5G ZORRO

Grant Agreement 871533

H2020 Call identifier: H2020-ICT-2019-2
Topic: ICT-20-2019-2020 - 5G Long Term Evolution

D4.2: Intermediate prototype of Zero Touch Service Mgmt with Security and Trust

Dissemination Level		
☒	PU	Public
☐	PP	Restricted to other programme participants (including the Commission Services)
☐	RE	Restricted to a group specified by the consortium (including the Commission Services)
☐	CO	Confidential, only for members of the consortium (including the Commission Services)

Grant Agreement no:	Project Acronym:	Project title:
871533	**5GZORRO**	**Zero-touch security and trust for ubiquitous computing and connectivity in 5G networks**

Lead Beneficiary:	Document version:
NXW	**V1.0**

Work package:
WP4 - Zero Touch Automation with Trust, Security and AI

Deliverable title:
D4.2: Intermediate prototype of Zero Touch Service Mgmt with Security and Trust

Start date of the project:	Contractual delivery date:	Actual delivery date:
01/Nov/2019 **(Duration 30 months)**	**30/Apr/2021**	**01/June/2021**

Editor(s)

Pietro G. Giardina (NXW)

List of Contributors

Participant	Short Name	Contributor
Nextworks	NXW	Pietro G. Giardina, Juan Brenes, Michael De Angelis, Elena Bucchianeri, Gino Carrozzo
i2CAT Foundation	i2CAT	Adriana Fernández-Fernández, Carlos Herranz Claveras, Javier Fernandez Hidalgo, Muhammad Shuaib Siddiqui
Universidad de Murcia	UMU	José María Jorquera Valero, Pedro Miguel Sánchez Sánchez, Manuel Gil Pérez, Gregorio Martínez Pérez,
Atos Spain	ATOS	Fernando Díaz Bravo, Guillermo Gómez Chavez
IBM Israel Science and Technology	IBM	David Breitgand, Kathrine Barabash
Altice Labs	ALB	André Gouveia, André Gomes, Gonçalo Machado, Francisco Sério, José Bonnet
Intracom	ICOM	Alberto Erspamer, Alberto Erspamer, Dimitrios Laskaratos, Vasileios Theodorou
Ubiwhere	UW	Francisco Cardoso, Filipa Martins
Fondazione Bruno Kessler	FBK	Rasoul Behravesh
Telefonica Investigacion y Desarrollo	TID	Diego R. López

List of Reviewers

Participant	Short Name	Contributor
IBM Israel Science and Technology	IBM	K. Barabash
Nextworks	NXW	Gino Carrozzo
i2CAT Foundation	i2CAT	Muhammad Shuaib Siddiqui

Change History

Version	Date	Partners	Description/Comments
0.1	15 Apr 2021	ATOS, UMU	Contribution from ATOS, UMU
0.2	21 Apr 2021	UMU	Contribution from UMU
0.3	07 May 2021	NXW, UMU, ATOS, IBM, UW, ICOM, ALB	Contribution from NXW, UMU, ATOS, IBM, UW, ICOM, ALB
0.4	13 May 2021	NXW, FBK, UW, I2CAT, ATOS, ALB	Contribution from NXW, UMU, ATOS, IBM, UW, ICOM, ALB
0.5	18 May 2021	NXW, ATOS, UMU	Contribution from NXW, UMU, ATOS
0.6	25 May 2021	NXW, UMU, UW, ATOS, ALB, ICOM, IBM	Contribution from NXW, UMU, UW, ATOS, ALB, ICOM, IBM
0.7	31 May 2021	IBM, NXW, UMU	Internal QA review from IBM, fixes from NXW and UMU
1.0	31 May 2021	NXW, i2CAT	Final QA review and submission

DISCLAIMER OF WARRANTIES

5GZORRO has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 871533. The content of this deliverable does not reflect the official opinion of the European Union. Responsibility for the information and views expressed in the deliverable lies entirely with the author(s).

Table of Contents

List of Tables

List of Figures

Executive Summary

This document reports on the intermediate prototypes of the 5GZORRO orchestration and security platforms produced by the Workpackage 4.

The prototypes described in this deliverable implement the first working version of the 5GZORRO solutions for advanced management of 5G Service and Slices with high level of security and trust, zero touch automation and intelligent resource selection and optimization. The prototypes are built to be integrated with an enhanced NFV management and orchestration solution (MANO) to provide intra and cross-domain services as well as network slice deployment, strictly coupled with 5G infrastructure management and monitoring functions.

The reported prototypes implement specific parts of the 5GZORRO Platform and specifically realise the initial functionalities of:

- The Security and Trust Orchestration sub-system
- The Intelligent and Automated Slice and Service Management sub-system
- The Enhancements to the MANO and Network Slicing sub-systems.

The Security and Trust Orchestration prototype enables the concept to trustworthiness in two different ways. At first, the Trust Management Framework implements a mechanism to measure the reputation of the stakeholders; then, the Trust Execution Environment Manager provides the secure execution of services into a trusted environment (Trusted Execution Environment – TEE), thus protecting it against malicious stakeholders. Services that can be instantiated in a TEE can be either stakeholder services, i.e., services that can be purchased from the 5GZORRO Marketplace, or directly 5GZORRO Platform Services (i.e., management and control functions). Security management is realized at both intra and cross domain levels and provides services such as secure cross-domain connections and intra-domain detection of failure events and threats.

The automated management of secure services and slices is instead realized with a specific prototype called ISSM (Intelligent Service and Slice Management). It consists of several specialized modules enabling the intelligent slice and service orchestration and optimization. The ISSM links the business services (e.g., Marketplace) provided by the 5GZORRO platform with the underlying MANO-based orchestration mechanisms to execute business orchestration of workflows and specific optimization actions based on several constrains (e.g., cost, efficiency, and trustworthiness). ISSM decisions are enforced by the Network Slice and Service Orchestrator (NSSO) which is the prototype for MANO enhancements provided by the 5GZORRO project. The NSSO manages the lifecycle of 5G communication services, maps them into proper network slices instantiated by the underlying NFVOs (e.g., ETSI OSM). The orchestration process includes interaction with other modules in charge of enhancing the base MANO mechanism: these implement resource and service monitoring functions as well as e-licensing verification mechanisms, both reported into this deliverable as prototypes.

This deliverable describes the current status of implementation for the modules designed in deliverable D4.1 [1], and specifically provides:

- Updates to design and API interfaces as they derived from the respective implementation sprints;
- Reference to the major prototype components and the respective source code repositories hosted in https://github.com/5gzorro.
- Reference to specific functional tests executed on the single components
- Results of a preliminary integration work in generic scenarios for the modules in a s stage of more mature development, along with information on the integration strategy adopted by 5GZORRO to facilitate the development and the integration of the different component of the platform.

1. Introduction

The orchestration of network services and slices still remains an active and central topic in all of the 5G management solutions. In fact, beside any abstractions that can be built on top of the management and orchestration (MANO) platforms, the final goal for any 5G network stakeholder is to have one or more network services deployed on one or more dedicated networks slices automatically and in very short timeframes.

Given this context evaluation which is common to many (if not all) the 5G management and orchestration architectures in state of the art, one of the objectives of 5GZORRO is to enhance the MANO mechanisms across various directions:

- be multi-stakeholder (i.e., multi-operator)
- be capable of increased automation in the establishment of the 5G services and slices (zero-touch) also when spanning multiple administrative domains (cross-domain)
- have security embedded in lifecycle management mechanism, i.e., with secure resource deployment and secure resource usage,
- have a platform capable to provide visibility via measurements of the level of trustworthiness of the other stakeholders.

In this sense, the secure establishment of network services and slices in a multi-stakeholder environment, with a high level of automation can be considered a challenging core objective of the 5GZORRO project.

To meet such requirements and achieve the aforementioned objective, a set of software modules have been designed and are being developed in 5GZORRO project which implement the concepts of security and trust, to be integrate with intelligent and automated management of services and slices.

This document contains information on the intermediate prototypes released at the end of May 2021 which derive from the completion of first development sprints and their initial integration as part of the 5GZORRO platform. The main functions covered by these prototypes and consequently the scope of the deliverable are:

- intermediate release of Trust and Security framework,
- intermediate release of Intelligent Slice and Service Management
- intermediate releases of MANO enhancements.

1.1. Document outline

This document is structured in two main parts.

