the FiwareLab¹⁵ node operated by ENG in its data center located in Vicenza (Italy).
- **Humanoid robot**: it is a SoftBank's Pepper Robot. It is the same device utilized in sub use case 4. The configuration of this host is described in section 6.3.3.
- Smarthome hub: is Raspberry Pi v4. It is the same device utilized in sub use case 4.

The components presented in Figure 52 are implemented as follows:

- The Human Task Manager is a Springboot application exposing a REST API to manage human tasks. In this context a human task is a data structure describing an activity supposed to be performed by a human. A human task carries indication of the person in charge of the execution of the activity, the start and completion deadlines, as well as the escalation action to undertake in case deadlines are missed. Using the Human Task operations users can change the state of task (e.g. move the state from "not started" to "started"). The Human Task Manager takes care of the management of the lifecycle of a task: it monitors the state changes of a task and may initiate actions (e.g. send a reminder to the owner of the task if a deadline is missed).
- **Monitoring** is the SEMIoTICS Monitoring, Diagnosis and Prediction (MDP) component developed in the context of WP4 and described in deliverable D4.9.
- Human-Robot Interaction (HRI) Manager is a Springboot application. It runs various computational intensive algorithms driving the social interaction between robots and humans. These algorithms are accessible to robotic field devices via a REST API. For the purposes of the sub use case 5 it implements a basic algorithm that allows the humanoid robot to decide whether to initiate an interaction with a patient. The algorithm allows the humanoid robot to start an interaction only if it evaluates that the ADL ongoing within the smart home can be interrupted (i.e. that the probabilities that the user refuses to interact are low). The HRI Manager uses the SEMIOTICS monitoring component to monitor the ADL ongoing within the smart home. The HRI Manager submits a query for each type of ADL. For the experimentation we defined queries for "cooking", "relaxing", "housekeeping" and "leaving home".
- Robotic Assistant Agent: it manages the goals of the robotic assistant. Goals are represented as tuples
 within the CloE-IoT Coordination Service. The RA agent continuously monitors the Coordination Service
 and selects from it the next goal for the robot.

¹⁵ https://www.fiware.org/developers/fiware-lab/



Coordination Server offers the components of a CloE - IoT distributed application the capability to coordinate their activities using the tuple space (or blackboard) paradigm. Following this paradigm the various components of a distributed application undertake their specific activities driven by the availability of specific data (tuples) in a shared memory space (tuple space). Besides writing tuples in the shared space an application can read/remove tuples matching a tuple template. The operations to read/remove tuples from the tuplespace come in two flavours: blocking and non-blocking. Blocking operations blocks the control flow of the caller until an instance of the requested tuple appears in the tuplespace. The shared tuplespace can be accessed whether via a proprietary interface (Coordination) or an interface following the FIWARE NGSIv2 standard. In the latter case tuples are presented as contexts to the client application.

For the purpose of the experimentation we supposed the sensors were deployed within an apartment having the layout shown in Figure 53.



FIGURE 53: SUB USE CASE 5 VIRTUAL APARTMENT

The measurements of the apartment were chosen to be compatible with the demo room at ENG facility in Rome. The demo room was equipped with light panels simulating the walls of the apartment (Figure 54). These mock ups served the purpose to force both the robot and experimenters impersonating a user to follow as much as possible the trajectories they would have followed within the real apartment. This was important in order to generate sensors activation pattern as much as possible close to those generated in a corresponding real apartment.

For the SARA use case we assumed the smart home was able to generate their types of events:

- time events H0...H23 representing hours of the day
- location events (e.g. ENTRANCE, LIVING, KITCHEN) indicating the presence of a user in a specific area of the apartment
- powering events indicating that an appliance (e.g. light bulb, oven) is turned on/off
- opening/closing events indicating the opening/closing of drawers and doors (of cupboard, wardrobe, main entrance)





FIGURE 54: ENG DEMO ROOM EQUIPPED WITH MOCKUPS STAYING FOR WALLS

We considered a subset of the ADLs typically used in this kind of experimentation: "Leaving home", "Cooking", "Relaxing", "Taking medication", "Eating", "Sleeping", "Bathing", "Toilet".

We experimented the use of SEMIOTICS MPD predictive queries [9] to characterize each ADL in term of the events emitted by the smart environment.

We defined a type of event (e.g. "CookingEvent") for each of the ADL considered in the experiments (e.g. "Cooking"). The MDP notifies to the HRI one of these event whenever it infers the execution of a specific ADL is ongoing.

The query request to notify an event of type "Cooking" to the endpoint "http://saraserver:9090/hri/notifications" (i.e. the HRI Server) whenever the causal model used by the MPD let to infer that the activation of the location sensor in the kitchen (i.e. "http://homegateway:9090/kitchen") immediately followed by the ignition of the oven (i.e. "http://homegateway:9090/oven") is caused (with likelihood 70%) by an (undetected) event of type "PrepareMealTask" within the SARA Human Task Manager.

```
{ "id":3100,
  "events":[
     {"name":"CookingEvent",
     "eventsPattern";
     "name":"CookingEventPattern",
     "condition":{"type":"location",
          "source":{"wot":"http://homegateway:9090/kitchen"}
        },
        "next":{
            "next":{
            "name":"TurnOnOven",
            "condition":{"type":"switchOn",
               "source":{"wot":"http://homegateway:9090/oven"}
        },
        "causedBy":{
            "name":"PrepareMealTaskAssignment",
        }
    }
```





The COVID-19 pandemics did not allow using the ENG Demo Room to run the experiments imaged to discover and exploit the causal model needed by the MDP to infer ongoing ADL.

Therefore we decided to introduce an activity simulator within the architecture. Figure 55 shows how the simulator is connected to the rest of the architecture of this sub use case:



FIGURE 55: ARCHITECTURE EXTEND WITH SIMULATOR

The ADL Simulator uses the WoT Servient interface to operate on the (simulated) appliances managed by the Smart Home Servient. The operations of ADL Simulator have the effect that the Smart Home servient produces powering events towards the Monitoring component.



The ADL Simulator uses the Fiware NGSIv2 to publish the location of the simulated agent.



FIGURE 56: OUTPUT OF THE ADL SIMULATOR

The Monitoring component uses the WoT signaler to receive events from the Smart Home Servient and the Fiware signaler [9] to receive events from the Coordination Server. This configuration allowed verifying the capability of the MDP to fuse events from two different platforms (i.e. WoT and Fiware).

The ADL Simulator receives the notifications about the detected events by implementing the Event Listener interface. The ADL Simulator uses this notification to verify that the activity detected by the MPD matches that being simulated and compute the accuracy of the detection.



8. OVERALL KPIS AND REQUIREMENTS (ALL)

This section aims to provide a summary of requirements and KPIs relevant for evaluation in Use Case 2. The five presented Sub Use Cases, together with the remaining Use Cases 1 & 3 aim to address and validate the requirements presented in Table 1, while the relevant KPIs are presented in Table 5.

Further information on use case -specific requirements related to Use Case 2 are provided in D2.3. The original methodology of KPI evaluation contained in D5.1 was updated where required and is mostly summarized here for brevity.

8.1. Related KPIs

In this sub section, we list the SEMIoTICS 'KPIs related to the use case 2. TABLE 1 KPIS RELATED TO UC2

KPI ID	KPI description	Role in this KPI
KPI-1.1	Number of SPDI Patterns	SFC patterns and privacy patterns are used in this use case
KPI-1.2	Machine-processable language for pattern	The pattern language is used for processing patterns
KPI-2.1	Semantic descriptions for 6 types of smart objects	The description of the semantics related to this use case are also defined
KPI-2.2	Data type mapping and ontology alignment	The interoperability mechanisms are intervened in order to ensure that the data format and data alignment in use case 2.
KPI-2.3	Semantic interoperability with 3 IoT platforms	The development of a specific signaller was used to make the CloE-IoT platform observable by the SEMIoTICS
KPI-3.1.1	Generating monitoring strategies in the 3 targeted IoT platforms	The SARA use case monitors events generated by two IoT platforms: SEMIoTICS and CloE-IoT
KPI-3.1.2	Fuse results from these monitors	The query language of the SEMIoTICS MPD component allowed us to define cross-platform queries
KPI-3.1.3	Performing predictive monitoring with an average accuracy of 80%	The SEMIoTICS MPD Component supports both prediction and diagnostic queries.
KPI-3.2	Delivery of a monitoring language	The SEMIoTICS MPD Component provides a language to define high-level events that should be produced as result of the monitoring activity.
KPI-4.2	Delivery of mechanisms with adaptation time of 15ms	The developed SFC patterns were used as a sufficient adaptation mechanism to offer proactive adaptations as evaluated in this use case



KPI-4.3	Delivery of adaptations mechanisms enabling improvement by at least 20%	the developed SFC patterns were used as a sufficient adaptation mechanism to offer reactive adaptations by means of monitoring and intelligence analytics, as evaluated in this use case
KPI-4.6	Development of new security mechanisms/controls	From an NFV perspective, SFC is leveraged to guarantee security procedures for each kind of traffic in UC2. This is done by concatenating different security enforcers (firewalls, Intrusion Detection Systems, Honeypots) and forcing traffic to travers them. As each element is configured with specific security rules according to the expected traffic, only authorized packets are expected to go through to the services' endpoints.
KPI-5.1	Deployment of a multi-domain SDN orchestrator	Multi-domain SDN orchestration is supported in this use cases by the deployment of the SDN/NFV-enabled SEMIoTICS architecture in multiple-locations around different countries of Europe.
KPI-5.2	Service Function Chaining (SFC) of a minimum 3 VNFs	This KPI aims at the orchestration of SFC able to provide security by the chaining of at least 3 VNFs. That is, from a centralized position in the SEMIoTICS architecture, the NFV component should be able to build and configure the SFC for each kind of traffic.
KPI-6.1	Reduce manual interventions required for bootstrapping of smart object in each use case domain by at least 80%.	The bootstrapping service for the smart objects involves the availability of other functional blocks, e.g. at the networking level the VNFs and SFC that allow to forward the traffic from the smart objects towards the backend. An implementation of these functional blocks, at the networking level, in the form of VNFs automates its availability for the bootstrapping process. In this regard, note that the NFV MANO automatizes the creation of the VNFs and its depleted on the form of the NFV the form of the Allow is depleted.
		deployment on top of the NFVI in the form of VMs. Also, it automatizes the configuration, software installation and programs executions in the VM at boot time. Therefore, such operations are expected to eliminate user intervention completely.
KPI-6.2	Leveraging upon FIWARE assets in developing the SEMIoTICS framework	The SARA Health Scenario uses KNOWAGE applications as part of SEMIoTICS GUI
KPI-6.3	Delivery of 3 prototypes of IIoT/IoT applications	The SARA use case delivered a first prototype of the SARA solution empowered with SEMIoTICS technology
KPI-7.1	Provision the SEMIoTICS building blocks	The provision of the components that are used in SARA use case

8.1.1. Objective 1 - Security, Privacy, Dependability and Interoperability (SPDI) Patterns



8.1.1.1. KPI-1.1: Number of SPDI Patterns

The developed SPDI patterns are specified in a document format, and are detailed in the deliverable D4.8 – "SEMIoTICS SPDI Patterns (final)". In the context of UC2, SFC related patterns were used to support the dynamic instantiation of SFCs as well as privacy related. These patterns were preinstalled in the Backend Pattern Engine (as presented in the Section 4).

Apart from the SFC patterns, the Privacy pattern is modified in such a manner to take into account the facts modified by the Security Manager.

The when part of the rule specifies:

- 1. Two Placeholder that act as the two endpoints. The Placeholder is a parent class for Host, IoTSensor, IoTGateway etc, which enables the rule to be more abstract without the need to specify a new rule for every class.
- 2. A sequence of the Placeholder indicating their interaction.
- 3. Two properties on for each Placeholder
- 4. The orchestration property that can be guaranteed through the application of the pattern, i.e., the privacy property in this case (\$PR).

The **then** part modifies the status of the orchestration property to state that the property holds.

	rule "Privacy – Verification"
2	when
	Placeholder(\$p1Id:=placeholderid)
	<pre>Property (\$p1Id:=subject, category=="privacy", \$value1:=value, satisfied==true)</pre>
	Placeholder(\$p2Id:=placeholderid)
6	<pre>Property (\$p2Id:=subject, category=="privacy", \$value2:=value,satisfied==true)</pre>
	<pre>Sequence(\$sId:=placeholderid, \$p1Id:=placeholdera, \$p2Id:=placeholderb)</pre>
	<pre>\$PR: Property (\$sId:=subject, category=="Privacy", \$value1+\$value2<2,satisfied==false)</pre>
9	then
10	<pre>modify(\$PR){satisfied=true};</pre>
11	<pre>System.out.println("Privacy holds");</pre>
12	end

FIGURE 57: PRIVACY PATTERN

8.1.1.2. KPI-1.2 - Machine-processable language for pattern

The verification of delivery of the SEMIOTICS Pattern Language and its usability for defining machine-processable patterns is achieved through document review of deliverable D4.8 – "SEMIOTICS SPDI Patterns (final)". The verification of the format of the derived machine-processable patterns is achieved through verification of the correct functionality of a Drools rules reasoning engine.

In more detail, a Drools rule that encodes a pattern includes the inputs of the pattern's components and the required property in Left Hand Side (LHS). When the conditions in the LHS are satisfied, then the rule is fired to execute the actions as described in its Right-Hand Side (RHS). In the RHS, the new requirements of the compositions or atomic components can be inserted, updated or deleted.

In Table 2, a few semantics are presented related to SFC patterns. In the LHS, the components which constitute the topology of the pattern are defined. Different facts such as Virtual Service Functions and Service Function Chains are included in the list. Moreover, the Req represents the constraints of the topology and the required property. In the RHS, the pattern provides the solution by inserting, modifying, updating or retracting facts from the knowledge base which will also update the inventory list in the controller. Each component is converted through the respective Java class to an understandable format to the Pattern Engine.



Туре	Syntax	Description				
rule	rule "name"	name of the rule				
Left Ha	Left Hand Side (LSH)					
when	Pattern Elements (Facts)					
	Function (type, instantiated)	match Virtual Service Network Functions				
	Chain(functions)	match Service function chains				
	Req (src, dest, category, satisfied)	match requirements of pattern such as source, destination, property category and satisfied				
	Conditional Elements					
	==	match conditions				
	contains	contains object (logical)				
	not	not match (logical)				
	!=	not match (arithmetic)				
Right I	Right Hand Side (RSH)					
then	Actions					
	modify (\\$fact)\{pro=pro'\}	modify knowledge base fact				
	retract (\\$fact)	retract knowledge base fact				
	insert (new Fact ())	insert knowledge base fact				
	update (\\$fact)	update knowledge base fact				
	Java commands	other Java language syntax				

TABLE 2: PATTERN RULE CONSTRUCTS

8.1.2. Objective 2 - Semantic Interoperability

8.1.2.1. KPI-2.1 - Semantic descriptions for 6 types of smart objects

Semantic descriptions for all the types of smart objects have been provided based on W3C Web of Things (WoT) standard. In parallel, the iotschema.org was used to semantically annotate each TD. The analysis and description of semantic models with the TD for the sensors that were used in SEMIoTICS UCs have been presented in D4.11-"Semantic Interoperability Mechanisms for IoT (final)". Particularly, in case of UC2, the following Table 3 highlights the main properties, data types and actions for Things.