In the first part, the current status of the modules is reported in terms of design updates, developments of the prototypes, and functional tests. In this sense, the first part of the deliverable has the same structure of deliverable D4.1 and can be considered as a follow-up. In particular, Section 2 focuses on the modules that implement the concepts of security and trust into the 5GZORRO platform. Sections 3 and 4 report the set of components that build the 5GZORRO orchestration stack, implementing the intelligent automation in the management of stakeholder 5G network services and slices and the enhancements of the MANO stack in terms of resource management, orchestration, and licensing.

In the second part, which maps into Section 5, an early integration work is reported against generic scenarios which will be then specialised with use case validation scenarios. The section is organised to cover three different integration scenarios:

- Zero-touch slicing
- e-licensing control
- data-driven actuation.

Section 6 contains some conclusions with reference to the matched project objectives and highlights to the contribution of this deliverable to related KPIs.

Some specific theoretical background at the base of the Trust Management Framework implementation has been reported in Appendix I, as complement to the core prototype description information.

2. Security and Trust Orchestration

The 5GZORRO Security and Trust layer introduces the security and trust functionalities required to ensure the selection of trusted third parties, protect a tenant service or application running in a computing node, and secure the communications between 5GZORRO components. In this vein, Security and Trust Orchestration plays an important role not only for the suitable 5GZORRO platform functioning but also for bringing a high-level trustworthiness. On account of this, the Security and Trust layer may be interpreted as an enabler both at intra-domain and inter-domain environments.

The 5GZORRO Security and Trust Orchestration sub-system is split into four modules:

- Trust Management Framework, which is responsible for determining stakeholders' trust score and establishing reliable relationships;
- Trusted Execution Environment Security Management, which enables critical workloads in multi-tenant and multi-stakeholder scenarios
- Intra-domain Security module, which offers internal security services such as detecting and mitigating feasible attacks and threats
- Inter-domain Security module, which guarantees a secure end-to-end communication through an on-demand VPN as a Service (VPNaaS).

In this deliverable the Intra-domain Security module is not presented as just preliminary tests on the Intra-domain security monitoring enablers have been executed (see Sec. 2.3). The work for integrating these enablers withing the 5GZORRO platform is planned for the next software release.

2.1. Trust Management Framework

5GZORRO ecosystem brings several novelties, for instance, enabling secure, flexible, and automated establishments that boost new compositions of resources and services in 5G networks by means of multi-stakeholder combination. 5GZORRO introduces the Trust Management Framework (TMF) that is the module in charge of guaranteeing the establishment of a trustworthy end-to-end chain across domains. Therefore, in a situation where a stakeholder needs to forecast trust levels of multiple stakeholders in order to identify the most appropriate entity, the Trust Management Framework will provide the required functionality to assess stakeholders' reputation, and in consequence, to ensure a reliable selection based on past experience and recommendations.

The scope of the Trust Management Framework is not only to evaluate stakeholders' trust score but also to manage the whole lifecycle of trust computation process. The main steps of a trust lifecycle management are:

- data collection process from available storage sources (Data Lake and dedicated trust database);
- assessment of trust information from own history and recommendations;
- decision-making based on a trust score
- trust information storage for future interactions;
- finally, the continuous evaluation of trust in an established relationship.

2.1.1. Design Updates

Novel considerations related to the integration of the Trust Management Framework with other 5GZORRO components (workflows) have emerged since the release of the related design in deliverable D4.1 (Jan. 2021). Therefore, a new internal workflow has been designed to better describe the interaction of the Trust Management Framework with other key modules and the triggered actions. As it can be observed in Figure

2-1, the Smart Resource and Service Discovery (SRSD) application module is the one in charge of launching the trust computation process for an initial list of offers pre-classified by the SRSD employing mechanisms such as intent-based priorities, imperative constraints and considerations provided by the consumer. Once the Trust Management Framework receives a set of offers to be evaluated, the framework will start the data collection process for each stakeholder associated with it. Hence, the following steps will be carried as many times as offers need to be assessed.

The Trust Management Framework in each domain should possess a previously generated key pair (intra-domain keys), as well as a Decentralised Identifiers (DID). Each 5GZORRO Platform Participant of a particular domain must use the public key of its Trust Management Framework to share trust data into the Data Lake. Additionally, the Trust Management Framework should possess another global key pair (inter-domain keys) that could be used by other 5GZORRO Platform Participants (from other domains) in order to provide domain recommendations. These recommendations may subsequently be used by other participants who do not have previous information about a specific service or resource. Both key pairs (intra- and inter-domain) should be created by the Identity and Permission Manager before the 5GZORRO Platform Participant interacts with the Trust Management Framework.

The impact of these changes is reflected in the workflow described in Figure 2-1.

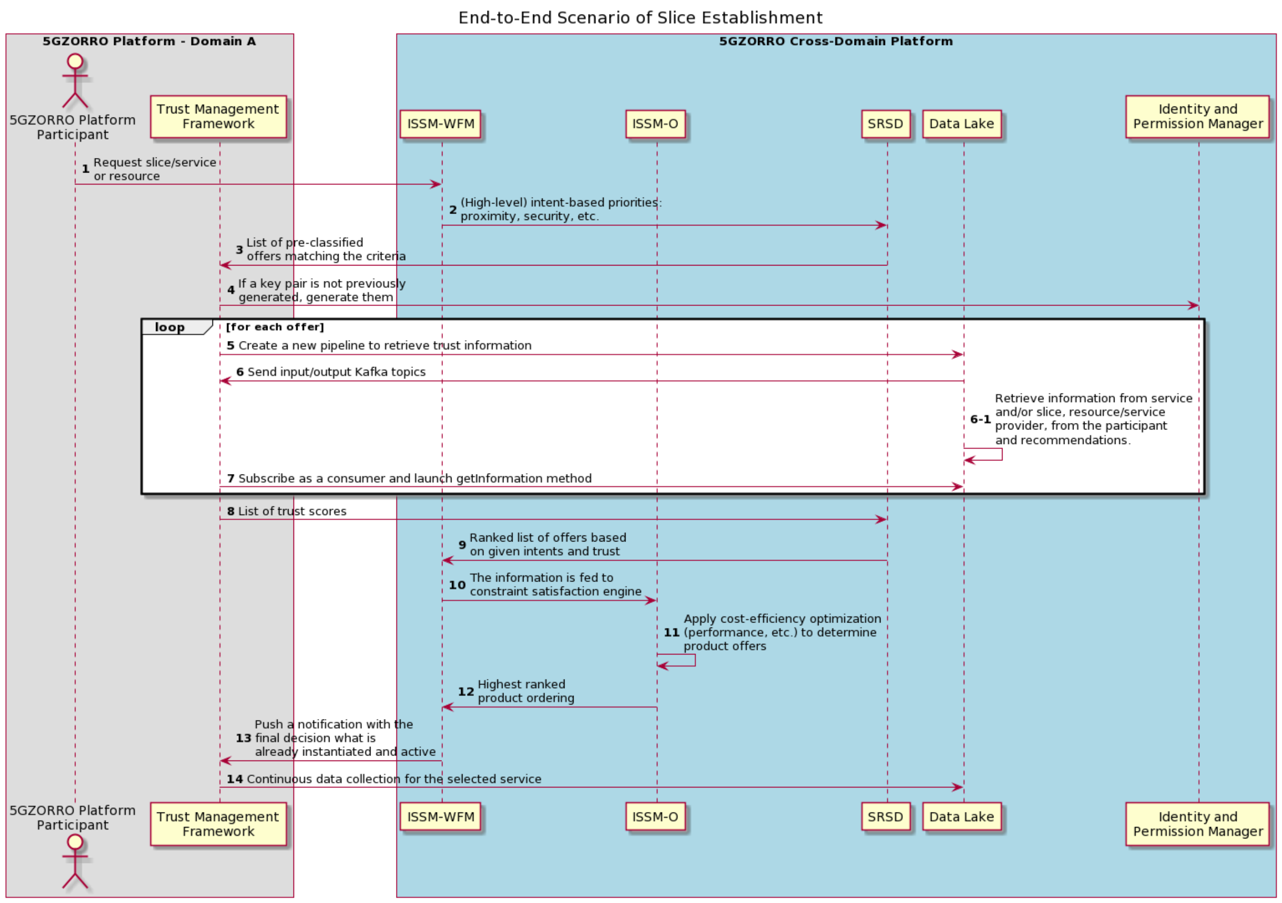

End-to-End Scenario of Slice Establishment
5GZORRO Platform - Domain A
5GZORRO Cross-Domain Platform
5GZORRO Platform Participant
Trust Management Framework
ISSM-WFM
ISSM-O
SRSD
Data Lake
Identity and Permission Manager
1 Request slice/service or resource
2 (High-level) intent-based priorities: proximity, security, etc.
3 List of pre-classified offers matching the criteria
4 If a key pair is not previously generated, generate them
loop [for each offer]
5 Create a new pipeline to retrieve trust information
6 Send input/output Kafka topics
6-1 Retrieve information from service and/or slice, resource/service provider, from the participant and recommendations.
7 Subscribe as a consumer and launch getInformation method
8 List of trust scores
9 Ranked list of offers based on given intents and trust
10 The information is fed to constraint satisfaction engine
11 Apply cost-efficiency optimization (performance, etc.) to determine product offers
12 Highest ranked product ordering
13 Push a notification with the final decision what is already instantiated and active
14 Continuous data collection for the selected service

Figure 2-1: Trust Management Framework integration with SRSD and Data Lake

The Trust Management Framework will launch the *startDataCollection* method (see sec. 2.1 of deliverable 4.1 for details). Such capability will create new Data Lake pipeline that will retrieve trust information for each 5GZORRO Platform Participant using the two templates defined in sec. 5.1 of deliverable 4.1 (ref. step 5 in Figure 2-1). As an outcome of this step, the Data Lake platform will generate a new input and output Kafka topics which will be forwarded to the Trust Management Framework. It should be outlined that Data Lake ought to gather information not only from the specific resource or service but also from its resource or service provider. In the same way, the Data Lake ought to retrieve information from the 5GZORRO Platform Participant who triggered a business flow but also recommendations from other participants. Hence, we may acquire trust information from up to 4 levels: a particular resource or service history, a particular resource or service recommendation(s), 5GZORRO Platform Participant history (an entity with which we want to establish a relationship of trust), and 5GZORRO Platform Participant recommendation(s).

Subsequently, the Trust Management Framework will be subscribed through its *getInformation* method as a consumer of the previous output Kafka topic provided by the Data Lake and will obtain all the information collected in the templates. This information will be utilised to perform a final trust score that will be forwarded to the SRSD. Given that we are assessing the confidence scores of a set of product offers but the final selected offer is not known at this stage, the Trust Management Framework will disable the continuous data collection process of each product offer via the *stopDataCollection* method.

After receiving all trust assessments from the list of offers, the Smart Resource and Service Discovery application will generate a ranked list of offers based on certain intent priorities as well as the trust scores. Then, such a list of offers will be sent to the Intelligent Slice and Service Manager. After that, the ISSM Optimizer (ISSM-O) will determine a cost-efficient allocation of resources and services available at the marketplace to implement the required slice or service. Hence, the ISSM-O should notify the final list of highest ranked product ordering to the ISSM Workflow Manager (ISSM-WFM). At this point, the ISSM-WFM will carry out multiple tasks, with the aid of other modules like the Network Slice and Service Orchestrator (NSSO), in order to configure and instantiate a slice. Once the slice is instantiated and active, the ISSM-WFM will notify the final decision to the Trust Management Framework. As a final step, the Trust Management Framework will launch a continuous data collection for the selected slice or service in order to start the entire trust lifecycle and update the trust score, in the case of some defined events or triggers are activated.

Due to the changed workflow presented above, it has been necessary to adapt and define new interfaces for the Trust Management Framework service:

- the previous *startDataCollection* method, which was responsible for starting and stopping the data collection, has been split into two methods the *startDataCollection* and *stopDataCollection* (see Table 2-1).
- two new interfaces (*requestTrustScores* and *notifyFinalSelection*) have been created in order to enable the interaction of the Smart Resource and Service Discovery application and the ISSM-WFM with the Trust Management Framework.

Details on these interface updates are reported in Table 2-1 below.

Table 2-1: Definition of new Trust Management Framework service interfaces

Operation name: **startDataCollection**		
Description	This method is responsible for starting a continuous collection data (e.g., Monitoring Analytic, Security Management Service, etc.) from Data Lake platform.	
Input Parameters	**Type**	**Description**
stakeholder_did	String	Stakeholder's DID on which it launches the process of continuous information recollection.
trust_parameters	Dictionary	Empty template with the data to be collected in the Data Lake.
*specific_events**	Boolean	Set of triggers that enable an event drive pre-processing to take place directly in the Data Lake.
Output Parameters	**Type**	**Description**
data_collection_status	Boolean	It returns true or false if the data collection cycle is active or not.

Notes
More detailed information about the dictionary can be found in Section 5.1 of Deliverable 4.1 (trust management framework information model). *Specific_events** is an optional parameter.

Operation name: **stopDataCollection**			
Description		This method is responsible for stopping a continuous collection data (e.g., Monitoring Analytic, Security Management Service, etc.) from Data Lake platform.	
Input Parameters		**Type**	**Description**
	stakeholder_did	String	Stakeholder's DID on which it disables the process of continuous information recollection.
Output Parameters		**Type**	**Description**
	data_collection_status	Boolean	It returns true or false if the data collection cycle is active or not.
Notes			

Operation name: **requestTrustScores**			
Description		This method is employed by the Smart Service and Resource Discovery application in order to evaluate trust scores of a list of pre-classified offers.	
Input Parameters		**Type**	**Description**
	list_offers	List of Product offers	Set of offers matching stakeholder's criteria such as intent-based priorities.
Output Parameters		**Type**	**Description**
	list_trust_scores	List of double	It returns the trust score computed for each offer.
Notes			

Operation name: **notifyFinalSelection**			
Description		This method is employed by the ISSM-WFM to notify the Trust Management Framework the final resource or service/slice selected and on which the method will be launched internally.	
Input Parameters		**Type**	**Description**
	id_participant	DID	Distributed identifier of the 5GZORRO participant who request a slice or resource.
	id_candidate	DID	Distributed identifier of the final stakeholder candidate.
	id_offer	DID	Distributed identifier of the final selected resource or service/slice.
Output Parameters		**Type**	**Description**
	result	Boolean	It indicates if the operation has been completed successfully or not.
Notes			

In relation to the aforementioned workflow updates, it has been required to review the previous Trust Management Framework Information Model presented in deliverable D4.1. In particular, the updated information model contemplates new trust parameters in order to assess the trust score of services and resource providers and their product offers.

More specifically, to correctly manage the new trust parameters defined in the Trust Management Framework Information Model, multiple equations have been defined which are used to compute the 5GZORRO stakeholder trust score (see Trust Management Framework Information Model). In particular, the Trust Management Framework leverages the well-known reputation model called *PeerTrust* [1] which is envisaged as a basis from which to design equations in line with the 5GZORRO ecosystem. Thus, these equations will be employed by one of the modules that make up the architecture of the TMF presented in deliverable D4.1, the Trust Assessment module. In this regard, all trust parameters as well as equations contemplated for calculating trust scores can be found in Appendix I.

2.1.2. Prototype implementation

The Trust Management Framework designed in deliverable D4.1 included various modules: Trust Information Sources, Trust Assessment, Trust Results and Evidence Storage, and Trust Level Update.

The current prototype covers in terms of modules and interfaces the Trust Information Source and the Trust Results and Evidence Storage modules. Instead, Trust Assessment and Trust Level Update modules are in an early implementation phase at the time of release of this deliverable (they will be part of the 5GZORRO full release).

Regarding the communication paradigm employed by the Trust Management Framework, it follows both a request/response paradigm and a publish/subscribe paradigm. The former is utilised by the 5GZORRO modules that need to launch the trust lifecycle, for instance, the SRSD. The latter is used by the Trust Management Framework itself in order to retrieve historical trust information, recommendations, and new events or triggers, as well as to send trust information about a stakeholder. In the case of publish/subscribe communication paradigm, such a framework will utilise the common Kafka instance which will be shared with other 5GZORRO modules (intra- and cross-domain communication fabric).

The definition of the Trust Management Framework interfaces is available in the following GitHub page*:* *https://5gzorro.github.io/Trust-management-framework/*.

Concerning the virtualization technology used for the software package, this module is released as a Docker container to facilitate orchestration and delivery together with other 5GZORRO modules.