ABLE 3: DESCRIPTION O	F A SUBSET	OF SENSORS	ON BOARD	USED IN UC2
-----------------------	------------	-------------------	-----------------	--------------------

Sensor	Properties	Data Types	
	obstacleL (ObstacleSensors)	Meters	
Ultrasonic range	obstacleC (ObstacleSensors)	Meters	
	obstacleR (ObstacleSensors)	Meters	
Acceloromotor	Acceleration	Number (float)	
Accelei Omelei	Resolution	Number (float)	

The SEMIoTICS components that are involved in semantic description of UC2 sensors/actuators, are detailed and explained in subsection 6.2 - "Interaction with SEMIoTICS framework". An example of the updated version of JSON-LD for the UC2 is following:

```
{
   "id":"urn:robotic-rollator-0",
   "title":"SARA Robotic Rollator",
   "@context":[
      "https://www.w3.org/2019/wot/td/v1",
      {
         "iot":"http://iotschema.org/",
         "schema":"http://schema.org/",
         "sara":"http://www.eng.it/sara/"
      },
      {
          "@language":"en"
      }
   ],
"@type":"Thing",
`~.".[
   "security":[
      "nosec_sc"
   ],
    properties":{
      "legR":{
          "@type":"iot:Distance",
         "iot:capability":"sara:RightLegProximitySensing",
         "description":"Current distance of the user's right leg from the back side of the rollator.
",
         "observable":true,
         "readOnly":true,
         "writeOnly":false,
         "type":"integer",
         "schema:minValue":0.0,
         "schema:maxValue":1.5,
"iot:distanceUnitCode":"iot:Meters",
         "forms":[
            {
                "href":"coap://sara-rollator.ddns.net:5683/robotic-rollator-0/events/legR",
                "op":[
                   "subscribeevent"
                ],
                "contentType":"application/json"
            },
```



```
{
                "href":"http://sara-rollator.ddns.net:8081/robotic-rollator-0/events/legR",
                "op":[
                   "subscribeevent"
                ],
                "subprotocol":"longpoll",
"contentType":"application/json"
            }
         ],
"data":{
             "type":"number"
         }
     },
"legL":{
          "@type":"iot:Distance",
         "iot:capability":"sara:LeftLegProximitySensing",
         "description":"Current distance of the user's left leg from the back side of the rollator."
ر
         "observable":true,
         "readOnly":true,
         "writeOnly":false,
         "type":"integer",
"schema:minValue":0.0,
         "schema:maxValue":1.5,
         "iot:distanceUnitCode":"iot:Meters",
         "forms":[
            {
                "href":"coap://sara-rollator.ddns.net:5683/robotic-rollator-0/events/legL",
                "op":[
                   "subscribeevent"
                ],
                "contentType":"application/json"
            },
            {
                "href": "http://sara-rollator.ddns.net:8081/robotic-rollator-0/events/legL",
                "op":[
                   "subscribeevent"
                ],
                "subprotocol":"longpoll",
                "contentType":"application/json"
            }
         ],
"data":{
"+vpe
             "type":"number"
         }
      }
   },
   'events":{
      "onchange":{
         "forms":[
             {
                "href":"coap://sara-rollator.ddns.net:5683/robotic-rollator-0/events/onchange",
                "op":[
                   "subscribeevent"
                ],
"contentType":"application/json"
            },
            {
                "href":"http://sara-rollator.ddns.net:8081/robotic-rollator-0/events/onchange",
```


```
"op":[
                   "subscribeevent"
                ],
                "subprotocol":"longpoll",
                "contentType":"application/json"
             }
          ],
          data":{
             "type":"integer"
         }
      },
"legR":{
          "@type":"iot:Distance",
          "iot:capability":"sara:RightLegProximitySensing",
          "description": "Current distance of the user's right leg from the back side of the rollator.
",
          "observable":true,
          "readOnly":true,
"writeOnly":false,
          "type":"integer"
          "schema:minValue":0.0,
          "schema:maxValue":1.5,
          "iot:distanceUnitCode":"iot:Meters",
          "forms":[
             {
                "href":"coap://sara-rollator.ddns.net:5683/robotic-rollator-0/events/legR",
                "op":[
                   "subscribeevent"
                ],
"contentType":"application/json"
             },
             {
                "href":"http://sara-rollator.ddns.net:8081/robotic-rollator-0/events/legR",
                "op":[
    "subscribeevent"
                ],
"subprotocol":"longpoll",
":"applicatio
                "contentType":"application/json"
             }
          ],
          "data":{
             "type":"number"
          }
      },
"legL":{
          "@type":"iot:Distance",
          "iot:capability":"sara:LeftLegProximitySensing",
          "description":"Current distance of the user's left leg from the back side of the rollator."
ر
          "observable":true,
          "readOnly":true,
          "writeOnly":false,
         "type":"integer",
"schema:minValue":0.0,
          "schema:maxValue":1.5,
          "iot:distanceUnitCode":"iot:Meters",
          "forms":[
             {
                "href":"coap://sara-rollator.ddns.net:5683/robotic-rollator-0/events/legL",
```



```
"op":[
                   "subscribeevent"
                ],
                "contentType":"application/json"
            },
            {
                "href": "http://sara-rollator.ddns.net:8081/robotic-rollator-0/events/legL",
                "op":[
                   "subscribeevent"
                ],
                "subprotocol":"longpoll",
                "contentType":"application/json"
            }
         ],
          "data":{
             "type":"number"
         }
      }
   }
}
```

8.1.2.2. KPI-2.2 - Data type mapping and ontology alignment

The data type mapping and ontology alignment was described in D4.11. "Semantic Interoperability Mechanisms for IoT (final)" with three scenarios focusing on the use cases considered in SEMIOTICS, which include data flow between smart objects and are linked in the composition structure defined by the SPDI patterns in T4.1. Namely, for the UC2 the above procedure is included in phase that the Robotic Assistant should reach the location where the (possible) fall event occurred (D2.5 – Figure 21). This phase is initiated by the Smart Environment sending to the Robot the request to move to the location of the event. Before this step, the interoperability mechanisms, that are triggered by the corresponding pattern rule, are intervened in order to ensure that the data format for the location between UC2 Smart Environment and UC2 Robot is the same, in order to achieve the target of the system (for example use both Degrees Minutes Seconds (DMS) or Decimal Degrees). The SEMIOTICS components and the interactions between them that are involved in this technique are highlighted in the sequence diagram (see Figure 55).



FIGURE 58: UC2 DATA MAPPING APPROACH

Furthermore, the demonstration of data type mapping and ontology alignment is part of the sub use case 2: Automated, Trustworthy Healthcare Connectivity (see subsection 4.2.6). For instance, the SFC GUI uses the Thing Directory component to discover the available nodes that can be used as source and destination of a flow (end points). Before the instantiation of a service function chain, Pattern Engine component sends a request to Backend Semantic Validator component to verify the interoperability between the end points. When the interoperability between source and destination is not present, the Backend Semantic Validator informs the Pattern Engine so that the latter can take appropriate actions (see Figure 59).





FIGURE 59: SUB USE CASE 2 - INTEROPERABILITY MECHANISMS

8.1.2.3. KPI-2.3 - Semantic Interoperability with 3 IoT Platforms

As it was mentioned in D4.9 – "SEMIOTICS Monitoring, prediction and diagnosis mechanisms (final)", the development of a specific signaller was used to make the CloE-IoT platform observable by the SEMIOTICS Monitoring Component. In fact, the said signaller was leveraged by the SEMIOTICS Monitoring Component in order to observe events generated by the CloE-IoT Event Manager via the FIWARE NGSI v2 interface. Furthermore, SEMIOTICS TDs are serialized to JSON-LD, which enables JSON data to be interlinked and structured based on semantic (D3.9 – "Bootstrapping and interfacing SEMIOTICS field level devices (final)". At the same time, FIWARE introduced the NGSI-LD, which is an evolution of the FIWARE NGSI v2 information model and has been updated/improved to support linked data (entity relationships), property graphs and semantics (exploiting the capabilities offered by JSON-LD)¹⁶. This aforementioned link has given rise to further investigation of data mapping and the interconnection between FIWARE and SEMIOTICS semantic concept (Table 4), as it was analysed in D4.11 – "Deliverable D4.11 Semantic Interoperability Mechanisms for IoT (final)".

¹⁶ https://fiware-datamodels.readthedocs.io/en/latest/ngsi-ld_faq/index.html



TABLE 4: OVERVIEW	OF MAPPING	FIWARE DATA	MODEL TO	O IOTSCHEMA.O	RG
-------------------	------------	--------------------	----------	---------------	----

FIWARE Data Model	SEMIoTICS based on iotschema.org					
Entity	Thing Description					
ld of Entity	Id of Thing Description					
Type of Entity	Type of Thing Description					
Attributes	Properties					

8.1.3. Objective 3 – Monitoring Mechanisms

8.1.3.1. KPI-3.1 - Delivery of a Monitoring Management Layer

8.1.3.1.1.KPI-3.1.1 - Generating monitoring strategies in the 3 targeted IoT platforms

As presented in section 7 the SARA use case needs to monitor events generated by two IoT platforms: SEMIoTICS and CloE-IoT. In particulars SARA monitors the location of the patient using the Fiware NGSI v2 interface offered by the CloE-IoT Coordination Server. SARA also monitors the usage of the appliances (Field Devices) within the monitored apartment using the Web Of Thing (WoT) interface offered by the Smart Home servient developed in SEMIoTICS.

The SEMIoTICS Monitoring, Prediction and Diagnosis (MPD) components allowed monitoring both target platforms thanks to the Fiware signaler and the WoT signaler described in [9]. Figure 60 shows the messages confirming the registration of the two signallers within the MPD instance used for the SARA UC.

•••	sem-mdp — com.docker.cli ∢ docker run -p 8080:8080 sem-mdp — 209×38								
Last login: Sat Dec 19 12:32:53 on ttys000									
he default interactive shell is now xSh. © update your actount to use sth, please run `chsh -s /bin/zsh`. `or more details, please visit https://support.apple.com/kb/HT200950. otspotxPepper:sem-mdp presenza\$ docker run -p 8080:8000 sem-mdp]									
()									
2020-12-19 15:45:03.640 TMF0 1 [main] e.s.monitoring.s 2020-12-19 15:45:03.652 TMF0 1 [main] e.s.monitoring.s 2020-12-19 15:45:03.652 TMF0 1 [main] e.s.monitoring.s 2020-12-19 15:45:06.052 TMF0 1 [main] e.s.monitoring.s 2020-12-19 15:45:06.055 TMF0 1 [main] e.s.c.c.c.[Tener 2020-12-19 15:45:07.097 TMF0 1 [main] e.s.c.c.c.f.Tener 2020-12-19 15:45:07.097 TMF0 1 [main] e.s.c.c.c.f.Tener 2020-12-19 15:45:07.097 TMF0 1 [main] e.s.c.c.c.f.Tener 2020-12-19 15:45:07.097 TMF0 1 [main] e.s.web.centext.	pringboot.MpdSpringboot : Starting MpdSpringboot 1.0.0 on 547d63fb00bc with PID 1 (/home/sem-mdp-sb-1.0.0.jar started by root in /home) pringboot.MpdSpringboot : No active profile set, falling back to default profiles: default incarciant standbarries : Starting foreit (material) ina.core.Standbarries : Starting forvlet engine: [Apache Toncart/9.0.20] t.](localhost].[/] : Intializing Spring enbedded WebApplicationContext ContextLoader : Root WebApplicationContext: initialization completed in 3334 ms								
2229-12-19 15:45:67,742 INFO 1 main main us.semiotics.semiol.2020 2229-12-19 15:45:67,743 INFO 1 main main us.semiotics.semiol.2020 2209-12-19 15:45:67,743 INFO 1 main main us.semiotics.semiol.2020 2209-12-19 15:45:67,747 INFO 1 main main us.semiotics.semiol.2020 2209-12-19 15:45:68.000 INFO 1 main main us.semiotics.semiol.2020 2209-12-19 15:45:68.016 INFO 1 main main us.s.concurrent 2209-12-19 15:45:68.016 INFO 1 main main us.s.concurrent 2209-12-19 15:45:68.016 INFO 1 main main	<pre>itering.controller : Registered eu.seniotics.monitoring.springbot.services.RemotSignaller@523dd182 with key MDP. itering.controller : Registered eu.seniotics.monitoring.springbot.services.RemotSignaller@523dd182 with key MDPE. itering.controller : Registered eu.seniotics.monitoring.springbot.services.RemotSignaller@523dd182 with key FIRLD. itering.controller : Registered eu.seniotics.monitoring.signaller.fibare.FibareSignaller@523dd182 with key MDPE. itering.controller : Registered eu.seniotics.monitoring.signaller.fibare.FibareSignaller@518db1 with key MDST. itering.controller : Registered eu.seniotics.monitoring.signaller.fibare.FibareSignaller@518db1 with key MDST. itering.controller : Registered eu.seniotics.monitoring.signaller.vet.WoStignaller@518db1 with key MDST. itering.controller : Registered eu.seniotics.monitor.ps.jonaller.vet.WoStignaller@518db1 with key MDST. itering.controller : Registered eu.seniotics.monitor.ps.jonaller.vet.WoStignaller@518db1 with key MDST. itering.controller : Registered eu.seniotics.monitor.ps.jonaller.vet.WoStignaller@518db1 with key MDST. itering.controller : Registered eu.seniotics.monitor.ps.jonaller.ps.jonaller@518db1 with key MDST. itering.controller : Registered eu.seniotics.monitor.ps.jonaller.ps.jonaller@518db1 with key MDST. itering.controller : Registered eu.seniotics.monitor.ps.jonaller@518db1 with key MDST. itering.controller : Registered eu.seniotics.monitor.ps.jonaller.ps.jonaller@518db1 with key MDST. itering.controller : Registered eu.sen</pre>								

FIGURE 60: REGISTRATION OF FIWARE AND WOT SIGNALERS WITHIN THE MPD COMPONENT.

8.1.3.1.2.KPI-3.1.2 - Fuse results from these monitors

For the purpose of monitoring of ADLs the SARA solutions needs to match patterns defined over events generated by a Fiware source (the CloE-IoT Coordination Server) and a WoT source (the Smart Home servient).



The query language of the SEMIoTICS MPD component allowed us to define cross-platform queries [9]. Query 1 in section 7.3 is an example of the cross-platform queries used for the implementation of the ADL monitoring.

8.1.3.1.3.KPI-3.1.3 - Performing predictive monitoring with an average accuracy of 80%

The SEMIoTICS MPD Component supports both prediction and diagnostic queries [9]. Prediction queries produce an event whenever the observation of a pattern of events let to infer, with a given likelihood, that a monitored source is going to emit an event of a given type. Diagnostic queries produce an event whenever the observation of a pattern of events let to infer, with a given likelihood, that a monitored source emitted an (unobserved) event of a given type. Both type of queries are supported by a causal model that the MPD discovers by means of the observation of the streams of events generated by the monitored event sources. Both types of queries pose the problem of the accuracy of the inference.

In the context of SARA use case we experimented the use of diagnostic queries to infer the ADL ongoing within a Smart Home (see chapter 7). The measurement of the accuracy for the type of inference done by diagnostic queries was done using streams of events generated by a simulated environment using the setting shown in Figure 55. The use of a simulated environment is motivated both by the COVID 19 pandemic and the fact that the use of a real environment would have required long periods of observations.

The training of the model was done using a supervised approach: for each type of ADL we defined a possible sequence of events. Each sequence of events was prefixed with a special type of event (TaskAssignement) identifier of the ADL represented by the subsequent carrying the events (e.g. ["Cooking","Kitchen","SwitchOvenOn"]). The TaskAssignment event at the beginning of each sequence acted as label for the rest of the sequence. The model was discovered by playing back the sequences by means of the simulator.

The actual measurements were taken by playing back the sequences with the exception of their first event (i.e. the label) and verifying the matching between the event generated by the MPD and the label of the sequence being played back. Figure 56 presents an example of output produced by the ADL simulator during a measurement run.

In the context of the simulated experiments the average measured accuracy was 78.7% .

8.1.3.2. KPI-3.2 - Delivery of a monitoring language (ENG)

The SEMIoTICS MPD Component provides a language to define high-level events that should be produced as result of the monitoring activity [9].

In the context of SARA use case the MDP query language was used to define the queries characterizing an ADL in terms of the events produced by the Smart Environment described in chapter 7. The following query 2 is an example of a query used in the SARA use case. The query 2 is a predictive query describing the pattern of events generated during the execution of the "Leave home" activity. The query 1 in chapter 7 is another example of MPD query defined for the purposes of the SARA use case.