2.1.3. Functional tests

The following table describes several functional features which allow checking the proper behaviour of different modules and activities carried out by the Trust Management Framework. It should be noted that all the tests described in Table 2-2 are not related to the integration of the Trust Management Framework with other 5GZORRO modules. Therefore, the following tests are internal to the TMF.

Table 2-2: Trust Management Framework functional tests

Name	Description	Passed (Yes/No/Partially)
Trust Parameters	Verify whether the dictionary received as parameter complies with the format of the described trust information model	Yes
Start data collection	Verify that The Trust Management Framework is able to create a new pipeline in Kafka	Yes
Receive information from a topic	Verify that The Trust Management Framework can retrieve trust information from a particular stakeholder to compute a trust score.	Yes
Send information to a topic	Verify that The Trust Management Framework is able to push information to a Kafka topic and also to the Data Storage in order to keep track of a trust relationship.	Yes

2.2. Trusted Execution Environment Security Management

The Trusted Execution Environment Security Management module focuses on the development of functionalities that allow 5GZORRO to protect their tenant service or application running in a computing node against a stakeholder with malicious intentions.

For that purpose, this module will integrate commercial Trusted Execution Environments for the software components execution to enhance the security and trust of the software.

As previously mentioned in the D4.1, SCONE framework modules [8] are integrated with the orchestration services in 5GZORRO to allow the deployment of critical services on SGX-enabled [9] nodes made available in the Marketplace. The orchestration services can then deploy a custom application into a secure enclave, ensuring end-to-end encryption and secure provisioning of the application data, and keys.

2.2.1. Design Updates

No design update has occurred with respect to the deliverable D4.1. In fact, the main focus of the prototype for this intermediate 5GZORRO release has been on testing the SCONE framework capabilities.

2.2.2. Prototype implementation

With a focus on the SCONE framework to enable TEE capabilities, a proof of concept was prepared to validate the integration of TEE with the orchestration services. A Docker container was deployed in a virtual machine with SGX-enabled capabilities in a Kubernetes cluster with two nodes, one with SGX-enabled capabilities from the cloud provider Azure [10] and another without those capabilities placed in an OpenStack instance. In this scenario, an application can run in a shared environment (i.e., OpenStack, Cloud, on-prem) without the need for services to trust in each other; in other words, the application can run in a non-trusted shared environment, since only the application itself (with the hashes generated by a special service called CAS, described below) can see its own security-relevant and runtime information. Every other service or other tenant will see encrypted information at runtime or at rest.

The adopted SCONE framework for TEE is based on four main components that allow deploying an application in a secure environment:

- **Session:** Entails all security-relevant details of a SCONE application. It includes every aspect of the container ecosystem such as commands to be executed on images, secrets, volumes and environment variables. Everything is then attested with the unique signature of the enclave which is authorized to retrieve the secrets from the CAS;
- **LAS (Local Attestation Service):** Service that runs locally, alongside the enclave, and the application that wants to access the secrets. This service is responsible to complete the application attestation and ensure that the hardware is SGX-enabled and share that information with CAS;
- **CAS (Configuration and Attestation Service):** Remote service which stores things like configurations, secrets and filesystem keys needed by the application in runtime, that were provided by the session. This service ensures that all secrets are protected from being visible by humans and are only visible inside the TEEs. These secrets and configurations will then be put in transit and shared with the application once the attestation takes place (when the application proves its integrity and authenticity);
- **Docker container:** The trusted application intended to run in an enclave that once attested by the LAS it will receive the configurations and secrets that are stored in the CAS.

Using these main four components, a stakeholder that wants to run an application in a secure environment can easily achieve that with a few steps:

- Update the application to run on top of SCONE cross-compiled docker images and mount the source code inside the container;
- Launch a new dedicated CAS, or use any existing one;
- Launch LAS alongside the enclave;
- Get the enclave SCONE hash;

- Upload the session to CAS with the enclave SCONE hash and all security details;
- Run the secure application in an enclave.

In the current version of the prototype the aforementioned steps are executed manually, whilst the target for the final release of the 5GZORRO orchestration platform is to integrate these steps as part of the automated zero-touch instantiation process.

For the proof of concept, a python image provided by SCONE and already compiled to run on SGX-enabled hardware was used as the base image for the container. The code mounted on top of the container prints the message sharing an environment variable that was uploaded using the SCONE session specification to the CAS via SCONE CLI. The CAS used for this proof of concept, was provided by the SCONE framework since they provide it publicly for test purposes.

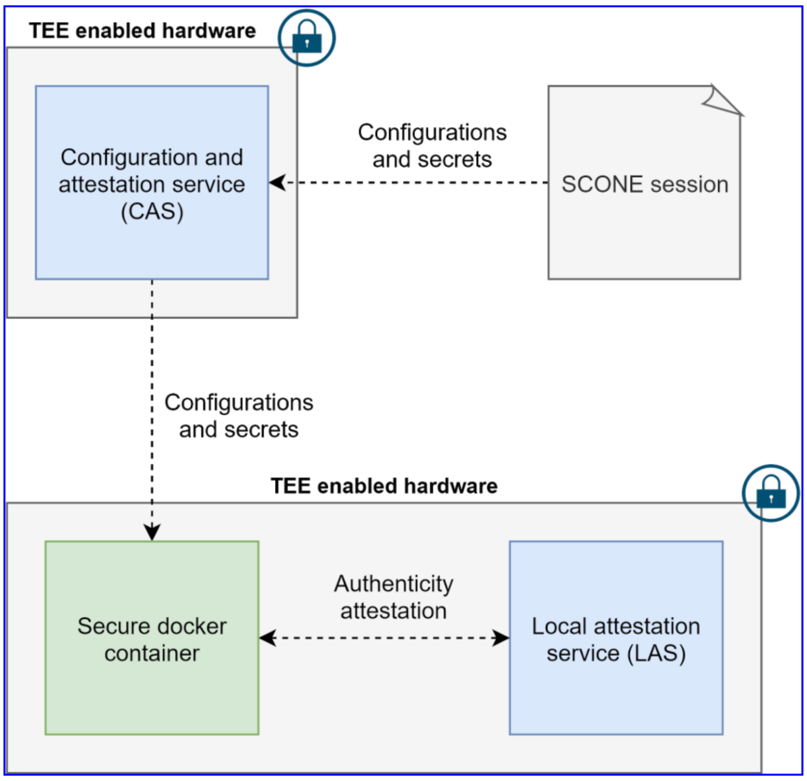

TEE enabled hardware
Configuration and attestation service (CAS)
Configurations and secrets
SCONE session
Configurations and secrets
TEE enabled hardware
Secure docker container
Authenticity attestation
Local attestation service (LAS)

Figure 2-2: TEE Proof of concept diagram

To start the service, a session must be created in the CAS service. Only then, the container with the secure Docker image can be launched in a Kubernetes node. This node is equipped with SGX-enabled capabilities, that are now part of the Kubernetes cluster with two nodes: one node with the required underlying SGX capabilities and one regular node without them. To ensure that the container was instantiated in the correct node (with SGX-enabled capabilities), it was configured taking advantage of the Kubernetes labelling system, which allowed to mark it as SGX-enabled node.

When the docker container starts in the intended node, the LAS that is running alongside the container attests the authenticity of it and the docker container was able to run the configured command (via SCONE session) and prints the intended environment variable. This environment variable cannot be found in the container environment since it is provided by CAS once the attestation takes place.

With this Proof of Concept (PoC), it was possible to achieve the intended validation, which was based on instantiating confidential or critical services in a pool of resources that may be considered untrusted by the requesting party. In fact, the PoC demonstrates how it is possible to run a service in a multi-domain environment whose underlying resources may be serving multiple tenants, potentially with direct access to either said containers or even the host. This validation has been done through a set of specified functional tests, that are presented in the following section in Table 2-3.

2.2.3. Functional tests

The functional tests carried out on this module are mainly related to the correct verification of the implemented proof of concept.