More in general, the MDP Query language was suitable to express all the queries needed for the purposes of the SARA use case.



```
"next":{
                "name":"Open wardrobe",
                "condition":{"type":"open",
                              "source":{"wot":"http://homegateway:9090/wardrobe"}
                        },
                "next":{
                    "name": "Move to entrance",
                    "condition":{"type":"location",
                          "source":{"fiware":"http://homegateway:9090/entrance"}
                        }.
                    "causedBy":{
                            "name":"GoForAWalkTaskAssignment",
                            "condition":{"type":"GoForAWalkTask",
                                        "source":{"source":"http://saraserver:9191/tasks/"},
                                        "likelihood":70.0
                                        }
                            }
                        }
                }
            }
        }
    }
],
"listeners":["http://localhost:8181/sara/securityalerts"]
                  QUERY 2: AN EXAMPLE OF SARA UC DIAGNOSTIC QUERY
```

8.1.4. Objective 4 – Multi-layered Embedded Intelligence

}

8.1.4.1. KPI-4.2 - Delivery of mechanisms with adaptation time of 15ms

To achieve the main goal of this KPI, the developed SFC patterns were used as a sufficient adaptation mechanism to offer proactive adaptations as evaluated in this use case. To accomplish the goal, the procedure begins from the time that a requirement is inserted until the triggering of the required adaptation steps toward the satisfaction of the requirement. To proceed to the evaluation of this KPI and investigate whether the goal of 15ms can be achieved, the time was measured under two different conditions: when there was the need to instantiate a chain and when both the creation and the instantiation of a chain was needed, based on the insertion of a requirement. The time was measured by monitoring the timestamps of the time of the requirement insertion and the time until the instantiated chain in OpenSource MANO. More specifically, the time of requirement insertion, adaptation triggering and complete was evaluated for the service function instantiation.

The first experiment is related to the instantiation of a chain when the descriptor exists. As can be seen in the Figure 61, when a chain descriptor exists, the required time from the verification of the requirement regarding the instantiation of a chain until the begin of the adaptation triggering is measured at 1ms, achieving the KPI for less than 15ms adaptation time. However, the time required to finalize the instantiation of the respective VNFs in the OpenStack is measured to almost 3sec, but this is related to the number of requested VNFs and resource capabilities of the devices holding the OpenStack.





Fact count is 14

The second experiment is focusing on the instantiation of a chain when the descriptor does not exist. As can be seen in the Figure 62, when a chain descriptor does not exist the required time from the verification of the requirement regarding the creation and the instantiation of a chain until the begin of the adaptation triggering is measured again at 1ms achieving the requirement for less than 15ms adaptation time. In addition, the time required to finalize the creation of the chain descriptor is less than 2ms. However, the instantiation of the respective VNFs is measured less than 3sec, but this is related again to the number of requested VNFs and resource capabilities of the devices holding the OpenStack.

FIGURE 61: CHAIN INSTANTIATION FROM AN EXISTING NETWORK DESCRIPTOR

Pattern Req	uirements							Topology	L. L.							
Application	Source	Destination		Service Functi	ions	Satisfied		Ø Edit		NS P	ackages					
GAIT Analysis	patient	doctor		kds firewall dj	tpi	True	Remove	Dector	Terra							
Fall Management	call_center	robot		load-balancer fir	reval	True	Remove		Al Services							
elect Application: WiT Analysis	Choose a patient	Source: Choo Desti	se a nation:	Choos Functi	se Service lons: 1-balancer	Submi	E.	00 Jan		Show	10 \$ entr	ies				
				_ids _free	llew			Lord Balance	freed	Short	Name 🏨	Ident	ifier 🕸	Description 1	Vendor 🕸	Version
Nodes		Functio	n Inst	ances	с	hains		Muble Access Pt	31	chain	-fall-load-	32aaf	011-ccb3-	chain	OSM	1.0
Node Name	IP Address	Name	Type		(Chain Name	Service Functions	Patent Home Gateway	Rand	Daian	cer-lirewali	4021- 556a	57890cc7			
patient	13.0.1.175	kds-VNF	kts	13.0.1.151		Chain1	Show Functions	000								
robot	13.0.1.214	freval-VNF	freval	13.0.1.152		Chain2	Phone Directions	Sneth	UTTR	chain	-gait-ids-	a0b5a	a2d1-87a9-	chain	OSM	1.0
doctor	13.0.1.237	dpi-VNF	dpi	13.0.1.153			STOW TO COOK			firewa	all-dpi	4a8e-	bf11-			
cal_center	13.0.1.170	load-balancer- VNF	load- balancer									coca	010/100			
2020-12-	15 02:28:	13.835	INFO	46195		[nio-9	080-exec-3] o.d.c.k.bui								
2020-12-	15 02:28:	13.945	INFO	46195		[nio-9	080-exec-3] o.d.c.k.bui	NS Inst	ances						
2020-12-	15 02:28:	14.168	WARN	46195		[nio-9	080-exec-3	<u>l o.a.maven</u> i								
Chain In	stance No	t Exists	:Tin	ne: 202	20-12	-15 02	2:28:21.079		Show 10	\$	entries					Se
Chain De	escriptor	Not Exis	ts :1	Time: 2	2020-	12-15	02:28:21.0	79								
Chain De	escriptor	Begin Cr	eatio	on :Tir	me: 2	020-12	-15 02:28:	21.08				1.0		Operational	Config	Detailed
Chain De	scriptor	Creation	Comp	oleted	:Tim	e: 202	0-12-15 02	:28:29.373	Name	12	Identifier	11	Nsd name	Status	Status	Status
Chain De	scriptor	Verified	:Tin	ne: 202	20-12	-15 02	:28:29.375		chain-fa	II-load-	7ab60e7c-2a8	∋-	chain-fall-loa	Id- running	configured	Done
Chain Be	gin Insta	ntiation	:Tin	ne: 202	20-12	-15 92	:28:29.376	ጎ 🗆	balance	-	4265-b1f6-		balancer-			-
Chain In	stantiati	on Compl	eted	:Time:	: 202	0-12-1	5 02:28:31	. 539	firewall		95469dbe014a		firewall			
Chain In	stance Ve	rified :	Time:	: 2020-	-12-1	5 02:2	8:31.54		chain-g	ait-ids-	e9b600ab-8e5	7-	chain-gait-id	S- running	configured	Done
	Cined.								firewall-	dpi	4684-be5f-		firewall-dpi			
0 roles	12100									- P -						

SEML

FIGURE 62: CHAIN CRETION AND INSTANTIATION

8.1.4.2. KPI-4.3 - Delivery of adaptations mechanisms enabling improvement by at least 20%

This KPI aims to improve past adaptations by means of monitoring and intelligence analytics. In order to accomplish this, the existing adaptations will be measured beginning from the time that a failure has been detected until the time that it was required to restore the above failure. The described methodology is based on two scenarios for achieving the required KPI. The first scenario is that, after a failure, there is no adaptation and this is the behavior prior to SEMIOTICS. In the second scenario there is an existing adaptation mechanism, which SEMIOTICS tries to improve upon. In the first scenario any future adaptation mechanism provided by SEMIOTICS will automatically imply 100% improvement given the fact that prior to SEMIOTICS there was no adaptation at all. In the second scenario the new mechanisms, will be measured by using appropriate metrics based on the use cases.

For the delivery of evolution mechanisms enabling the detection and analysis of the effects of past adaptations the improvement of future adaptations, SFC patterns were used as well. More specifically, the developed SFC patterns were used as a sufficient adaptation mechanism to offer reactive adaptations by means of monitoring and intelligence analytics, as evaluated in this use case. To accomplish the goal, the procedure begins from the time that an event such as a failure is required until the triggering of the required adaptation steps to the end of the restoration of the updated action. To do so, it was measured the time between two different conditions the detection of an event such a failure starting from the detection of the failure until the triggering of the instantiation of a new one. For instance, when a failure has been identified such as the failure of a service function, the result will be related to the not satisfaction of the requirement inserted in the past for end to end traffic forwarding through the respective chain, resulting the lack of end to end connectivity.

The first experiment includes the detection of a failure of an instantiated chain and the time required to instantiate a new one based on the existing descriptor. As can be seen in the Figure 63, when a chain instance is deleted, the failure is detected triggering the patterns to trigger the adaptation procedure based on the inserted requirement.



Are you	sure want to delete	chain-fall-loa	ad-balancer-firewall	17	Cancel OK	🖌 Home	NS In	2020- 2020- 2020-	12-15 02 12-15 02 12-15 02	2:22: 2:22: 2:22:	44.219 44.340 44.630	INF INF WAR	0 46 0 46 N 46	195 195 195	[n: [n: [n:	10-901 10-901 10-90	30-exe 30-exe 30-exe	c-9] o.d c-9] o.d	.c.+
Show 10 Name 11 chain-fail- load- balacer-	a ontries Identifier II - a3b97a39- 82af-45d5- a09d-	Nsd name II chain-tall- kad- balancer-	Operational Status	Config Status II Confectua	. Search Detailed Status	Actions	al New I	Chain Chain Chain Chain Chain Chain	Instance Instance Descrip Begin J Instante Instante	e Fa ce No otor insta iati	ilure [ot Exist Verific ontiatio on Comp crified)etec s :T ed :T on :T olete :Tim	tion ime: ime: ime: d :T e: 2	:Ti 202 202 202 ime: 020-	me: 202 0-12-19 0-12-19 0-12-19 2020-2 12-15 (0-12 02: 02: 02: 02: 12-15 02:22	 15 02 22:52. 22:52. 02:52. 02:22 55.64 	:22:52.4 413 414 414 2:55.66 52	12
firewall chain- gait-ids- firewall- dpi	c9c31474385d 33b31d29- 1bb2-4c0d- 95ff- 498575ca2098	firewall chain- gait-ida- firewall- dpl			Done	i A Actions *		6 rut Fact	es fired count is NStan	, 19 Ces	S								
NS Instan	ices						A New N		INSTANCE	0 + -			n	UTER	O LALNCH INSTA	NCE I	HLETE INSTA	MORE ACTIO	- 198
Show 10	entries				Searc	she		Dis	playing 7 items	in and				Protect	Availability	Pase	er Age	Actions	
		Nsd	Operational	Config				0	Instance Name	Name	IP Address	Flavor	Key Pair	starus	Zone	Stak			
Name III chain-fall- load-	Identifier a3b97a39- 82af-45d5-	Nsd name 11 chain-fall- load-	Operational Status 11 terminating	Config Status II terminating	Detailed Status	Actions i m	۵	0	Instance Name chain gait-idu-fee wait-dpi-3-dpi-VN F-1	Name sburnut 8-minim al	externalNet 172.113.40.38 afs 13.0.1.153	freesalt- VM-fre	Key Pair	Active	2014 0 1010	None Russ	ing 9 minute	s CREATE SNAPSHOT	r •
Name III chain-fall- load- balancer- firewall	Identifier 11 a3b97a39- 82af-45d5- a09d- c9c31474385d	Nsd name II chain-fail- load- balancer- firewall	Operational Status II terminating	Config Status II terminating	Detailed Status Stage 3/3 delete all. 0/1. Deleting ns from VIM.	Actions i m Actions •			chain gait-idu-fire wait-dpi-3 dpi-VN F-1 chain-gait-idu-fire wait-dpi-2 firemall- VNF-1	Name sbornut 8-minim al sbornut 8-minim al	P Address externalNet 172.113.40.38 afe 13.0.1.153 afs 13.0.1.152 externalNet 172.113.40.48	Flavor Sreenali- VM-By Sreenali- VM-By	Key Pair Isanabria Isanabria	Active Active	2014 0 nova 0 nova	None Run	ang Breinsta ing Breinsta	E CREATE SNAPSHOT	t +

FIGURE 63: CHAIN INSTANTIATION AFTER THE DELETION OF AN INSTANTIATED CHAIN

The second experiment includes both the deletion of a chain instance and the deletion of the respective descriptor procedure followed by the creation and the instantiation of a new one. As can be seen in the Figure 64, when a chain instance is deleted, the failure is detected triggering the patterns to trigger the adaptation procedure based on the inserted requirement.

Are you sure want to	delete chain-gait-ids	-firewall-dpi?		×	
				Cancel OK	2020-12-15 02:40:37.759 INFO 46646 [nio-9080-exec-1] o.d.c.k.b
				2	2020-12-15 02:40:37.874 INFO 46646 [nio-9080-exec-1] o.d.c.k.b
how 10 \$ entrie	¢)			\$	2020-12-15 02:40:38.141 WARN 46646 [nio-9080-exec-1] o.a.maven
	Nsd	Operational	Config		Chain Descriptor Failure Detection :Time: 2020-12-15 02:40:45.519
Name II Identifier	II name II	Status II	Status II	Detailed Status	Chain Instance Not Exists :Time: 2020-12-15 02:40:45.519
chain- 33b31d29	- chain-	Transfer 1	Configured	Done	Chain Descriptor Not Exists :Time: 2020-12-15 02:40:45.52
gait-ids- 1bb2-4c0	d- gait-ids-				Chain Descriptor Begin Creation :Time: 2020-12-15 02:40:45.521
ipi 498575ca	2098 dpi				Chain Descriptor Creation Completed :Time: 2020-12-15 02:40:49.267
·					Chain Descriptor Verified :Time: 2020-12-15 02:40:49.269
Are you sure want t	o delete chain-gait-k	ds-firewall-dpi?		×	Chain Begin Instantiation :Time: 2020-12-15 02:40:49 27
				Cancel OK	Chain Jegin Instantiation Completed Time: 2020-12-15 02:40.47.27
				Claincer Cit	chain instantiation completed .11me. 2020-12-15 02:40:51.432
					Chain Instance Verified :Time: 2020-12-15 02:40:51.434
Show 10 \$ ent	ries				7 rules fired
Short Name	Identifier I	7 Description	IT Vendor	Version II Acti	Fact count is 15
chain-gait-ids-	a0b5a2d1-87a9-	chain	OSM	1.0	
	c0cabd1a7706			4	
hosts	935(8aa5-aede-	hosts	OSM	1.0 4	

FIGURE 64: CHAIN CREATION AND INSTANTIATION WHEN A NETWORK DESCRIPTOR AFTER THE DELETION OF THE RESPECTIVE INSTANCE AND DESCRIPTOR

As can be observed by both experiments, the adaptation mechanism provided by SEMIoTICS imply 100% improvement given the fact that prior to SEMIoTICS, there was no adaptation at all in the described use case.



Finally, the capability for adaptation improvements based on patterns strengthen the importance of the implemented framework.

8.1.5. Objective 5 – IoT-aware Programmable Networks

8.1.5.1. KPI-5.1 - Deployment of a multi-domain SDN orchestrator

Multi-domain SDN orchestration is supported in this use cases by the deployment of the SEMIoTICS architecture in multiple-locations around different countries of Europe. More specifically, the network orchestration is applied through the NFV architecture located in CTTC premises in Barcelona that is able to enable traffic forwarding as described in this deliverable through the different end hosts in countries such as Italy and Greece. Moreover, the interconnection with the different components of SEMIoTICS architecture and backend services distributed in other countries and domains such as in Poland and Germany, provide a sufficient description of the multiple domain orchestration under this use case as can be seen in the Figure 65. Finally, the multi-domain orchestration in use case 2 can guarantee the TRL5 since all the involved components in the workflow including the SDN controller in OpenStack in combination with the respective VNFs can guarantee the TRL5 level requirement.



FIGURE 65: MULTI-DOMAIN USE CASE COMPONENT ORCHESTRATION AND DISTIRBUTION

8.1.5.2. KPI-5.2 - Service Function Chaining (SFC) of a minimum 3 VNFs

This KPI has been fulfilled, as we have commented above in sub use case 2. Namely, we have created SFCs that have three VNFs, which is onboarded and instantiated properly in the NFV testbed. Thereby, the NFV MANO component is able to build and configure the SFC for each kind of traffic properly. In the next figures, we show how the SFC have effectively three VNFs and that they are running in the NFV testbed.









- 8.1.6. Objective 6 Development of a Reference Prototype
- 8.1.6.1. KPI-6.1 Reduce manual interventions required for bootstrapping of smart object in each use case domain by at least 80%.

As commented above, the bootstrapping service for the smart objects involves the availability of other functional blocks, e.g. at the networking level the VNFs and SFC that allow to forward the traffic from the smart objects towards the backend.