Table 2-3: Trusted execution environment security management functional tests

Name	Description	Passed (Yes/No/Partially)
Create docker image	Create docker image on top of SCONE cross-compiler docker images	Yes
Setup trusted node	Setup a node with SGX-enabled capabilities	Yes
Instantiate LAS	Setup LAS service alongside the enclave	Yes
Upload SCONE session	Upload SCONE session to CAS with the secrets that intended to run on the docker container	Yes
Instantiate docker container	Instantiate docker container in the trusted node	Yes
Retrieve secrets	Retrieve secrets from the docker container that are passed by CAS once LAS attests its authenticity	Yes
Validate docker container security	Check if the secrets that should be passed by CAS were not in the container environment and the running code was encrypted	Yes

2.3. Intra-domain Security enablers

The Intra-domain Security module receives network traffic data as input and produces network events and alerts about potential failures or security threats using three detection mechanisms as stated in deliverable D4.1: Rule-based intrusion detection, Protocol specification and Behaviour analysis.

2.3.1. Design Updates

Currently there are no design updates as the software modules described in the previous version of the deliverable seem to work well. Well-defined and standardized software platforms are currently deployed and in use.

2.3.2. Prototype implementation

In the current implementation, the Zeek platform [11] is used as a network security monitoring tool. This platform monitors network events using various virtual Terminal Access Points (vTAPs) and produce live (real-time) traffic data which are then aggregated and analysed within Elasticsearch platform [12]. Finally, data is sent to Kibana platform [13] for visualization. Stored network data analysis is supported as well.

Elasticsearch is a distributed, RESTful search and analytics engine capable of storing data and searching it in near real time. Kibana is a browser-based analytics and search dashboard for Elasticsearch. All the software modules are deployed as Docker containers in a well-defined Docker network configuration.

Currently, to deliver the JSON text based Zeek logs to Elasticsearch, Filebeat [14] is used. It reads the Zeek log files and delivers them to Elasticsearch. When providing data to Elasticsearch, a pipeline is specified for all events that are inserted inside the Elasticsearch index. After that, all the Zeek logs can be accessed through Kibana dashboard.

The intra domain security module is able to identify virtual attacks which are implemented to test our workflow. Current open issue is the lack of actual real data which will be available after our exposure to an initial setup of the integration flow. Currently the actual implementation follows blacktop/docker-zeek project as published in https://github.com/blacktop/docker-zeek. Current GitHub repository can be found at: https://github.com/5GZORRO/intrasecurity .

2.3.3. Functional tests

Our software tests refer to the successful deployment of the various containers needed for the module and the successful data sharing communication between them.

Table 2-4: Intra-domain functional tests

Name	Description	Passed (Yes/No/Partially)
Zeek service	Successfully deployed. Links with Filebeat (to be considered).	YES
Kibana service	Successfully deployed. Links with Elasticsearch.	YES
Elastisearch service	Successfully deployed.	YES
Filebeat service	Successfully deployed. Links with Elasticsearch and Kibana.	YES

2.4. Inter-domain Security at the communication level

The Inter-domain Security module is in charge of managing the establishment of secure and reliable connections between different domains within 5GZORRO. Thus, this module offers VPN-as-a-service, so that the interconnection between domains is done automatically without any previous configuration. For this purpose, it integrates the key derivation necessary to establish the secure connection with the Identity & Permissions Management module, obtaining the cryptographic keys and information from it.

2.4.1. Design Updates

From Deliverable D4.1, the main design change is related to the deployment in the module. Previously, each client of a domain had to be configured as a client of the domain to which the secure connection was to be generated. Now, the inter-domain security module is designed to be deployed in the network gateways of each domain, so that they act as an intermediary between the end entities and the external domain. This reduces the complexity of configuring each network entity acting as a VPN client to the external domain. It also reduces the number of credentials required for the authentication and secure connection generation process.

According to this design change, it has been necessary to add new interfaces (see Table 2-5) that are required to generate the connection between gateways. In this way, the connection can be generated directly by calling functions of the gateway acting as a client. In this sense, these interfaces should have strong authentication requirements, allowing only to trigger its functionality to trusted entities from external domains.

These new interfaces are:

Table 2-5: Definition of new Inter-domain Security Establishment service interfaces

Operation name: **add_client**			
Description		This method adds a new client to the gateway acting as VPN server. The client is identified using its public key.	
Input Parameters		**Type**	**Description**
	client_public_key	String	Public key of the client to be added, in Curve25519 format.

Output Parameters		Type	Description
	assigned_ip	String	Assigned IP to the new client.
	vpn_port	Integer	Port where the vpn server is running.
	server_public_key	String	Public key of the vpn server.
Notes			

Operation name: **remove_client**			
Description		This method adds a new client to the gateway acting as VPN server. The client is identified using its public key.	
Input Parameters		**Type**	**Description**
	ip_address_server	String	Public key of the client to be removed, in Curve25519 format.
Output Parameters		**Type**	**Description**
	result	Integer	It indicates if the client public key has been removed successfully or not.
Notes			

2.4.2. Prototype implementation

In the current prototype, an initial version of the entire set of interfaces has been implemented to verify its suitability and to test the desired functionalities. However, this implementation only covers the functional requirements to successfully establish a secure connection between domains. The functionalities which will be implemented for the next release are: a) Client-to-gateway and gateway-to-gateway authentication; b) Integration with the Identity and Permission manager module for key management.

The prototype is implemented following a request/response setup based on a Restful API. The definition of the interfaces is available as Swagger file at https://5gzorro.github.io/inter-secure-channel-setup/#/

The prototype has been implemented using Python 3, with the following additional packet requirements:

- *flask* and *flask_restful*, for the Restul API setup [15]
- *Gevent*, for networking management [16]
- *Werkzeug*, for the WSGI server [17]

For the VPN service setup and management, WireGuard VPN [3] has been leveraged. WireGuard is an open-source software implementing VPNs with state-of-the-art cryptography and an easy setup. This tool aims to provide faster connections than previous solutions such as IPSec [4] or OpenVPN [5]. WireGuard works in a client-server setup in which VPN connections are added using new network interfaces. This configuration enables to have different VPN connections to different domains in the same client. The only configuration needed is to define which IP range is redirected to each WireGuard interface.

To enable the automated installation and configuration of WireGuard, the module includes an interface named "launch" which is in charge of deploying WireGuard in the target machine and of configuring the required network properties to enable traffic forwarding. Additional packets installed during setup are linux-kernel-headers and openresolv.

The GitHub repository of the module is available at: https://github.com/5GZORRO/inter-secure-channel-setup.

The current prototype is released in the form of a Virtual Machine (VM) as the module is intended to be deployed in gateways (which are rarely deployed as containers). However, it might be possible to deploy the module in containers if the required dependencies are fulfilled.

2.4.3. Functional tests

The functional tests (see Table 2-6) carried out on this module are mainly related to the verification of the correct functioning of the implemented methods.

Table 2-6: Inter-domain security functional tests

Name	Description	Passed (Yes/No/Partially)
Module setup	The module can be automatically installed and configured.	Yes
Client addition	A new client is connected to the VPN, obtaining a client IP in the VPN.	Yes
Client removal	A client is disconnected, removing its IP from the known list.	Yes
Connection establishment	The automated connection establishment between two VMs running in different subnets is tested.	Yes
Connection deletion	In the current design we use the public key to search for the client to be removed.	Yes

3. Intelligent and Automated Slice & Service Management

The 5GZORRO Intelligent and Automated Slice and Service Management (ISSM) functional block focuses on automated management of secured cross-domain slices and services within them. ISSM is thus responsible for enforcing business transactions both at the system level by interacting with NSSO and with alternative slicing technologies that might be developed in the future, as well as by managing business transaction contexts across the entire 5GZORRO platform allowing a principled, repeatable, auditable, and trustworthy interaction among the multiple components of the platform to realize a specific business flow.

3.1. ISSM Workflow Manager (ISSM-WFM)

The ISSM Workflow Manager (ISSM-WFM) executes orchestration workflows in a context of a business transaction, such as extending a slice across a second domain in cooperation with the Network Slice and Service Orchestration.

3.1.1. Design Updates

There are no significant changes in the ISSM-WFM design since deliverable D4.1. In deliverable D2.3 [18], updated orchestration workflows have been presented which better capture the intended behaviours in scenarios such as cross-domain slice establishment, scale-out and optimization. Moreover, always in D2.3 updates have been provided about how ISSM-WFM integrates with the rest of the functional 5GZORRO architecture.

3.1.2. Prototype implementation

The current ISSM-WFM prototype executes Workflow 3-6, Trustworthy Slice Setup with Third Party Resources, reported in D2.3. The prototype implementation can be found in https://github.com/5GZORRO/issm/. As such it can be considered as fully implemented.