The NFV MANO automatizes the creation and instantiation of the VNFs on top of the NFVI in the form of VMs. Also, it automatizes the configuration, software installation and programs executions in the VM at boot time. In the annex B we illustrate these automation processes thoroughly. Namely, we show how the OSM, automates the creation, onboarding and instantiation of the SFC and VNF, and it only requires that the user fills some configuration files. Thereby, the manual intervention is almost avoided. Moreover, as explained in the annex B, the cloud init files associated to the VNFs allow to define the software packages and the initial configuration of the VM that implements the VNFs. Thereby, the manual intervention is almost avoided as well.

8.1.6.2. KPI-6.2 - Leveraging upon FIWARE assets in developing the SEMIOTICS framework

The aim of this KPI was to analyse the General Enablers developed in the project FIWARE for their possible use in the SEMIoTICS project. The possibility study integration of the FIWARE General Enablers integration with the SEMIoTICS has been performed. The analysis included all FIREWARE components in total. The result of this



analysis is described in Deliverable 5.2 in section 4.2. In the SARA Health Scenario is used only one FIWARE General Enabler - Knowage. It is the open source business analytics suite that provides the creating various types of visualizations starting from simple tables, through many types of graphs ending on interactive maps. It offers many business analytics tools like periodic reporting, business predictions, and interactive cockpit. In the following figures you will find an example of KNOWAGE applications as part of SEMIOTICS GUI.



SEMI

GuiHub										
Toggle Sidebar		MyCock	pit MyCockpit						c c c c :	: ×
Cockpit Cockpit Smart Things SPDI Patterns	•		880000 - 90000 - 720000 - 640000 - 560000 - 480000 - 400000 - 240000 - 240000 - 160000 - 80000 - 0 -	MF		Clenca Manual Skilled Manual Professional Management	Clear selection 500,000 70,000 450,000 65,000 350,000 55,000 350,000 55,000 350,000 55,000 250,000 35,000 250,000 35,000 250,000 35,000 150,000 25,000 150,000 35,000 100,000 32,000	IS HILL Sales 240,000 - 20,000 - 20,000 - 20,000 - 180,000 - 180,000 - 140,000 - 140,000 - 100,000 -	ation horizon 2020 cole High track High track High track	•
				month	gender	sal	25	costs unit_s	ales num_customers	1
			05		м	68,708	27,512	32,742	10,512	
			02		M	69,020	27,668	32,675	10,457	
			01		M	71,033	28,414	33,658	10,748	
			07		F	76.870	30,832	36,934	11,526	
			05		F	70,844	28,356	33,360	10,658	
			03		F	75,599	30,210	35,680	11,374	
			04		м	67,585	27,001	31,950	10,222	
			09		м	70,161	28,058	33,143	10,579	
			10		F	67,698	27,075	31,981	10,208	
									1 to 10 of 24 $-$ 14 \leq Page 1 of 3 $>$ >1	
			Product an	alysis Custon	er analysis SEMIoTICS analysis	5				

8.1.6.3. KPI-6.3 - Delivery of 3 prototypes of IIoT/IoT applications

The SARA use case delivered a first prototype of the SARA solution empowered with SEMIoTICS technology. Sub use case 1 presents some of the GUI presented by SARA to the Doctor in the context of the Gait Analysis Assistive Task. The other sub use cases demonstrate how other SEMIoTICS are deployed within SARA.

8.1.7. Objective 7 – Promote the Adoption of EU Technology Offerings Internationally

8.1.7.1. KPI-7.1 - Provision the SEMIoTICS building blocks

This KPI will provide the key technological building blocks at TRL 6, 5 or 4. The following tables show the achieved TRL level of components used in SARA-Health Scenario.

Component	Layer	Baseline TRL	Target TRL	Current TRL	References
Backend orchestrator	Application	N/A (new)	6	6	D4.13
Backend Semantic Validator	Application	N/A (new)	4	4	Section 4.2.7, D4.13
GUI	Application	N/A (new)	6	6	D.4.13
Local embedded intelligence	Field	2	4	4	Section 3, D.4.13



Local thing directory	Field	6 (w/o adaptations)	6	6	D.4.13
Monitoring	Application	N/A (new)	4	4	D.4.13
NFV Orchestrator	NFV	6 (w/o adaptations)	6 (w/ adaptations)	6	Section 4.2.2, D3.8
Pattern Engine	Application	N/A (new)	4	4	Section 4.2.5, D.4.13
Pattern Orchestrator	Application	N/A (new)	4	4	Section 4.2.4, D.4.13
Security Manager	Field	4 (w/o adaptations)	5	5	D.4.13
Security Manager	Application	4 (w/o adaptations)	5	5	D.4.13
SFC Manager	SDN	6 (w/o adaptations)	6 (w/ adaptations)	6	D3.8, D3.11
Thing Directory	Application	6 (w/o adaptations)	6 (unchanged)	6	D.4.13
VIM Connector	SDN	6 (w/o adaptations)	6 (unchanged)	6	D3.7
Virtualized Infrastructure Manager	NFV	6 (w/o adaptations)	6 (w/ adaptations)	6	Section 4.2.3, D3.8
VNF Manager	NFV	6 (w/o adaptations)	6 (w/o adaptations)	6	Section 4.2.2, D3.8

8.2. Related framework requirements

In this section we consider the set of requirements that were described in deliverable D2.3 within the context of the SEMIoTICS's framework. Thereby, we list the subset of requirements from D2.3 that are relevant for this sub use case and we indicate in which part of the functional block architecture they are needed.

TADLE			
IABLE :	b REQUIREMENTS	RELAIDED	10 002

Req-ID	Description	Reference	Fulfilment
R.GP.2	Scalable infrastructure due to the fast- paced growth of IoT devices.	Section 4	The architecture tackled herein is highly scalable. Thanks to the NFV framework, more virtual computing and storage resources can be easily allocated when needed for the scalability purposes.



R.NL.1/ R.BC.1	Controller Node requirement: At least 6 CPU cores and 32 GB RAM.	Section 4	This is in line with the requirements that have already reported above e.g. the OpenStack controller or the OSM, see section 4.4. Note that, in addition, it is required that the controller nodes have around 100 GB of storage memory,
R.NL.2/ R.BC.2	Controller Node requirement: At least 2 Network interfaces.	Section 4	This is a requirement applicable to the computer holding the controller, e.g. the OpenStack controller.
R.NL.3/ R.BC.3	Controller Node Requirement: Linux OS.	Section 4	The controller nodes at the NFV MANO context, e.g. the OpenStack controller, require Linux OS.
R.NL.5/ R.BC.5/ R.BC.6/ R.BC.7	Hypervisor Nodes Requirement: At least 4 CPU cores and 8 GB RAM, at least 2, 1Gbps Network interfaces.	Section 4	This is fulfilled or in line with the requirements for the compute nodes of the NFVI, see section 4.4.
R.NL.6/ R.BC.8/ R.BC.9	Hypervisor Nodes: KVM and Linux Containers (LXD) must be supported by the Hypervisor Linux OS.	Section 4	This is supported in the compute nodes that form the NFVI, e.g. at the cloud level.
R.S.1	The confidentiality of all network communication MUST be protected using state-of-the-art mechanisms.	Section 3 and D3.7	The requirement was achieved in SDN connection with ovs switches by enabling SSL, and the Communication between Pattern Orchestrator and Pattern Engines with SSL.
R.S.2	Authentication and authorization of the stakeholders MUST be enforced by the Network controller, e.g. through access and role-based lists for different levels of function granularities (overlay, customized access to service, QoS manipulation, etc.)	Section 3 and D3.2	The Security Manager can provide authorization to the users entering the controller.
R.P.4	A short data retention period MUST be enforced and maintaining data for longer than necessary avoided	Section 5	Attribute-Based Encryption can be used for enforcing a maximum access-time to stored data.
R.P.8	Data MUST be stored in encrypted form.	Section 5	ABE achieve this requirement since it is used for encrypting user-specific data.
R.P.9	Repeated querying for specific data by applications, services, or users that are not intent to act in this manner SHALL be blocked.	Section 5	The Policy Enforcement Points (PEP) will be seeing all requests and can thus detect and mitigate such attempts.



R.P.13	The user SHALL be able to control the privacy mechanisms (i.e. redemption period, data granularity and dissemination, and anonymization technique)	Section 5	This can be enforced by using Attribute- Based Encryption as described in Sub-use case 3. The user is then able to enforce his/her privacy-concerns and wishes by encrypting the user specific data with the help of ABE.
R.GSP.9	The SARA system SHALL provide robust mechanisms to protect Patient-related data.	Section 5	This is fulfilled with the use of Security Manager as described in Sub-Use Case 3
R.GSP.10	The SARA system MUST fully comply with all relevant Italian laws governing the privacy, security and storage of sensitive Patient health-related data.	Section 5.3.4	SEMIOTICS Pattern Engines and Security Manager allows to implement patterns that implement GDPR-compliant behaviour.
R.UC2.1	The SEMIoTICS platform SHOULD support time- and safety-critical requirements by allowing SARA application logic to be deployed on resource-constrained edge gateways (e.g. smartphones, vehicles, mobile robots). SEMIoTICS platform functionalities SHOULD be locally available even in case of failure of communication with the SEMIoTICS cloud nodes.	Section 3 Section 6	SEMIoTICS TS Encoder is available locally on the Robotic Rollator Hub. SEMIoTICS Local Thing Directory and IoT Gateway is available locally on the Robotic Rollator Hub.
R.UC2.2	The SEMIoTICS platform SHOULD support the SARA solution to manage the trade-off between different requirements (e.g. reliability, power consumption, latency, fault-tolerance) by allowing both SARA application logic and platform features to be distributed over a cluster of gateways (SARA Hubs).	Section 4	This requirement is fulfilled with the use of SFCs and NFV.
R.UC2.3	The SEMIoTICS platform SHOULD guarantee proper connectivity between the various components of the SARA distributed application. The SARA solution is a distributed application not only because it uses different cloud services (e.g. AREAS Cloud services, AI services) from different remote computational nodes, but also because the SARA application logic itself is distributed across various edge nodes (SARA Hubs).	Section 3 Section 4	This requirement is fulfilled with the use of SFCs and NFV.
R.UC2.5	The SEMIoTICS platform should allow the SARA solution to discover the IoT devices that are registered in the system. IoT devices deployed by the SARA solution are expected to register themselves into the system using various standard protocols (e.g. LwM2M, MQTT, Bluetooth LE, ZigBee, etc.).	Section 6	All the sensors onboard of the RR, are exposed through the CoAP protocol and combined with a node-wot servient.



R.UC2.6	The SEMIoTICS platform SHOULD allow the SARA solution to retrieve the resources exposed by registered devices via their object model (i.e. a data structure wherein each element represents a resource, or a group of resources, belonging to a device). The SEMIoTICS platform SHOULD support at least the OMA LWM2M object model.	Section 6	The semantic descriptions for all the types of smart objects are based on W3C Web of Things (WoT) standard Each sensor, actuator or thing from all SEMIoTICS use cases were identified and for each smart object one Thing Description (TD) was provided.
R.UC2.11	The SEMIoTICS platform SHOULD allow a SARA component to request a group of devices to take/initiate an action (e.g. turn on/off a light bulb).	Section 6	The SARA ZigBee Servient creates an http binding for the Hue hence it is possible to control the light bulb using standard REST call over http for turning the light on.
R.UC2.12	The SEMIoTICS platform SHOULD allow SARA components to delegate to the platform the computation of complex functions over the data received by field devices. These computations may result either in the generation of higher-level observation events (e.g. significant Patient events abstracted form sensor data) towards the ACS or in sensors configuration parameters (including actuators command). The SARA components MAY specify computations either as Dataflow or as Finite State Machine.	Section 7	This requirement is fulfilled with the interaction of Human Task Manager, Human Robot Interaction Manager, Robotic Assistant Agent, Coordination Server and Monitoring Component.
R.UC2.14	The SEMIoTICS platform MAY allow the SARA solution to access information concerning the location of mobile robotic nodes and to maintain SARA specific location information (e.g. collected map data). This would allow the SARA hub or cloud to send to robotic devices high level navigation related commands (e.g. "go to location").	Section 7	This requirement is fulfilled with the use of the MDP queries.
R.UC2.15	The SEMIoTICS platform SHOULD provide low latency connectivity between the SARA hubs and cloud services (i.e. AREAS cloud services and AI services) to allow offloading of near real-time computation intensive tasks to the cloud.	Section 1, Section 4	This is related to the network management sub use case.
R.UC2.16	The SEMIoTICS platform SHOULD support secure bootstrap, configuration and diagnostic of SARA hubs and devices.	Section 5	In order to monitor that the system correctly enforces the intended behaviour, for this we describe how SEMIoTICS monitoring is used as a solution for periodical self-audit capabilities by monitoring for pattern-compliance.



9. CONCLUSION

9.1. Observed Obstacles

The most observed challenges for the implementation of this use case were related to the multi-domain deployment of the different components of the SEMIoTICS architecture that participated in SARA use case. There were constraints related to the private and public networking aspects realizing the implementation of this use cases closer to the reality of assistive living. Especially the current pandemic situation, which the project faced during the last year, pushed the use case 2 participants to work efficiently to overpass any constraint and achieve the overall scope and objectives of this use case.

9.2. Concluding Remarks

This deliverable presented the results of the validation of the SEMIoTICS technologies in the context of the development of SARA a solution in the domain of Ambient assisted living.

This document also identified the challenges subsumed by the implementation of two of the SARA assistive tasks (*Falls Management Assistive Task* and *Gait Analysis Assistive Task*).

Finally, this document presented the SEMIoTICS-based implementation of SARA by five points of view (sub use cases) each focusing one of main areas addressed by the SEMIoTICS project: (a) security, privacy, dependability and safety: (b) IoT semantic interoperability; (c) embedded intelligence and (d) automated trustworthy network management.



10. REFERENCES

- 1. ETSI, "ETSI.org: Network Functions Virtualisation (NFV); Architectural Framework," 2013. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf. [Accessed July 2020].
- 2. J. Serra et al. "Network Functions Virtualization for IoT (final)", SEMIoTICS deliverable D3.8, February 2020.
- 3. "Open Source MANO (OSM)" [Online]. Available: https://osm.etsi.org/. [Accessed July 2020].
- 4. "OpenStack" [Online]. Available: https://www.openstack.org/. [Accessed July 2020].
- 5. "OSM installation". [Online]. Available: https://osm.etsi.org/docs/user-guide/03-installing-osm.html. [Accessed July 2020].
- 6. "OpenStack Ansible, Train version" [Online]. Available: https://docs.openstack.org/openstack-ansible/train/. [Accessed July 2020].
- 7. F. Klement et al. "SEMIOTICS KPIs and Evaluation Methodology". SEMIOTICS deliverable D5.1, January 2020.
- 8. M. Falchetto et al. "Requirements specification of SEMIoTICS framework", SEMIoTICS deliverable D2.3, November 2019.
- 9. D. Presenza et al. "SEMIoTICS Monitoring, Prediction and Diagnosis Mechanisms (final)", SEMIoTICS deliverable D4.9, April 2020



APPENDIX A - NETWORK INFRASTRUCTURE VALIDATION

It is worth mentioning that a VPN has been configured at CTTC to let external users to access the NFV testbed described above. In Figure B.1, we show that the execution of the VPN connection instruction, based on a configuration file, is successful. Then, in Figure B.2 and Figure B.3, it is shown that the VPN allows to access the OSM and OpenStack GUIs, respectively. Moreover, below it will be shown, that the external VPN users can use properly the OSM and OpenStack services. To this end, a generic VNF and they corresponding Network Service (NS) will be created, onboarded to OSM and instantiated on top the NFVI, which is managed by the OpenStack. Observe that in OSM a VNF is always within the context of NS, which can be interpreted as a higher layer entity. In fact, a NS can contain multiple VNFs.

Thu Jun 25 16:50:11 2020 OpenVPN 2.4.4 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZ0] [LZ4] Thu Jun 25 16:50:11 2020 library versions: OpenSSL 1.1.1 11 Sep 2018, LZO 2.08 Thu Jun 25 16:50:11 2020 Incoming Control Channel Authentication: Using 256 bit message Thu Jun 25 16:50:11 2020 CPU/DP: Preserving recently used remote address: [AF_INET]84. Thu Jun 25 16:50:11 2020 Socket Buffers: R=[212992->212992] S=[212992->212992] Thu Jun 25 16:50:11 2020 UDP link local: (not bound) Thu Jun 25 16:50:11 2020 UDP link remote: [AF_INET]84.88.61.201:3000 Thu Jun 25 16:50:11 2020 NOTE: UID/GID downgrade will be delayed because ofclient, - Thu Jun 25 16:50:11 2020 VERIFY OK: depth=1, CN=SMARTECH CA Thu Jun 25 16:50:11 2020 VERIFY OK: depth=1, CN=SMARTECH CA Thu Jun 25 16:50:11 2020 VERIFY OK: depth=1, CN=SMARTECH CA Thu Jun 25 16:50:11 2020 VERIFY KU 0K Thu Jun 25 16:50:11 2020 VERIFY COM: depth=0, CN=openvpn-server Thu Jun 25 16:50:11 2020 VERIFY COM: depth=0, CN=openvpn-server Thu Jun 25 16:50:11 2020 VERIFY COM: depth=0, CN=openvpn-server Thu Jun 25 16:50:12 2020 SDRT COMTROL (Jopenvpn-server] PUSH REQUEST' (status=1) Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: reer-id set Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: reer-id set Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: dept-annel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: dept-annel crypto options modified Thu Jun 25 16:50:12 2020 DITONS IMPORT: dept-dannel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: dept-dannel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: datannel crypto options modified		jserr	a@js	erra-Latitu	ude-54	\$480:~\$ sudo openvpnconfig semiotics_vpn.ovpn
Thu Jun 25 16:50:11 2020 library versions: OpenSSL 1.1. 11 Sep 2018, LZO 2.08 Thu Jun 25 16:50:11 2020 Outgoing Control Channel Authentication: Using 256 bit message Thu Jun 25 16:50:11 2020 TCP/UDP: Preserving recently used remote address: [AF_INET]84. Thu Jun 25 16:50:11 2020 OCP/UDP: Preserving recently used remote address: [AF_INET]84. Thu Jun 25 16:50:11 2020 UDP link local: (not bound) Thu Jun 25 16:50:11 2020 UDP link local: (not bound) Thu Jun 25 16:50:11 2020 UDP link remote: [AF_INET]84.88.61.201:3000 Thu Jun 25 16:50:11 2020 UDP link remote: [AF_INET]84.88.61.201:3000, sid=0ce72 Thu Jun 25 16:50:11 2020 VERIFY 0K: depth=1, CN=SMARTECH CA Thu Jun 25 16:50:11 2020 VERIFY KU 0K Thu Jun 25 16:50:11 2020 VERIFY CK: depth=0, CN=openvpn-server Thu Jun 25 16:50:11 2020 VERIFY CKU 0K Thu Jun 25 16:50:12 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA38 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: there connection Initiated with [AF_INET]84.88 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: there and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: totic options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: totic options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjucting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjucting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjucting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjucting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: adjucting link_mtu to 1624 Thu Jun 25 16:50:12 2020 INTONI IMPORT: adjucting link_mtu to 1624 Thu Jun 25 16:50:12 2020 INTONI IMPORT: adjucting link_mtu to 1624 Thu Jun 25 16:50:12 2020 INTONI IMPORT: adjucting link_sc2.	ĺ	Thu J	Jun 2	5 16:50:11	2020	OpenVPN 2.4.4 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4]
Thu Jun 25 16:50:11 2020 Outgoing Control Channel Authentication: Using 256 bit message Thu Jun 25 16:50:11 2020 Incoming Control Channel Authentication: Using 256 bit message Thu Jun 25 16:50:11 2020 TCP/UDP: Preserving recently used remote address: [AF_INET]84. Thu Jun 25 16:50:11 2020 Socket Buffers: R=[212992->212992] S=[212992->212992] Thu Jun 25 16:50:11 2020 UDP link local: (not bound) Thu Jun 25 16:50:11 2020 UDP link remote: [AF_INET]84.88.61.201:3000 Thu Jun 25 16:50:11 2020 VDE: UID/GID downgrade will be delayed because ofclient, - Thu Jun 25 16:50:11 2020 VESI: Initial packet from [AF_INET]84.88.61.201:3000, sid=0ce72 Thu Jun 25 16:50:11 2020 VESI: Initial packet from [AF_INET]84.88.61.201:3000, sid=0ce72 Thu Jun 25 16:50:11 2020 VESIFY KU OK Thu Jun 25 16:50:11 2020 VERIFY KU OK Thu Jun 25 16:50:11 2020 Opentopn-server] Peer Connection Initiated with [AF_INET]84.88 Thu Jun 25 16:50:12 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256 GCM SHA38 Thu Jun 25 16:50:12 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256 GCM SHA38 Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REPLY,route 10.1.14.2 25 16.80.21,peer-id 4.cipher AES-256-CCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: ediguisting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 DUTIONS IMPORT: data channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: data channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 OUTIONS	1	Thu J	Jun 2	5 16:50:11	2020	library versions: OpenSSL 1.1.1 11 Sep 2018, LZO 2.08
Thu Jun 25 16:50:11 2020 Incoming Control Channel Authentication: Using 256 bit message Thu Jun 25 16:50:11 2020 TCP/UDP: Preserving recently used remote address: [AF_INET]84. Thu Jun 25 16:50:11 2020 UDP link local: (not bound) Thu Jun 25 16:50:11 2020 UDP link remote: [AF_INET]84.88.61.201:3000 Thu Jun 25 16:50:11 2020 UDP link remote: [AF_INET]84.88.61.201:3000 Thu Jun 25 16:50:11 2020 VEIFY UDI downgrade will be delayed because ofclient, - Thu Jun 25 16:50:11 2020 VEIFY OK: depth=1, CN=SMARTECH CA Thu Jun 25 16:50:11 2020 VERIFY UDI downgrade will be delayed because ofclient, - Thu Jun 25 16:50:11 2020 VERIFY UDI depth=1, CN=SMARTECH CA Thu Jun 25 16:50:11 2020 VERIFY UDI depth=1, CN=SMARTECH CA Thu Jun 25 16:50:11 2020 VERIFY UDI depth=0, CN=openypn-server Thu Jun 25 16:50:11 2020 UDI Control Channel: TLSV1.3, ctpher TLSV1.3 TLS AES 256 GCM_SHA38 Thu Jun 25 16:50:11 2020 [openypn-server] Peer Connection Initiated with [AF_INET]84.88 Thu Jun 25 16:50:12 2020 UDIINS IMPORT:riconfig/up options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT:riconfig/up options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:13 2020 /sbin/ip route add 1	1	Thu J	Jun 2	5 16:50:11	2020	Outgoing Control Channel Authentication: Using 256 bit message
Thu Jun 25 16:50:11 2020 TCP/UDP: Preserving recently used remote address: [AF_INET]84. Thu Jun 25 16:50:11 2020 UDP link local: (not bound) Thu Jun 25 16:50:11 2020 UDP link remote: [AF_INET]84.88.61.201:3000 Thu Jun 25 16:50:11 2020 UDP link local: (not bound) Thu Jun 25 16:50:11 2020 VOTE: UID/GID downgrade will be delayed because ofclient, - Thu Jun 25 16:50:11 2020 VERIFY OK: depth=1, CN=SMARTECH CA Thu Jun 25 16:50:11 2020 VERIFY KU OK Thu Jun 25 16:50:11 2020 VERIFY KK: depth=0, CN=openypn-server Thu Jun 25 16:50:11 2020 VERIFY KK: depth=0, CN=openypn-server Thu Jun 25 16:50:11 2020 VERIFY KK: depth=0, CN=openypn-server Thu Jun 25 16:50:11 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256 GCM_SHA38 Thu Jun 25 16:50:12 2020 EDRICY DK: depth=0, CN=openypn-server] Thu Jun 25 16:50:12 2020 EDRICY DKIC [openypn-server] 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REPLY,route 10.1.14.2 25 10.8.0.21,peer-id 4,cipher AES-256-CCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT:ifconfig/up options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: end options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OUTIONS I	1	Thu J	Jun 2	5 16:50:11	2020	Incoming Control Channel Authentication: Using 256 bit message
<pre>Thu Jun 25 16:50:11 2020 Socket Buffers: R=[212992->212992] S=[212992->212992] Thu Jun 25 16:50:11 2020 UDP link local: (not bound) Thu Jun 25 16:50:11 2020 UDP link remote: [AF_INET]84.88.61.201:3000 Thu Jun 25 16:50:11 2020 UDF Link remote: [AF_INET]84.88.61.201:3000, sid=0c72 Thu Jun 25 16:50:11 2020 VERIFY 0K: depth=1, CN=SMARTECH CA Thu Jun 25 16:50:11 2020 VERIFY KU oK Thu Jun 25 16:50:11 2020 VERIFY KW OK Thu Jun 25 16:50:11 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA38 Thu Jun 25 16:50:12 2020 [openvpn-server] Peer Connection Initiated with [AF_INET]84.88 Thu Jun 25 16:50:12 2020 OPITONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPIIONS IMPORT: toitconfig/up options modified Thu Jun 25 16:50:12 2020 OPIIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPIIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPIIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPIIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPIIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPIIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPIIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPIIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPIIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPIIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: data channel cipher 'AES-256-GCM' Initialized with 2 Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: data channel cipher 'AES-256-GCM' Initialized with 2 Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: data channel cipher 'AES-25</pre>	1	Thu J	Jun 2	5 16:50:11	2020	TCP/UDP: Preserving recently used remote address: [AF_INET]84.
Thu Jun 25 16:50:11 2020 UDP link local: (not bound) Thu Jun 25 16:50:11 2020 UDP link remote: [AF_INET]84.88.61.201:3000 Thu Jun 25 16:50:11 2020 NOTE: UID/GID downgrade will be delayed because ofclient, - Thu Jun 25 16:50:11 2020 VERIFY KU: depth=1, CN=SMARTECH CA Thu Jun 25 16:50:11 2020 VERIFY KU OK Thu Jun 25 16:50:11 2020 VERIFY CK: depth=0, CN=openvpn-server Thu Jun 25 16:50:11 2020 OPENTON Comparison of the server Authentication, ex Thu Jun 25 16:50:11 2020 OPENTON Comparison of the server of the server Authentication Thu Jun 25 16:50:12 2020 ONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: roitc options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: roitc options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 ROUTE GATEMAY 192.168.0.1/255.255.256.0CM' initialized with 2 Thu Jun 25 16:50:12 2020 ROUTE GATEMAY 192.168.0.1/255.255.256.0CM' initialized with 2 Thu Jun 25 16:50:12 2020 ROUTE GATEMAY 192.168.0.1/255.255.256.0CM' initialized with 2 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.30.0/16 via 10.8.0.21 Thu	1	Thu J	Jun 2	5 16:50:11	2020	Socket Buffers: R=[212992->212992] S=[212992->212992]
Thu Jun 25 16:50:11 2020 UDP link remote: [AF_INET]84.88.61.201:3000 Thu Jun 25 16:50:11 2020 NDTE: UID/GID downgrade will be delayed because ofclient, - Thu Jun 25 16:50:11 2020 TLS: Initial packet from [AF_INET]84.88.61.201:3000, sid=0ce72 Thu Jun 25 16:50:11 2020 VERIFY KU OK Thu Jun 25 16:50:11 2020 VERIFY KU OK Thu Jun 25 16:50:11 2020 VERIFY KU OK Thu Jun 25 16:50:11 2020 VERIFY EKU OK Thu Jun 25 16:50:11 2020 VERIFY EKU OK Thu Jun 25 16:50:11 2020 VERIFY KU OK Thu Jun 25 16:50:11 2020 VERIFY OK: depth=0, CN=openypn-server Thu Jun 25 16:50:11 2020 Control Channel: TLSv1.3, ctpher TLSv1.3 TLS_AES_256_GCM_SHA38 Thu Jun 25 16:50:11 2020 Control Channel: TLSv1.3, ctpher TLSv1.3 TLS_AES_256_GCM_SHA38 Thu Jun 25 16:50:12 2020 SENT CONTROL [openypn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 SENT CONTROL [openypn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 RUDTE_GATEWAY 192.168.0.1/255.255.255.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP tx queue length set to 100 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /s	1	Thu J	Jun 2	5 16:50:11	2020	UDP link local: (not bound)
Thu Jun 25 16:50:11 2020 NOTE: UID/GID downgrade will be delayed because ofclient, - Thu Jun 25 16:50:11 2020 TLS: Initial packet from [AF_INET]84.88.61.201:3000, sid=0ce72 Thu Jun 25 16:50:11 2020 VERIFY KU OK Thu Jun 25 16:50:11 2020 VERIFY EKU OK Thu Jun 25 16:50:11 2020 VERIFY OK: depth=0, CN=openypn-server Thu Jun 25 16:50:11 2020 VERIFY KU OK Thu Jun 25 16:50:11 2020 VERIFY OK: depth=0, CN=openypn-server Thu Jun 25 16:50:11 2020 Control Channel: TLSV1.3, cipher TLSV1.3 TLS_AES_256 GCM_SHA38 Thu Jun 25 16:50:12 2020 [openvpn-server] Peer Connection Initiated with [AF_INET]84.88 Thu Jun 25 16:50:12 2020 SENT CONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: peer-id set Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 DOTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 DOTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 ROUTE_GATEWAY 192.168.0.1/255.255.255.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 ROUTE_GATEWAY 192.168.0.1/255.255.255.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP tay queue length set to 100 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbi	1	Thu J	Jun 2	5 16:50:11	2020	UDP link remote: [AF_INET]84.88.61.201:3000
Thu Jun 25 16:50:11 2020 TLS: Initial packet from [AF_INET]84.88.61.201:3000, sid=0ce72 Thu Jun 25 16:50:11 2020 VERIFY KK: depth=1, CN=SMARTECH CA Thu Jun 25 16:50:11 2020 VERIFY KU OK Thu Jun 25 16:50:11 2020 VERIFY CK: depth=0, CN=openvpn-server Thu Jun 25 16:50:11 2020 VERIFY CK: depth=0, CN=openvpn-server Thu Jun 25 16:50:11 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_CCM_SHA38 Thu Jun 25 16:50:12 2020 ESNT CONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 ESNT CONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REPLY,route 10.1.14.2 25 10.8.0.21,peer-id 4,cipher AES-256-GCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 Intoming Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 INVTAP device tun0 opened Thu Jun 25 16:50:12 2020 INVTAP davice tun0 opened Thu Jun 25 16:50:13 2020 /Sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /Sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /Sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /Sbin/ip route add 10.1.13.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /Sbin/ip route add 10.1.13.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /Sbin/ip route add 10.1.3.0.1/32 via 10.8.0.21 Thu Jun 25 1	1	Thu J	Jun 2	5 16:50:11	2020	NOTE: UID/GID downgrade will be delayed because ofclient, -
Thu Jun 25 16:50:11 2020 VERIFY OK: depth=1, CN=SMARTECH CA Thu Jun 25 16:50:11 2020 VERIFY KU OK Thu Jun 25 16:50:11 2020 Validating certificate extended key usage Thu Jun 25 16:50:11 2020 VeRIFY EKU OK Thu Jun 25 16:50:11 2020 VERIFY EKU OK Thu Jun 25 16:50:11 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA38 Thu Jun 25 16:50:11 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA38 Thu Jun 25 16:50:12 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA38 Thu Jun 25 16:50:12 2020 SENT CONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REPLY,route 10.1.14.2 25 10.8.0.21,peer-id 4,cipher AES-256-GCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: oute options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: peer-id set Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: peer-id set Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: data channel cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: data channel cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 INCTAP TX queue length set to 100 Thu Jun 25 16:50:13 2020 JUN/TAP TX queue length set to 100 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21	ĺ	Thu J	Jun 2	5 16:50:11	2020	TLS: Initial packet from [AF_INET]84.88.61.201:3000, sid=0ce72
Thu Jun 25 16:50:11 2020 VERIFY KU OK Thu Jun 25 16:50:11 2020 Validating certificate extended key usage Thu Jun 25 16:50:11 2020 VERIFY EKU OK Thu Jun 25 16:50:11 2020 VERIFY EKU OK Thu Jun 25 16:50:11 2020 VERIFY OK: depth=0, CN=openvpn-server Thu Jun 25 16:50:11 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256 CCM_SHA38 Thu Jun 25 16:50:11 2020 [openvpn-server] Peer Connection Initiated with [AF_INET]84.88 Thu Jun 25 16:50:12 2020 SENT CONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REPLY,route 10.1.14.2 25 10.8.0.21,peer-id 4,cipher AES-256-GCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT:ifconfig/up options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 ROUTE_CATEWAY 192.168.0.1/255.255.255.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:13 2020 JuN/TAP device tun0 opened Thu Jun 25 16:50:13 2020 JuN/TAP device tun0 opened Thu Jun 25 16:50:13 2020 JuN/TAP device tun0 up mu 1500 Thu Jun 25 16:50:13 2020 /sbin/jp link set dev tun0 up mu 1500 Thu Jun 25 16:50:13 2020 /sbin/jp route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/jp route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/jp route add 10.1.32 via 10.8.0.21 Thu Jun 25 16:50:13	1	Thu J	Jun 2	5 16:50:11	2020	VERIFY OK: depth=1, CN=SMARTECH CA
Thu Jun 25 16:50:11 2020 Validating certificate extended key usage Thu Jun 25 16:50:11 2020 ++ Certificate has EKU (str) TLS Web Server Authentication, ex Thu Jun 25 16:50:11 2020 VERIFY EKU 0K Thu Jun 25 16:50:11 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA38 Thu Jun 25 16:50:11 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA38 Thu Jun 25 16:50:12 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA38 Thu Jun 25 16:50:12 2020 SENT CONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REPLY,route 10.1.14.2 25 10.8.0.21,peer-id 4,cipher AES-256-GCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: djusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: djusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: dist cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Untgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 TUN/TAP X queue length set to 100 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.32 via 10.8.0.21 Thu Jun 25 16:50:13 2020	1	Thu J	Jun 2	5 16:50:11	2020	VERIFY KU OK