The definition of the business flow implementing Flow 3-6 of D2.3 can be found in https://github.com/5GZORRO/issm/blob/master/flows/issm-sensor.yaml.

Additional business flows will be realized as the rest of 5GZORRO platform matures and will be reported in deliverable D4.3.

3.1.3. Functional tests

The functional tests to validate ISSM-WFM API with respect to the Workflow 3-6 reported in D2.3 are presented in Table 3-1. The tests have been performed using mock-up APIs for all involved components. The ISSM-WFM APIs have also been tested for the workflows involving actual the actual components, such as NSSO, and Data Lake, as described in Subsection 5.2.

Table 3-1: ISSM-WFM functional tests

Name	Description	Passed (Yes/No/Partially)
Create flow	Create a new business flow	Yes
Execute flow	Execute a new business flow	Yes
Get progress	Get progress of a business flow using web hook	Yes
Delete flow	Remove a business flow from ISSM-WFM	Yes
Continuous progress	Observe a progress of a flow with GUI	Yes

3.2. ISSM Optimizer (ISSM-O)

The ISSM Optimizer (ISSM-O) optimizes cost-efficiency and cost-trustworthiness trade-offs of network services and slices required to be created in a context of a specific business transaction subject to constraints such as security and service area. ISSM-O continuously optimizes services and slices that have been already set up in previous transaction flow executions.

3.2.1. Design Updates

Main design updates originated on top of the design baseline provided in deliverable D4.1 are as follows:

- The ISSM-O scope of operation has been split into two parts: (a) intra-domain and (b) inter-domain. The reason for this functional split is that ISSM-O does not have a centralized visibility into the MNOs' NFVIs and inventories. Therefore, the inter-domain component performs optimization at the Marketplace level determining candidate domains for resource instantiation while the intra-domain component optimizes resource allocation within MNO domains. The intra-domain component of ISSM-O can be regarded as optional in the sense that each domain can use its own optimization tools to allocate resources within a domain. However, this task is far from trivial. Building on our previous work, we proposed an intra-domain approach as follows.
- The ISSM-O intra-domain component is responsible for optimizing MNO's resource allocation. Unlike the ISSM-O inter-domain component that does not have any information about the resources, topology, and infrastructure inside the domains, ISSM-O intra-domain performs resource optimization with complete knowledge about the whole resources that are under the control of that specific MNO. The optimization in the domain mainly happens intending to minimize resource utilization and provisioning cost.

The intra-domain ISSM-O is designed considering different types of computing, transport, and radio resources available to be allocated to the requests. Three objective functions have been identified to optimize various aspects of the network based on the MNO's desired goal:

- The first objective function aims to minimize resource utilization for the MNO by dynamically allocating resources to the requests upon the need. Employing this approach enables the MNO to re-schedule the resource allocation in order to avoid resource under/over utilization.
- The second objective function tries to avoid excessive usage of the transport links, and instead, it prefers to consolidate the workload of requests in close proximity of each other.
- Finally, the third objective function performs a cost optimization trade-off inside the domain. Inside each domain, resources can have different usage costs for the MNO; therefore, achieving a cost-optimized resource allocation inside the domain can lead to many economic advantages.

3.2.2. Prototype implementation

The ISSM-O prototype has been implemented in simulation for this deliverable and its integration within the ISSM architecture for the 5GZORRO platform is planned for the next 5GZORRO release.

Optimization solvers such as OptaPlanner (https://www.optaplanner.org/) and Gurobi (https://www.gurobi.com/) are being evaluated for integration as part of ISSM-O implementation. Full description of the implementation of the ISSM-O module, its functional validation, and its integration with the rest of ISSM for inter-domain optimization will be reported in D4.3.

4. MANO and Slicing Enhancements

This section describes the component of the 5GZORRO Platform devoted to the Management and orchestration of services and slices. From an architectural point of view, the set of services offered by the orchestration modules is mainly oriented to the internal consumption by other components in the platform, rather than by the 5GZORRO stakeholders themselves. In this sense, such set of modules offers functionalities related to resource management and monitoring, e-licensing management as well as to control and network slice and service orchestration across multiple and different administrative domains. These functionalities are available towards the upper and more abstracted layers of the 5GZORRO platform, such as the Intelligent Slice and Service Management, that exploits them for building service and slices based on an intelligent composition of the available resources.

4.1. Virtual Resource Manager

The Virtual Resource Manager (VRM) is in charge of the management and monitoring of the resources related to the 5G Virtualisation Platform available in each stakeholder domain adopting the 5GZORRO Platform. The management of the resources is implemented through a set of functionalities that enables the owner of the resources to store and catalogue them in terms of templates, descriptors and packages and includes the onboarding of the resources into the internal catalogue of the target VIM/orchestrator, when needed. Another feature related to the resource management is the direct support to the 5G Offer Catalogue. In this case, the resource manager implements an internal translation logic, invocable on demand, that translates the descriptor of the resources stored into the proper service and resources models defined by the TM Forum (see Section 4.1.2.1). The possibility to implement a service that enables the VRM to be used as VNF/CNF remote repository is still under discussion. For what concerns the monitoring, the internal module called MDA (Monitoring Data Aggregator) retrieves resource statistics from the underlying virtualization platform (e.g., NFVO, Radio Controller, etc.), aggregates them on the basis of a certain configurable rules and, finally, sends them to the platform Data Lake via Kafka Bus.

4.1.1. Design Updates

The VRM is designed to be a multi-container application, with the aim to provide the high level of flexibility required to immediately tackle any changes/updates in both resource composition and in the underlying 5G virtualised infrastructure. Due to this and in part to the fact that some features are still under definition or design, the final composition of such set of containers may vary during the following design and implementation phases, up to the final version of the prototype.

The base services provided by the VRM have been reported in deliverable D4.1. In this section we describe the updates to such set of features and also the initial software design of the components that were not ready to be reported in D4.1.

Figure 4-1 shows to two main elements building the VRM and their relationship with the other components of the 5GZORRO platform. The Resource Manager contains the logic to manage the resource DB and interact with the Resource and Service Offer Catalogue (RSOC), which is the target of the resource definition translation process. The Resource Manager also offers an interface the MDA exploits for retrieving information used to reach the different components of the 5G Virtualisation platform, namely NFVO, VIM, Radio and Network Controller. Same interface is exploited by the e-Licensing modules to retrieve the coordinates to access the NFVO.

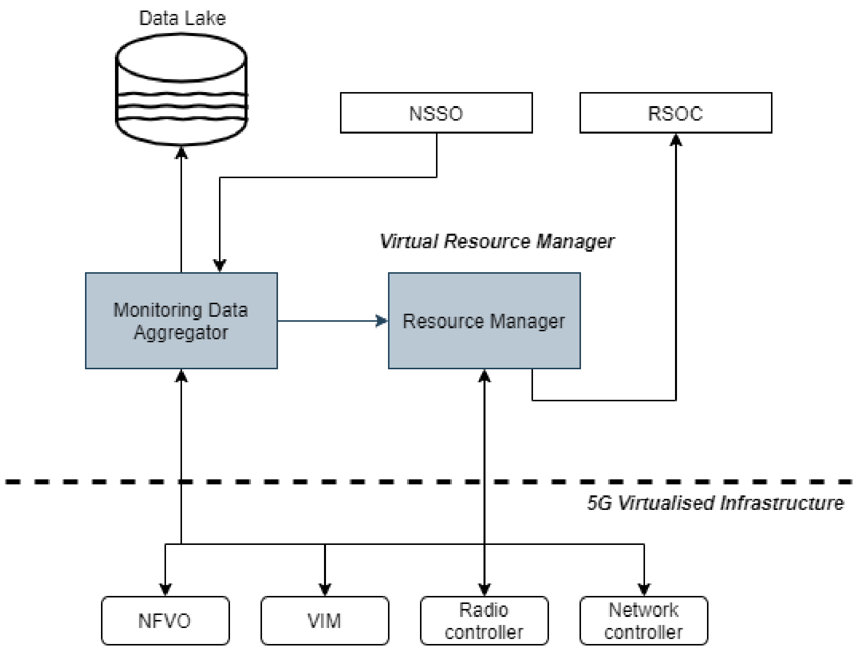


Figure 4-1: Main VRM modules and interaction with other components of the 5GZORRO platform

Focusing into the Resource Manager module, it consists of several modules in charge to implement the functionality of resource storage (descriptors and, where required, packages), model translation, and 5G virtualisation infrastructure DB, as depicted in Figure 4-2.