Thu Jun 25 16:50:11 2020 ++ Certificate has EKU (str) TLS Web Server Authentication, ex Thu Jun 25 16:50:11 2020 VERIFY EKU OK Thu Jun 25 16:50:11 2020 VERIFY OK: depth=0, CN=openvpn-server Thu Jun 25 16:50:11 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_CCM_SHA38 Thu Jun 25 16:50:11 2020 [openvpn-server] Peer Connection Initiated with [AF_INET]84.88 Thu Jun 25 16:50:12 2020 SENT CONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REPLY,route 10.1.14.2 25 10.8.0.21,peer-id 4,cipher AES-256-CCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 Data Channel: cipher 'AES-256-CCM' initialized with 2 Thu Jun 25 16:50:12 2020 INCOMIG Data Channel: Cipher 'AES-256-CCM' initialized with 2 Thu Jun 25 16:50:12 2020 INUTAP tay 192.168.0.1/255.255.255.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.32 v	1	Thu J	Jun 2	5 16:50:11	2020	Validating certificate extended key usage
Thu Jun 25 16:50:11 2020 VERIFY EKU OK Thu Jun 25 16:50:11 2020 VERIFY OK: depth=0, CN=openvpn-server Thu Jun 25 16:50:11 2020 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA38 Thu Jun 25 16:50:11 2020 [openvpn-server] Peer Connection Initiated with [AF_INET]84.88 Thu Jun 25 16:50:12 2020 SENT CONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REPLY,route 10.1.14.2 25 10.8.0.21,peer-id 4,cipher AES-256-GCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: orte options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: orte options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: peer-id set Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 OUTGONG Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 INCTAP TX queue length set to 100 Thu Jun 25 16:50:13 2020 JUN/TAP TX queue length set to 100 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113		Thu J	Jun 2	5 16:50:11	2020	++ Certificate has EKU (str) TLS Web Server Authentication, ex
Thu Jun 25 16:50:11 2020 VERIFY OK: depth=0, CN=openvpn-server Thu Jun 25 16:50:11 2020 [openvpn-server] Peer Connection Initiated with [AF_INET]84.88 Thu Jun 25 16:50:12 2020 SENT CONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REPLY,route 10.1.14.2 25 10.8.0.21,peer-id 4,cipher AES-256-GCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OUTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 Duta Channel: using negotiated cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 ROUTE_GATEWAY 192.168.0.1/255.255.255.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.4.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.4.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.4.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.2.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.2.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.2.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip ro		Thu J	Jun 2	5 16:50:11	2020	VERIFY EKU OK
Thu Jun 25 16:50:11 2020 Control Channel: TLSV1.3, cipher TLSV1.3 TLS_AES_256_GCM_SHA38 Thu Jun 25 16:50:11 2020 [openypn-server] Peer Connection Initiated with [AF_INET]84.88 Thu Jun 25 16:50:12 2020 SENT CONTROL [openypn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REPLY,route 10.1.14.2 25 10.8.0.21,peer-id 4,cipher AES-256-GCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: dajusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: dajusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: dajusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OUtgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 INVTAP device tun0 opened Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 do_ifconfig, tt->did_ifconfig_ipv6_setup=0 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up at 10.8.0.221 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.3.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.3.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.3.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.3.0.1/32 via 10.8.0.21		Thu J	Jun 2	5 16:50:11	2020	VERIFY OK: depth=0, CN=openvpn-server
Thu Jun 25 16:50:11 2020 [openvpn-server] Peer Connection Initiated with [AF_INET]84.88 Thu Jun 25 16:50:12 2020 SENT CONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REPLY,route 10.1.14.2 25 10.8.0.21,peer-id 4,cipher AES-256-GCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 Data Channel: using negotiated cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 Data Channel: cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 OUTGING Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 INCOMIG Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 TUN/TAP 192.168.0.1/255.255.55.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 TUN/TAP TX queue length set to 100 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up tu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.3.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.3.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.3.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.3.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.3.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.3.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip ro		Thu J	Jun 2	5 16:50:11	2020	Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA38
<pre>Thu Jun 25 16:50:12 2020 SENT CONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1) Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REPLY,route 10.1.14.2 25 10.8.0.21,peer-id 4,cipher AES-256-GCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: oute options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: peer-id set Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel cripto options modified Thu Jun 25 16:50:12 2020 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 TUN/TAP TX queue length set to 100 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2</pre>		Thu J	Jun 2	5 16:50:11	2020	[openvpn-server] Peer Connection Initiated with [AF_INET]84.88
Thu Jun 25 16:50:12 2020 PUSH: Received control message: 'PUSH_REPLY,route 10.1.14.2 25 10.8.0.21,peer-id 4,cipher AES-256-GCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 Data Channel: using negotiated cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:13 2020 doifconfig, tt->did_ifconfig_ipv6_setup=0 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21		Thu J	Jun 2	5 16:50:12	2020	SENT CONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1)
<pre>10.8.0.21,peer-id 4,cipher AES-256-GCM' Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 Data Channel: using negotiated cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 ROUTE_GATEMAY 192.168.0.1/255.255.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 do_ifconfig, tt->did_ifconfig_ipv6_setup=0 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.22 peer 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.0.1.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21</pre>		Thu J	Jun 2	5 16:50:12	2020	PUSH: Received control message: 'PUSH_REPLY, route 10.1.14.2 25
<pre>Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: timers and/or timeouts modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT:ifconfig/up options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: geer-id set Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 Data Channel: using negotiated cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Route Cateway 192.168.0.1/255.255.255.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 do ifconfig, tt->did ifconfig jpv6_setup=0 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.3.0.12 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.3.0.12 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.3.0.12 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.3.0.13 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.3.0.13 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.3.0.13 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.3.0.13 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.3.0.13 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.3.0.13 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.3.0.13 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.3.0.13 via 10.8.0.21</pre>		10.8	3.0.2	1,peer-id 4	4,cip	ner AES-256-GCM'
<pre>Thu Jun 25 16:50:12 2020 OPTIONS IMPORT:trconfig/up options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: peer-id set Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 ROUTE_GATEWAY 192.168.0.1/255.255.05.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP TX queue length set to 100 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21</pre>		Thu J	Jun 2	5 16:50:12	2020	OPTIONS IMPORT: timers and/or timeouts modified
Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: route options modified Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: peer-id set Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 Data Channel: using negotiated cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 ROUTE_GATEWAY 192.168.0.1/255.255.255.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 TUN/TAP X queue length set to 100 Thu Jun 25 16:50:13 2020 do ifconfig, tt->did_ifconfig_ipv6_setup=0 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 108.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 108.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 108.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 UID set to nogroup Thu Jun 25 16:50:13 2020 UID set to nobody		Thu J	Jun 2	5 16:50:12	2020	OPTIONS IMPORT:itconfig/up options modified
Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: peer-id set Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 Data Channel: using negotiated cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 ROUTE GATEWAY 192.168.0.1/255.255.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 TUN/TAP TX queue length set to 100 Thu Jun 25 16:50:12 2020 do_ifconfig, tt->did_ifconfig_ipv6_setup=0 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.0.1.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21		Thu J	Jun 2	5 16:50:12	2020	OPTIONS IMPORT: route options modified
Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: adjusting link_mtu to 1624 Thu Jun 25 16:50:12 2020 OPTIONS IMPORT: data channel crypto options modified Thu Jun 25 16:50:12 2020 Data Channel: using negotiated cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 ROUTE_GATEWAY 192.168.0.1/255.255.05.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 do ifconfig, tt->did ifconfig_ipv6_setup=0 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.0.14.2 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.0.10.10.10.10.10.10.10.10.10.10.10.		lhu .	Jun 2	5 16:50:12	2020	OPTIONS IMPORT: peer-id set
<pre>Ihu Jun 25 16:50:12 2020 OPIIONS IMPORI: data channel crypto options modified Thu Jun 25 16:50:12 2020 Data Channel: using negotiated cipher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 Outgoing Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 ROUTE_GATEWAY 192.168.0.1/255.255.255.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP TX queue length set to 100 Thu Jun 25 16:50:12 2020 do_ifconfig, tt->di_ifconfig_ipv6_setup=0 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 (sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 (sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 (sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 (sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 Jbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 Jbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 Jbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 Jbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 Jbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 Jbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 Jbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 Jbin/ip route add 172.113.0.0/16 via 10.8.0.21</pre>		lhu .	Jun 2	5 16:50:12	2020	OPTIONS IMPORT: adjusting link_mtu to 1624
<pre>Ihu Jun 25 16:50:12 2020 Data Channel: using negotiated ctpher 'AES-256-GCM' Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipher 'AES-256-GCM' initialized with 2 Thu Jun 25 16:50:12 2020 ROUTE_GATEWAY 192.168.0.1/255.255.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 do_ifconfig, tt->did_ifconfig_ipv6_setup=0 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up rul 1500 Thu Jun 25 16:50:13 2020 /sbin/ip addr add dev tun0 local 10.8.0.221 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 GID set to nogroup Thu Jun 25 16:50:13 2020 UID set to nobody </pre>		lhu J	Jun 2	5 16:50:12	2020	OPTIONS IMPORT: data channel crypto options modified
Thu Jun 25 16:50:12 2020 Outgothg Data Channel: Cipher 'AES-256-0CM' initialized with 2 Thu Jun 25 16:50:12 2020 ROUTE_CATEWAY 192.168.0.1/255.255.05.0 IFACE=wlp2s0 HWADDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 TUN/TAP TX queue length set to 100 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip addr add dev tun0 local 10.8.0.22 peer 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 GID set to nogroup Thu Jun 25 16:50:13 2020 UID set to nobody		lhu .	Jun 2	5 16:50:12	2020	Data Channel: using negotiated cipher 'AES-256-GCM'
Thu Jun 25 16:50:12 2020 Incoming Data Channel: Cipner As-256-0CM initialized with 2 Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 TUN/TAP device tun0 opened Thu Jun 25 16:50:12 2020 tun/TAP TX queue length set to 100 Thu Jun 25 16:50:12 2020 do ifconfig, tt->did_ifconfig_ipv6_setup=0 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip addr add dev tun0 local 10.8.0.22 peer 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 GID set to nogroup Thu Jun 25 16:50:13 2020 UID set to nobody			Jun 2	5 16:50:12	2020	Outgoing Data Channel: Cipher "AES-256-GCM" initialized with 2
Thu Jun 25 16:50:12 2020 KOUTE_OALEWAY 192.168.0.1/255.255.255.0 TFACE=wtp250 HWAUDR=68 Thu Jun 25 16:50:12 2020 TUN/TAP TX queue length set to 100 Thu Jun 25 16:50:12 2020 do ifconfig, tt-sdid_ifconfig_ipv6_setup=0 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip addr add dev tun0 local 10.8.0.22 peer 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 GID set to nogroup Thu Jun 25 16:50:13 2020 UID set to nobody			Jun Z	5 16:50:12	2020	Incoming Data Channel: Cipner AES-256-GCM Initialized with 2
Thu Jun 25 16:50:12 2020 TUN/TAP device tune opened Thu Jun 25 16:50:12 2020 do_ifconfig, tt->did_ifconfig_ipv6_setup=0 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip addr add dev tun0 local 10.8.0.22 peer 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 108.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 (sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 (sbin/ip route add 108.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 UID set to noproup Thu Jun 25 16:50:13 2020 UID set to nobody		INU J	JUN Z	5 16:50:12	2020	RUUIE_GATEWAY 192.168.0.1/255.255.255.0 IFACE=WLD2S0 HWADDK=68
Thu Jun 25 16:50:12 2020 TOW/TAP TX queue tength set to 100 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip addr add dev tun0 local 10.8.0.22 peer 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 GID set to nogroup Thu Jun 25 16:50:13 2020 UID set to nobody		INU J	Jun 2:	5 16:50:12	2020	TUN/TAP device tune opened
Thu Jun 25 16:50:13 2020 do_rtointg, tt-sdtd_ttcinttg_type_setup=0 Thu Jun 25 16:50:13 2020 /sbin/ip link set dev tun0 up mtu 1500 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.22 peer 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 GID set to nogroup Thu Jun 25 16:50:13 2020 UID set to nobody		Thu J	Jun Z	5 16:50:12	2020	do if coofie tt odid if coofie inv(cotup-0
Thu Jun 25 16:50:13 2020 /sbin/ip addr add dev tuno local 10.8.0.22 peer 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 GID set to nogroup Thu Jun 25 16:50:13 2020 UID set to nobody		Thu J		5 16:50:12	2020	do_treaming, tr->dta_treaming_tpv6_setup=0
Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.22 peer 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.1.14.2/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 GID set to nogroup Thu Jun 25 16:50:13 2020 UID set to nobody		Thu J		5 16:50:15	2020	/sbin/ip tillk set dev tuno losal 10 0 0 22 page 10 0 0 21
Thu Jun 25 16:50:13 2020 /sbin/ip route add 172.113.0.0/16 via 10.8.0.21 Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 GID set to nogroup Thu Jun 25 16:50:13 2020 UID set to nobody		Thu J		5 16:50:15	2020	/sbin/ip add add dev tuno tocat 10.0.0.22 peer 10.0.0.21
Thu Jun 25 16:50:13 2020 /sbin/ip route add 10.8.0.1/32 via 10.8.0.21 Thu Jun 25 16:50:13 2020 GID set to nogroup Thu Jun 25 16:50:13 2020 UID set to nobody		Thu J		5 16:50:15	2020	/sbin/ip route add 172 112 0 0/16 via 10.8.0.21
Thu Jun 25 16:50:13 2020 GID set to nogroup Thu Jun 25 16:50:13 2020 UID set to nobody		Thu J		5 16.50.15	2020	/sbin/ip route add 10.2.115.0.0/10 vta 10.0.0.21
Thu Jun 25 16:50:13 2020 UID set to nobody		Thu _	1011 2. 1010 2	5 16.50.13 5 16.50.13	2020	
The Self 25 10.50.15 2020 OID Self to hobody		Thu		5 16.50.13 5 16.50.13	2020	UID set to nobody
The Jup 25 16:50:13 2020 WARNING: this configuration may cache passwords in memory u		Thu J		5 16.50.13	2020	WARNING: this configuration may cache passwords in memory u
Thu Jun 25 16:59:13 2020 Initialization Sequence Completed		Thu _	Jun 2	5 16:50:13	2020	Initialization Sequence Completed

FIGURE B.1 VPN THAT ENABLES EXTERNAL USERS TO ACCESS THE CTTC'S NFV TESTBED.



172.113.10.4/auth/?next=/		
=		
	Open Sour	rce MANO
	Log	gin
	Sign in to start	your session
	Username	1
	Password	
		Sign In
		Sign in
OSM		

FIGURE B.2 VPN ALLOWS ACCESS TO OSM AT CTTC'S NFV TESTBED.

172.113.10.10/auth/login/?next=/

	openstack.
Conec	tarse
Usuario	
Contraseñ	a
	۲
	Iniciar sesión

FIGURE A.3 VPN ALLOWS ACCESS TO OPENSTACK AT CTTC'S NFV TESTBED.



Next, as it was mentioned above, a generic VNF and its corresponding NS are created, onboarded to OSM and instantiated on top of the NFVI. This shows that the external VPN users can access properly the NFV testbed and use its NFV services, i.e. OSM and OpenStack. To this end, we use the OSM command line interface (cli). Thereby, first it is shown in Figure 6 that the external users can enter via ssh to the VM that contains the OSM.

<pre>jserra@jserra-Latitude-5480:~\$ ssh iotworld@172.113.10.4's password: Welcome to Ubustu 18 04 4 LTS (CNU)</pre>	iotworld@172.113.10.4)					
* Documentation: https://help.ubu * Management: https://landscap * Support: https://ubuntu.co	untu.com pe.canonical.com com/advantage)					
System information as of Thu Jun	25 14:54:54 UTC 2020						
System load: 0.68 Usage of /: 27.0% of 90.09GB Memory usage: 35% Swap usage: 0% Processes: 302	Users logged in: IP address for enp5s0: IP address for lxdbr0: IP address for docker0: IP address for docker_gwbridge:	0 172.113.10.4 10.181.16.1 172.17.0.1 172.18.0.1					
=> There is 1 zombie process. * "If you've been waiting for the macOS, the wait is over. Learn beam of the macOS.	perfect Kubernetes dev solution how to install Microk8s on macOS	for ."					
<pre>https://www.techrepublic.com/article/how-to-install-microk8s-on-macos/ * Canonical Livepatch is available for installation Reduce system reboots and improve kernel security. Activate at: https://ubuntu.com/livepatch</pre>							
13 packages can be updated. 0 updates are security updates.							
*** System restart required *** Last login: Thu Jun 25 11:26:01 202 iotworld@semiotics-osm-big:~\$	20 from 172.113.10.1						
	LEVALTUAT CONTAINO THE CO						

FIGURE B.4 SSH TO THE VM THAT CONTAINS THE OSM.

After entering to the VM that hosts the OSM, we will follow a set of steps to create, instantiate and terminate the generic VNF and the generic NS. Note that, the end user only needs to interact with the OSM, which communicates internally with the OpenStack. Moreover, the interaction with the OSM is through a set of configuration files that define the computing, storage and networking features of the VNF and the NS. These configuration files are so-called VNF descriptor (VNFd) and Network Service descriptor (NSd), respectively. Thereby, the next steps will be followed below to create, instantiate and terminate the VNF and NS.

- Create generic NSd and VNFd.
- Generate VNF/NS packages.
- Onboard the VNF/NS packages to OSM library.
- Instantiate the NS (and the VNF).
- Check that we can access the VM created for the VNF instantiation.
- Terminate the NS.



Create generic NSd (and VNFd).

Create VNFd, NSd folder structure

First, OSM needs to create a folder structure for the NSd, VNFd and its related files, such as the cloud init, which defines the initial configuration of the VM that will hold the VNF. This is accomplished by using a shell script provided by OSM, it is called "generate_descriptor_pkg". In Figure 7, we show how that folder structure is created correctly.

<pre>iotworld@semiotics-osm-big:~/osm/vnfd\$ generate_descriptor_pkg.sh -t vnfdimage ubuntu18-minimal -c generic iotworld@semiotics-osm-big:~/osm/vnfd\$ ls generic_vnfd iotworld@semiotics-osm-big:~/osm/vnfd\$ cd generic_vnfd/ iotworld@semiotics-osm-big:~/osm/vnfd/generic_vnfd\$ ls README charms cloud_init generic_vnfd.yaml icons images scripts iotworld@semiotics-osm-big:~/osm/vnfd/generic_vnfd\$</pre>	
<pre>iotworld@semiotics-osm-big:~/osm/nsd\$ ls iotworld@semiotics-osm-big:~/osm/nsd\$ generate_descriptor_pkg.sh -t nsd -c generic iotworld@semiotics-osm-big:~/osm/nsd\$ ls generic_nsd iotworld@semiotics-osm-big:~/osm/nsd\$ cd generic_nsd/ iotworld@semiotics-osm-big:~/osm/nsd/generic_nsd\$ ls README charms generic_nsd.yaml icons ns_config scripts vnf_config iotworld@semiotics-osm-big:~/osm/nsd/generic_nsd\$</pre>	

FIGURE B.5 CREATION OF THE FOLDER STRUCTURE NEEDED TO CREATE VNFd and NSd.

Edit VNFd

After creating the folder structure for the VNFd and NSd, we can edit the VNFd, which is a yaml configuration file. In Figure A.6 we present the snapshot of the yaml file that we have used to specify the VNFd. In the sequel, the important parts of this file are discussed. The tag "id" is the unique identifier for the VNF and it is important to recall it, as it is used in the NSd. The tag "mgmt-interface" is the interface over which the VNF is managed. Moreover, the "cp" within it just specifies the type of management endpoint, in our case "cp" means that we will use a connection point. Another important tag is the "vdu", which stands for virtual description unit, and it specifies the features of the VM that will host the VNF. The "vm-flavor" indicates the computing, memory and storage features of the VM that will host the VNF. Thereby, note that we define a VM with 1 virtual CPU, 1 GB of RAM and no persistent storage, as the "image" that we discuss next defines enough storage for this test. The tag "image" indicates the image that will be used to create the VM. The snapshot for this cloud init file is actually described below. The "interface" tag within the "vdu" tag specifies the interfaces for the vdu.

Cloud init file specification

This file is the one that specifies the initial configuration that we aim for the VM that will host the generic VNF. Note that its name is specified in the vnfd, as commented above. In Figure A.7, we provide the snapshot of this cloud init file and describe its functionalities. First, note that we have a field called "users". This allows to add users to the system. Note that we have added a user called "generic". Finally, within this user, there is an important field to be added, the "ssh_authorized_keys". This is important, because here we add the public ssh keys of the users that will access the VM that hosts the VNF.



vnfd:vnfd-catalog: vnfd: id: generic_vnfd
name: generic_vnfd short-name: generic_vnfd description: Generated by OSM package generator vendor: OSM version: '1.0' mgmt-interface: cp: vnf-cp0 # Atleast one VDU need to be specified vdu: id: generic_vnfd-VM name: generic_vnfd-VM description: generic_vnfd-VM count: 1 vm-flavor: vcpu-count: 1 memory-mb: 1024 storage-gb: 0 image: 'ubuntu18-minimal' cloud-init-file: 'generic' interface: # Specify the external interfaces name: eth0 type: EXTERNAL virtual-interface: type: VIRTIO external-connection-point-ref: vnf-cp0

FIGURE B.6 VNFd THAT DESCRIBES A GENERIC VNF.

Edit NSd

Next, in Figure B.8 we display a snapshot of the yaml file that we edited to specify the NSd. The relevant tags are described in the sequel. First, note that the tag "id" determines the unique identifier for the NS. The tag "constituent-vnfd" indicates which VNFs are part of the NS. In our case we have just the generic VNF, whose identification is "generic_vnfd" and is specified through the tag "vnfd-id-ref". Note that, in the vnfd the "id" tag has to correspond with that value. Then, we have the tag "vld", which is a description of the virtual links used by the NS for networking connections. In our case, note that the tag "type" is set to "ELAN". This indicates that the virtual link is a service to connect VNFs. The tag "mgmt.-network" set to "true" means that this is a VIM management network. The tag "vim-network-name" describes the name of the network in the VIM account, in our case "externalNet" is the name that was given such network in the OpenStack framework. Finally, the tag "vnfd-connection-point-ref" describes the connection points for the virtual links towards a vnf. We can see that this is a connection towards our generic VNF as the tag "vnd-id-ref" is set to "generic_vnfd".



GNU nano 2.9.3	generic
#cloud-config	
users:	
- default	
- name: generic	
lock_passwd: false	
<pre>sudo: ["ALL=(ALL) NOPASSWD:ALL\nDefaults:generic !requiretty"]</pre>	
passwd: \$1\$SaltSalt\$nQWEtCJCy/mlLI0pj15fd.	
shell: /bin/bash	
ssh_authorized_keys:	
 ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCuQr8qPujnrFSfp0AmMhMGpnylD1wsAKn+HL 	Jr9mY6RorsQ6
 ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDrlzTl2qA8ErEnaRW+zKHjDUDJ5Xhk2wrmel 	-C+S2ienq2rd
 ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQC4AcxbsG8P4H4sT5x9AcxnVpxoKB8sxXV3M 	/HCu+2xiuBGq
runcmu:	
- systet -w het.tpv4.tp_forward=1	

FIGURE B.7 CLOUD INIT FILE ASSOCIATED TO THE GENERIC VNF.

```
GNU nano 2.9.3
                                                                                                                                               generic nsd.yaml
nsd:nsd-catalog:
     nsd:
           id: generic_nsd
name: generic_nsd
           short-name: generic_nsd
description: Generated by OSM package generator
vendor: OSM
           version: '1.0'
           constituent-vnfd:
                # needs to be unique, sta
# vnfd-id-ref is the id of the VNFD
# Multiple constituent VNFDs can be specified
member-vnf-index: 1
vnfd-id-ref: generic_vnfd
           vld:
           # Networks for the VNFs
- id: generic_nsd_vld0
                 name: management
                 short-name: management
                 type: ELAN
                 mgmt-network: 'true'
                 vim-network-name: 'externalNet'
                # provider-network:
# overlay-type: VLAN
# segmentation_id: <update>
                 vnfd-connection-point-ref:
                 # member-vnf-index-ref - entry from constituent vnf
# vnfd-id-ref - VNFD id
                      member-vnf-index-ref: 1
                       vnfd-id-ref: generic_vnfd
# NOTE: Validate the entry below
                       vnfd-connection-point-ref: vnf-cp0
```

FIGURE B.8 NSd ASSOCIATED TO THE GENERIC VNF.



• Generate VNFd/NSd packages.

At this point, the VNFd and NSd are already edited. Thereby, we can generate the NSd, and VNFd packages, which are required in the onboarding process of the OSM. This means, the process of having our NS and VNF packages available in the OSM library. To generate those packages we only need to execute the shell script "generate_descriptor_pkg" provided by the OSM, which needs the NS and VNF folder structure that we created above in Figure 7.

<pre>iotworld@semiotics-osm-big:~/osm/vnfd\$ generic_vnfd iotworld@semiotics-osm-big:~/osm/vnfd\$ iotworld@semiotics-osm-big:~/osm/vnfd\$ generic_vnfd generic_vnfd.tar.gz iotworld@semiotics-osm-big:~/osm/vnfd\$</pre>	ls generate_descriptor_pkg.sh -t vnfd -N generic_vnfd/ ls
<pre>iotworld@semiotics-osm-big:~/osm/nsd\$ generic nsd</pre>	ls
iotworld@semiotics-osm-big:~/osm/nsd\$ iotworld@semiotics-osm-big:~/osm/nsd\$	<pre>generate_descriptor_pkg.sh -t nsd -N generic_nsd/ ls</pre>
<pre>generic_nsd generic_nsd.tar.gz iotworld@semiotics-osm-big:~/osm/nsdS</pre>	

FIGURE B.9 GENERATION OF VNFd AND NSd PACKAGES.

Onboard the VNF/NS packages to OSM library.

Next, in Figure B.10 it is demonstrated that we are able to onboard properly the NSd and VNFd packages into the library of OSM. Recall that these are a set of configuration files that describe the properties of our VNF and the requirements in terms of computing and networking that it has. Also, they describe the features of the VM that will host the VNF and the initial configuration and software packages installations that we need in the VM. We have called the NSd and VNFd as generic_nsd and generic_vnfd, respectively. It can be observed that OSM has onboarded properly the NSd and VNFd, as they appear on the list of available packages in the OSM library. Note that the osm instruction "osm nsd-list" and "osm vnfd-list" were used.

• Instantiate the NS.

Then, in Figure B.11 we trigger the instantiation of the NS that we have onboarded in OSM for our generic VNF. This means that OSM will communicate with the OpenStack controller, which creates the VM that will host our VNF on top of the virtual resources exposed by the OpenStack compute node. For that, obviously, the OSM takes into account the information embedded within the NSd and VNFd. Note that to trigger the NS instantiation we used the OSM command "osm ns-create –ns_name generic –nsd_name generic_nsd". Observe that you must specify for the nsd_name option the name of the NSd that you want to instantiate, otherwise the NS will not be instantiated.



iotworld@semiot	ics-osm-big:~/osm\$ osm nsd-list	
+	-+	
++	-+	
+	-+	
iotworld@semiot	<mark>ics-osm-big:~/osm</mark> \$ osm vnfd-list	
++- nfpkg name ++-	+ id +	
<pre>iotworld@semiot b0fe4c07-0c8f-4 iotworld@semiot 3de46d77-78f5-4 iotworld@semiot</pre>	+ ics-osm-big:~/osm\$ osm vnfd-create ./vnf b80-a5b8-25920ebd9538 ics-osm-big:~/osm\$ osm nsd-create ./nsd/ 5bd-8242-0597922b7ea7 ics-osm-big:~/osm\$ osm nsd-list	d/generic_vnfd.tar.gz generic_nsd.tar.gz
nsd name	id	
generic_nsd	3de46d77-78f5-45bd-8242-0597922b7ea7	
iotworld@semiot	ics-osm-big:~/osm\$ osm vnfd-list	
nfpkg name	id	
generic_vnfd	b0fe4c07-0c8f-4b80-a5b8-25920ebd9538	
iotworld@semiot	ics-osm-big:~/osm\$	

FIGURE B.10 ONBOARD VNFd AND NSd PACKAGES TO OSM.

iotworld@semiotics-	<mark>osm-big:</mark> ~\$ osm ns-list							
ns instance name +	ns instance name id date ns state current operation error details							
To get the history of all operations over a NS, run "osm ns-op-list NS_ID" For more details on the current operation, run "osm ns-op-show OPERATION_ID" iotworld@semiotics-osm-big:~\$ osm ns-createns_name genericnsd_name generic_nsd Vim account: semiotics_playground_train_vm_001 &fb2ac7a-53d4-4237-8a78-1d7c6e98324d iotworld@semiotics-osm-big:~\$ osm ns-list								
+ ns instance name	lid	date	ns state	current operation			error details	
generic	8fb2ac7a-53d4-4237-8a78-1d7c6e98324d	2020-06-25T20:10:17	BUILDING	INSTANTIATING (b295	5e6e5-b623-493b-a	ac53-3e3c9f277c99)	N/A	
+ + + + + + + + + + + + + + + + + + +								
ns instance name	id	date	ns state	current operation	error details	+		
- generic	8fb2ac7a-53d4-4237-8a78-1d7c6e98324d	2020-06-25T20:10:17	READY	IDLE (None)	N/A			
To get the history of For more details on iotworld@semiotics-of	of all operations over a NS, run "osm ns the current operation, run "osm ns-op-s osm-big:~\$	s-op-list NS_ID" show OPERATION_ID"						

FIGURE B.11 INSTANTIATE THE NS RELATED TO THE GENERIC VNF.

• Check that the we can access the VM created for the VNF instantiation.