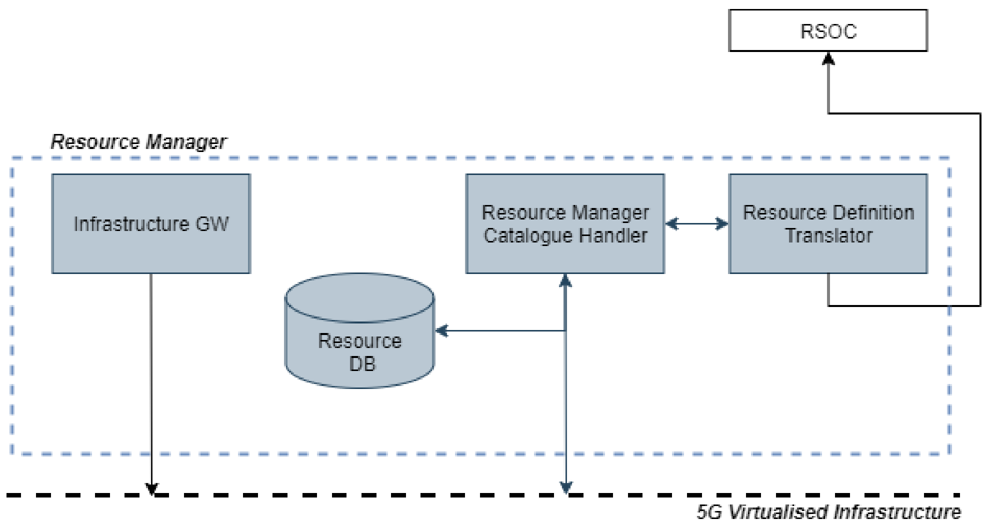

RSOC
Resource Manager
Infrastructure GW
Resource Manager Catalogue Handler
Resource Definition Translator
Resource DB
5G Virtualised Infrastructure

Figure 4-2: Detailed view of the modules composing the Resource Manager

The *Infrastructure GW* provides the interface and the logic to access the underlying 5G Virtualised infrastructure, avoiding the need for components like MDA and e-Licensing modules to directly access the provider internal infrastructure. The *Resource DB* contains the resource definitions expressed in terms of descriptors and, when required, in terms of references to certain resource packages (e.g., the DB contains a VNF descriptor and a reference to the corresponding VNF package). The resource DB is managed by the *Resource Manager Catalogue Handler (RMCH)* that offers the interface and the logic to access to the resource information. The connection between the RM Catalogue Handler and the 5G Virtualised Infrastructure represents the capability of the Resource Manager to access the internal catalogues of the different elements to add (onboard), update, and remove resources. Finally, the *Resource Definition Translator* provides the logic to translate the resources, as defined in the Resource DB, to the proper models defined by the TM Forum as well as the required interface to store such translations to the RSOC.

As depicted in Figure 4-3, the MDA intends to receive from NSSO a monitoring spec with the needed information/identifiers and therefore starting the monitoring stage within the timeline defined. MDA collects metric values from the virtualization platform (for the current time, it collects from NFVO OSM), aggregates them (or not, depending on the measurement type), and forwards the encrypted hash of the monitored data to a Kafka topic.

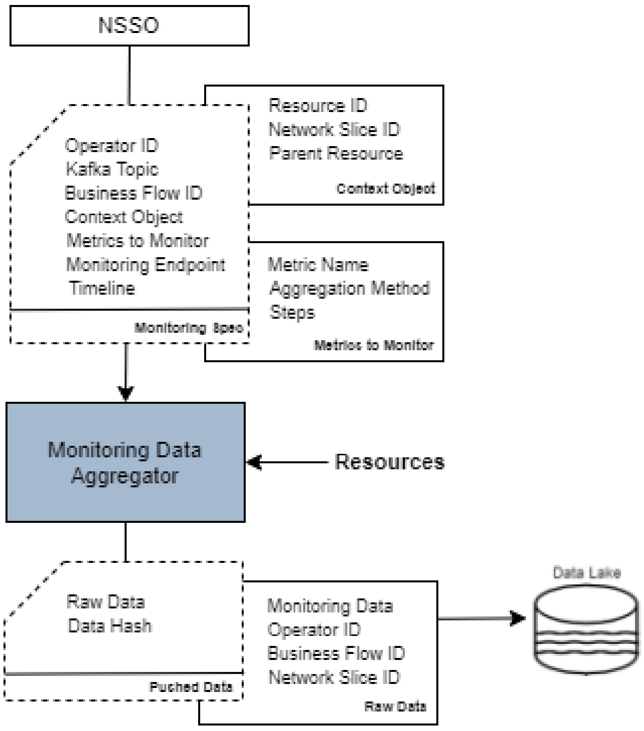


Figure 4-3: Detailed view of MDA functionality Prototype implementation

4.1.2. Prototype implementation

This section reports the current status of the implementation of the Resource Manager and the Monitoring Data Aggregator, which are the two main modules building the VRM. As anticipated, the base idea is to build a multi-container application, therefore, Docker has been selected as the target virtualization platform for the deployment of the VRM.

4.1.2.1. Resource Manager

With respect to design discussed above, the current implementation of the Resource Manager covers partially the Resource Manager Catalogue Handler and the Resource Definition Translator (RDT), whose internal architecture is detailed in Figure 4-4.

The component provides a North-bound interface (NBI) to expose translation services to the administrators who require them and a South-bound interfaces (SBI) to interact with the RSOC and the Resource Manager Catalogue Handler. Both the interfaces are REST-based. An NBI Controller implements the logic to manage the requests from the NBI, whose set of REST APIs is specified in OpenAPI format. When a translation request comes from the NBI, the NBI Controller invokes the Resource Manager Handler to retrieve the target resource definition through the RMCH, then translates it by exploiting the translation rules implemented internally. Finally, the NBI Controller requests the storage of the translated resource to the RSOC through the RSOC Handler.

As can be noted from the figure, the interaction with the RMCH is represented with a dashed arrow, since this part has not been yet implemented, so the translation tests have been performed by pass the description of the resource as a payload of a dedicated API in the NBI. On the contrary, the interaction with the RSOC has been implemented and tested.

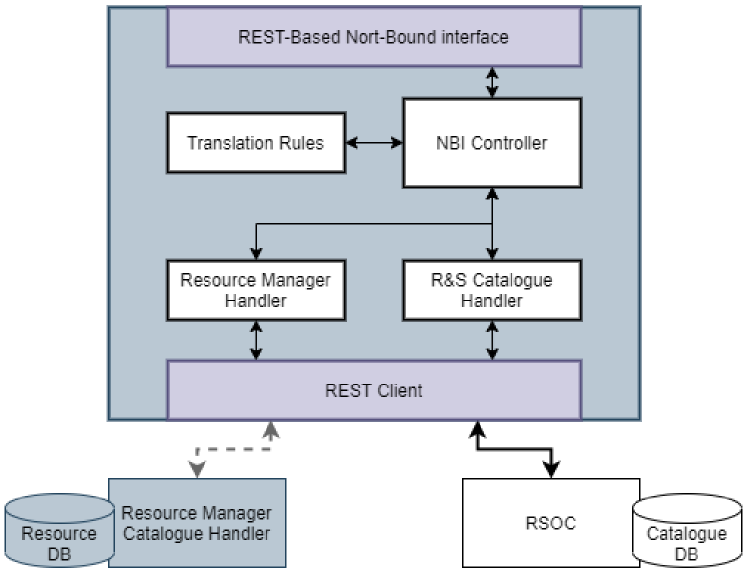


Figure 4-4: Resource Definition Translator internal architecture

The Translation Rules currently implemented provide the necessary logic to translate VNFD, PNFD and NSD defined according to ETSI SOL-006 [19] standard to the correspondent TM Forum models for resources (TMF API 634 [20]) and services (TMF API 633 [21]). In particular, the NSDs are translated as TM Forum services while the VNF and PNF descriptors as TM Forum resources. The RDT is implemented in JAVA 8 and based on the Spring Boot framework [22]. The source code is available on 5GZORRO GitHub project space here: https://github.com/5GZORRO/resource-definition-translator.