The NS instantiation creates the VM that holds the VNF, on top of the NFVI. Thereby, it is important to check if we have access to such VM. In order to obtain the IP of the VM we can just execute the openstack command "openstack server list". OSM also provides this IP in the information of the NS instance. In Figure B.12 we show that we can access the VM that holds the generic VNF.

jserra@jserra-Latitude-5480:~\$ ssh generic@172.113.40.28 The authenticity of host '172.113.40.28 (172.113.40.28)' can't be established. ECDSA key fingerprint is SHA256:pKnp+EqPs+dWKAgQqsOvxnsOHwwCFvg8+EVD8Jong9c. Are you sure you want to continue connecting (yes/no)? yes Warning: Permanently added '172.113.40.28' (ECDSA) to the list of known hosts. Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-1053-kvm x86_64) * Documentation: https://help.ubuntu.com * Management: https://landscape.canonical.com * Support: https://ubuntu.com/advantage This system has been minimized by removing packages and content that are not required on a system that users do not log into. To restore this content, you can run the 'unminimize' command. 0 packages can be updated. 0 updates are security updates. The programs included with the Ubuntu system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright. Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law. generic@generic-1-generic-vnfd-vm-1:~\$ pwd /home/generic generic@generic-1-generic-vnfd-vm-1:~\$

FIGURE B.12 CHECK THE ACCESS TO THE VM CREATED TO HOLD THE GENERIC VNF.

• Terminate the NS

Finally, we show that we can terminate the NS, which shows the overall lifecycle of a NS and its associated VNFs. To this end, we use the OSM command "osm ns-delete" and its argument must specify the id of the NS instance that we want to terminate. In Figure B.13 we show that we can terminate the NS instance associated to the generic VNF.



iotworld@semiotics-	osm-big:~\$ osm ns-list						
ns instance name	i d	date	ns state	current operation	error details	í i	
generic	8fb2ac7a-53d4-4237-8a78-1d7c6e98324d	2020-06-25720:10:17	I READY	IDLE (None)	N/A	i i	
To get the history For more details on iotworld@semiotics- Deletion in progres iotworld@semiotics-	of all operations over a NS, run "osm n: the current operation, run "osm ns-op- osm-big:-\$ osm ns-delete 8fb2ac7a-53d4- s sn-big:-\$ osm ns-list	s-op-list NS_ID" show OPERATION_ID" 4237-8a78-1d7c6e98324d	1				
ns instance name	ld	date	ns state	current operati	on		error details
generic	8fb2ac7a-53d4-4237-8a78-1d7c6e98324d	2020-06-25T20:10:17	TERMINATI	IG TERMINATING (f2	52f714-eae9-4554	-b37c-db5043c5ff10)	N/A
To get the history For more details on iotworld@semiotics-	of all operations over a NS, run "osm n the current operation, run "osm ns-op- osm-big:-S osm ns-list	s-op-list NS_ID" show OPERATION_ID"					
ns instance name	ld date ns state current operat	lon error details					
+ To get the history For more details on iotworld@semiotics-	of all operations over a NS, run "osm n: the current operation, run "osm ns-op-: osm-big:-5	s-op-list NS_ID" show OPERATION_ID"					

FIGURE B.13 TERMINATE THE NS RELATED TO THE GENERIC VNF.

Validation of the NBI between NFV and OSS

The experiments that validate the NBI interface, which allows the connection between the NFV and external OSS, have been carried out thoroughly in SEMIOTICS' deliverable D3.8. Thereby, we refer the reader to D3.8 for full details on that validation.

Validation of the SFC capability in NFV

Next, we provide the experiments that validate the configuration of the SFC capability in the NFV testbed. Recall that SFC is enabled in OpenStack by installing and configuring the "networking-SFC" plugin for the OpenStack Neutron sub-block, see section 4.3.1.

First, in Figure B.14 we present the NSd that is used to carry out the SFC tests that validate the SFC capability in the NFV testbed. The NSd is presented in three segments to fit the figure in one page. The first segment is the top left figure, the second one is the top right and the last segment is the bottom one. The relevant tags of the .yaml file for the SFC purposes are described next.

- constituent-vnfd: Lists the VNF identifiers of the VNFs that belong to the SFC.
- ip-profiles: This is used to define an inner sub-network for the SFC. This means that the VNFs that belong to the SFC will communicate through this network. Their IPs are only visible to VM that belong to the network, i.e. they are not visible to other VM in the CTTC's NFV testbed.
- vld: Here the networks' link of the VNF's that belong to the SFC are defined. Note that there are two
 networks. One inner network, which is the one commented in the previous point. The other is an external
 network, i.e. it allows the connectivity to other VMs within the CTTC's NFV premises. It also allows the
 connectivity with other computers that are within the VPN context, though outside CTTC's premises.
- vnffgd: This is the tag that defines the behaviour of the SFC. The "classifier" tag defines which traffic is captured by the SFC and the ingress link, which is associated to a given VNF. In our case, the ingress is through the "sfc_generic_endpoint_vnfd" and we capture IP traffic whose source is 13.0.1.101 at port 2 and destination 13.0.1.102 at port 80. The "rsp" tag defines the order, in terms of the VNFs, that the traffic will follow to traverse the inner part of the SFC. In our case, we just consider one inner VNF, sfc_mpls_vnfd.



nsd:nsd-catalog: vld: # Networks for the VNFs - id: externalNet name: externalNet nsd: id: sfc_link_generic_nsd name: sfc_link_generic_nsd short-name: sfc_link_generic_nsd short-name: externalNet mgmt-network: 'true' description: Generated by OSM package generator type: ELAN vim-network-name: 'externalNet vim-network-name: 'externalNet' vnfd-connection-point-ref: wember-vnf-index-ref: 1 vnfd-id-ref: sfc_generic_endpoint_vnfd vnfd-connection-point-ref: vnf-cp0 vnfd-id-ref: sfc_generic_endpoint_vnfd vnfd-id-ref: sfc_generic_endpoint_vnfd vnfd-id-ref: sfc_mpls_vnfd vnfd-id-ref: sfc_mpls_vnfd vnfd-connection-point-ref: vnf-cp0 vendor: OSM version: '1.0' # Specify the VNFDs that are part of this NSD constituent-vnfd: member-vnf-index: 1 vnfd-id-ref: sfc_generic_endpoint_vnfd member-vnf-index: 2 vnfd-id-ref: sfc_generic_endpoint_vnfd - id: sfc_lan name: sfc_lan short-name: sf member-vnf-index: 3 vnfd-id-ref: sfc_mpls_vnfd ip-profiles: name: sfc_generic_ip_profile description: Inter VNF Link ip-profile-params: ip-version: ipv4 subnet-address: 13.0.1.0/24 dhcp-params: count: 50 enabled: true start-address: 13.0.1.100 dns-server: address: 8.8.8.8 vnffqd: - id vnffg1 name: vnffg1 description: vnffg1 short-name: vnffg1 vendor: vnffg1 version: '1.0 rsp: id: rsp1 name: rsp1 vnfd-connection-point-ref: member-vnf-index-ref: 3 order: 0 vnfd-egress-connection-point-ref: vnf-cp1 vnfd-id-ref: sfc_mpls_vnfd vnfd-ingress-connection-point-ref: vnf-cp1 classifier: id: class1 name: class1 member-vnf-index-ref: 1 rsp-id-ref: rsp1 vnfd-connection-point-ref: vnf-cp1 vnfd-id-ref: sfc_generic_endpoint_vnfd match-attributes: - id: HTTP

destination-ip-address: 13.0.1.102

source-ip-address: 13.0.1.101

destination-port: 80

ip-proto: 6

1.

short-name: sfc_lan
ip-profile-ref: sfc_generic_ip_profile
type: ELAN
provider-network:
- overlay-type: VXLAN
vin-network-name: 'sfc' vnfd-connection-point-ref: wnfd-id-ref: sfc_generic_endpoint_vnfd vnfd-id-ref: sfc_generic_endpoint_vnfd vnfd-connection-point-ref: vnf-cp1 ip-address: 13.0.1.101 member-vnf-index-ref: 2 vnfd-id-ref: sfc_generic_endpoint_ vnfd-connection-point-ref: vnf-cp1 ip-address: 13.0.1.102 member-vnf-index-ref: 3 vnfd-id-ref: sfc_mpls_vnfd undf_id-ref: sfc_mpls_vnfd endpoint_vnfd vnfd-connection-point-ref: vnf-cp1

source-port: 2 FIGURE B.14 NSD FOR THE SFC EXPERIMENTS IN THE NFV TESTBED.

In Figure B.15 we present the first experiment to validate the SFC capability in the NFV testbed. In this case, all the VMs are within the CTTC's NFV testbed. Namely, the source of traffic is considered to be the first VNF of the SFC, i.e. the one whose IP is 13.0.1.101 and the destination the last VNF i.e. the one whose IP is 13.0.1.102.



We generated the traffic using the netcat utility¹⁷, which allows to write and read from network connections using TCP. Thereby, at the destination we also use netcat to receive the transmitted TCP message. Moreover, we use the tcpdump utility¹⁸ to analyse the network traffic and see whether the traffic is captured by the middle VNF of the SFC. Figure B.15 shows how the TCP packet sent by the source is received at the destination. We can also see that the SFC captures the packet and it traverses the middle VNF.



FIGURE B.15 SFC EXPERIMENT IN THE NFV TESTBED, WHEN THE TRAFFIC SOURCE AND DESTINATION ARE WITHIN THE NFV TESTBED.

In Figure B.16, we did an experiment that is similar to the one of Figure B.15. However, in this case, the source of traffic and the destination are VMs that are within the NFV testbed but that do not belong to the SFC. The aim is to assess whether the SFC can receive and forward traffic that is generated outside the SFC. Note that scenario is closer to a real scenario where the NFV testbed is integrated with other SEMIoTICS' components. In order to forward the traffic that arrives to the SFC to a given destination we make use of the linux iptables¹⁹ utility. Namely, the next two shell command instructions are leveraged, which have been included in the cloud-init associated to the first VNF of the SFC for its automation, i.e. the one with IP 13.0.1.101.

- sudo iptables -t nat -A PREROUTING -p tcp --dport 2 -j DNAT --to-destination 13.0.1.102:80
- sudo iptables -t nat -A POSTROUTING -o ens4 -j MASQUERADE

¹⁷ https://linuxize.com/post/netcat-nc-command-with-examples/

¹⁸ https://www.tcpdump.org/manpages/tcpdump.1.html

¹⁹ https://help.ubuntu.com/community/lptablesHowTo



4. Note that these two instructions refer to the nat table, as iptables defines tables containing chains of rules for the treatment of packets and nat is the one that allows to change the source and/or target ip-address in packets. Also, observe that the nat table works by using two chain of rules, PREROUTING and POSTROUTING. The former is the chain where the packets enter before a routing decision is made. The latter is the chain where packets enter after the routing decision is made and before they are released to the hardware. In the PREROUTING command we say that all the TCP packets that arrive associated to port 2 we want to forward them to the IP 13.0.1.102 at port 80. Note that this is the last VNF of the SFC. The POSTROUTING command just forwards the outgoing packets through the network interface ens4.



FIGURE B.16 SFC EXPERIMENT IN THE NFV TESTBED, WHEN THE TRAFFIC SOURCE IS ANOTHER NS OF THE NFV TESTBED AND DESTINATION IS OUTSIDE CTTC NFV TESTBED.

The same rationale is applied at the last VNF of the SFC to forward the traffic to an external machine, a local PC in the VPN, whose IP is 192.168.169.100:



- sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.169.100:5000
- sudo iptables -t nat -A POSTROUTING -o ens3 -j MASQUERADE

In Figure B.16, we see how the traffic generated in the middle left figure passes through the first VNF of the SFC, on the top left. Then, it passes through the middle VNF of the SFC at the bottom, and finally leaves the SFC by traversing the last VNF of the SFC at the top right figure. Finally, it arrives to the external machine, as the middle right figure shows.

Finally, in Figure B.17 we have done a similar experiment than in Figure B.16. However, in this case the source of traffic is located at the IoT GW in the Engineering premises, i.e. outside CTTC premises. Similarly, the destination is a backend server located in the Engineering premises as well. This validates the end-to-end connectivity. In terms of the iptable commands, we used in the first VNF of the SFC the same commands than for Figure B.16, as we want to route the incoming traffic until the last VNF of the SFC. The iptable commands of the last VNF of the SFC are also similar to the ones of Figure B.16, we just changed the destination IP address where we want to route the traffic:

- sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 151.15.17.243:5000
- sudo iptables -t nat -A POSTROUTING -o ens3 -j MASQUERADE

In Figure B.17, on the top left, we show how the external traffic generated at Engineering arrives to the first VNF of the SFC, note that the traffic displays the external IPs of the SFC, i.e. 172.x.x.x. Namely, 172.113.40.1 is the IP of the external router of the NFV testbed. And 172.113.40.198 is the external IP of the first VNF of the SFC. Also, note in the bottom figure that the first VNF of the SFC chain forwards the traffic to the last VNF of the SFC. To show this situation we display the IP of the internal network of the SFC, i.e. 13.x.x.x. Finally, on the top right figure we show how the last VNF of the SFC routes the traffic to the backend server located at Engineering, whose IP and port are 151.15.17.243 and 5000, respectively. This figure also shows that the server at Engineering sends an ACK message, so it receives the traffic.



generic@sfc-fwd-1-sfc-generic-endpoint-vnfd-vm-1:~ Archivo Edtar Ver Buscar Terminal Ayuda win 2048, options [nop,nop,15 val 1/2291016 ecr 769757462], length 165: HTTP 10:54:52.581718 fa:16:3e:49:b8:19 > fa:16:3e:35:e2:7a, ethertype IPv4 (0x0800), length 66: 13.6.1.101.49000 > 13.0.1.102.80: Flags [.], ack 37, win 502, options [nop,nop,T5 val 76975769 ecr 172291016], length 0 10:54:52.65266 fa:16:3e:49:b8:19 > fa:16:3e:35:e2:7a, ethertype IPv4 (0x0800), length 66: 13.6.1.101.49000 > 13.0.1.102.80: Flags [.], ack 203, win 561, option s [nop,nop,T5 val 769757768 ecr 172291016], length 0 10:54:53.113733 fa:16:3e:49:b8:19 > fa:16:3e:35:e2:7a, ethertype IPv4 (0x0800), length 74: 13.6.1.101.49004 > 13.0.1.102.80: Flags [5], seq 38819413, win 64240, options [nss 1356,asckok,T5 val 76975827 ecr 0,nop,wscale 7], length 0 10:54:53.192503 fa:16:3e:49:b8:19 > fa:16:3e:35:e2:7a, ethertype IPv4 (0x0800), length 74: 13.6.1.101.49004 > 13.0.1.102.80: Flags [5], seq 4819556864 ck 888 19414, win 65535, options [mss 1358,nop,wscale 6,nop,nop,T5 val 172291702 ecr 76 9758227,sack0K,eol], length 0 10:54:53.273972 fa:16:3e:49:b8:19 > fa:16:3e:35:e2:7a, ethertype IPv4 (0x0800), length 76: 13.6.1.101.49004 > 13.0.1.102.80: Flags [.], ack 1, win 592, options [nop,nop,T5 val 76975839] ecr 172291702], length 0 10:54:53.200769 fa:16:3e:49:b8:19 > fa:16:3e:35:e2:7a, ethertype IPv4 (0x0800), length 66: 13.6.1.101.49004 > 13.0.1.102.80: Flags [P.], seg 12:14, ack 1, win 592, options [nop,nop,T5 val 769758392 ecr 172291702], length 1313: HTTP: 10:54:53.310076 fa:16:3e:49:b8:19 > fa:16:3e:35:e2:7a, ethertype IPv4 (0x0800), length 1379: 13.0.1.101.49004 > 13.0.1.102.80: Flags [P.], seg 12:14, ack 1, win 592, options [nop,nop,T5 val 769758392] ecr 172291702], length 1313: HTTP 10:54:53.35847 fa:16:3e:35:e2:7a > fa:16:3e:49:b8:19, ethertype IPv4 (0x0800), length 1379: 13.0.1.101.49004 > 13.0.1.102.80: Flags [.], ack 1, win 206, options [nop,nop,R5 val 172291866 ecr 769758392], length 0 10:54:53.35847 fa:16:3e:35:e2:7a generic@sfc-fwd-1-sfc-generic-endpoint-vnfd-vm-1: ~ 1

FIGURE B.17 SFC EXPERIMENT IN THE NFV TESTBED, WHEN THE TRAFFIC SOURCE AND DESTINATION ARE THE IOT GW AND BACKEND SERVER AT ENG PREMISES.