The RMCH is the adaptation of the open source 5G Catalogue by Nextworks [23] for the purposes of 5GZORRO VRM. In particular, the implementation of the support to ETSI SOL-006 is in progress, as the current version of the 5G Catalogue implements a descriptor data model based on ETSI SOL-001 [24].

As for the RDT, the resource management logic is programmed in Java8 and based on Spring Boot, while the Resource DB consists of an instance of PostgreSQL [25] and of a set of directories in the local filesystem to store the VNF packages and the related metadata. The logic that manages the Resource DB already implements the onboarding of the descriptors on OSM, while the onboard of the related VM images on the VIM is under development. For these two interactions, the 5G Catalogue uses two specific Java packages:

- *J-OSMClient* [26] – to interact with OSM
- *OpenStack4j* [27] – to interact with the VIM (OpenStack)

4.1.2.2. Monitoring Data Aggregator

Regarding the main workflow of the Monitoring Data Aggregator (MDA), it can be subdivided into four fundamental courses of action:

1. receiving the monitoring spec from NSSO, containing critical information about which metrics will be monitored and from them which will be aggregated following a specific method (for instance, average, sum, or standard deviation), the associated business ID flow, the network slice ID, among others;
2. collecting values of the metrics from resources (at this stage, this module collects from OSM);
3. performing metric aggregations within an aggregation step;
4. pushing encrypted hashed data into a Kafka topic after signing the collected metrics.

Concerning the communication paradigm, exercised by the MDA prototype, it supports simultaneously a request/response and a publish/subscribe setup. Regarding the first one, it is based on FastAPI [28], a Python framework that enables the use of a REST interface, and it will be exploited and employed by modules such as Vertical Slicer to create/enable monitoring specs with the metrics that will be monitored. It is worth mentioning that the endpoints prepared for the request/response paradigm are described on the following readme page: https://github.com/5GZORRO/mda/blob/main/README.md.

Furthermore, the API description is available as an OpenAPI JSON file at https://github.com/5GZORRO/mda/blob/main/doc/openapi.json. The latter paradigm is used by MDA itself, publishing monitoring data to a Kafka topic, received on the monitoring specs.

This component has been implemented using Python v3, leveraging the following tools and libraries:

- *FastAPI* [28], for the REST interface;
- *Kafka-python* [29], for the Kafka producer;
- *Psycopg2* [30], the PostgreSQL driver for Python;
- *RSA* [31], for encrypt operations;
- *Timeloop* [32], to run periodic tasks after a certain interval;
- *Sqlalchemy* [33], SQL toolkit, and Object Relational Mapper.

The GitHub repository of the MDA module is: https://github.com/5GZORRO/mda. For the current version, the monitoring specification including the monitoring endpoint has been updated, where MDA fetches the metric values, and a context object containing the slice ID, the resource ID, and the parent resourceID of this last one. For more information, a wiki page linked to the GitHub repository describes these fields: https://github.com/5GZORRO/mda/wiki/Monitoring-Spec-Example.

The module is deployed in a Kubernetes cluster as the target virtualization technology, facilitating a more reliable integration with the partners and their 5GZORRO modules.

4.1.3. Functional Tests

This section reports the set of main functional tests performed during the implementation of the VRM Resource Manager modules. In particular, Table 4-1 and Table 4-2 describe the test set for the RDT module (as part of the Resource Manager) and the MDA, respectively.

Table 4-1: Resource Definition Translator functional tests

Name	Description	Passed (Yes/No/Partially)
Descriptors Translation	Translation of descriptors for VNF, PNF and NS from ETSI SOL-006 to TMF 633 and 634	Yes
RSOC invocation	Invocation of the RSOC to store the translated services/resources	Yes

Table 4-2: Monitoring Data Aggregator functional tests

Name	Description	Passed (Yes/No/Partially)
Receiving a monitoring spec	The module receives, from Vertical Slicer, a configuration containing the metrics to monitor, the duration of the monitoring stage, and the unique identifiers associated with the operator and business flow ID.	Yes
Collecting values of the monitored metrics	The module collects metric values, from OSM, within the step received from the monitoring spec.	Yes
Pushing data into a Kafka topic	The module, after signing the collected data, that is, performing the hashing and encryption operations, pushes it into a Kafka topic.	Yes

4.2. Network Slice and Service Orchestrator

The Network Slice and Service Orchestrator (NSSO) is responsible for processing communication service level lifecycle management actions, namely instantiate, terminate and scale received from the Intelligent and Automated Slice & Service Management (ISSM). With this scope, the NSSO performs the mapping of the communication service into network slices, across multiple domains, and interacts with the 5G virtualized infrastructure for the lifecycle management of the resources associated with the network slices. This mapping is performed based on rules and algorithms which can be updated dynamically as part of the management control loops. Moreover, the NSSO is responsible for configuring the monitoring metrics associated to the service, and the correspondent e-Licenses to be attached to the service instances.

4.2.1. Design Updates

The design and implementation of the NSSO has been evolved with a more mature definition and implementation of the interfaces between the different components, illustrated in Figure 4-5. In particular, the interface exposed to the ISSM has been defined based on the initial interface available from the software components and updated to allow passing IDs related to the 5G Offer transaction, which are passed to other modules to map resource allocations, and service instances to the smart contracts associated with the offer. The initial interfaces towards the Monitoring Data Aggregation (MDA) and the e-Licensing Manager modules have been established. Work is still ongoing to define the interface between the NSSO and the Network Service Mesh Manager (NSMM), to provision the connectivity between the different components of the end-to-end service, and between NSSOs in other domains to provision slices on top of 3rd party resources.

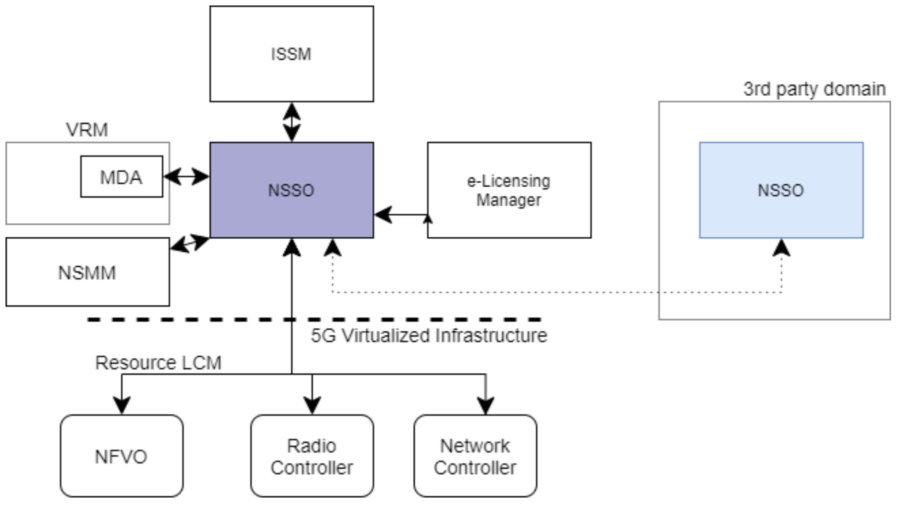


Figure 4-5: Network Slice and Service Orchestrator interfaces

The updated interfaces used for this release of the 5GZORRO platform are shown in Figure 4-6 and Figure 4-7. In particular, Figure 4-6 shows the updated interface between the ISSM and the NSSO, in which two parameters were added to the service instantiation request to allow to receive the ID of the product offer in the 5GZORRO Marketplace, and the Transaction Id representing the Id of the offer acquisition.

The interface between the NSSO and the e-Licensing Manager was updated to allow passing the Product Id from the 5GZORRO Marketplace, as shown in Figure 4-7. This interface is used by the NSSO, during the service instantiation phase, to register the e-Licensing allocation with the Network Service Instance (NSI) ID. This way the e-License Manager can confirm the entitlement of the tenant to deploy the VNFs used by the NSI. We refer to Sec. 4.3 for further details regarding the e-Licensing Manager.

The NSSO interface toward the MDA, shown earlier in Figure 4-3 , was updated also to allow passing the Product Id and Transaction Id, and with the definition of the Metric information model. The updated interface enables the NSSO to configure the metrics to be collected and attach the metrics to the Ids established in the 5GZORRO Marketplace to enable improved monitoring and analysis mechanisms in the Data Lake.

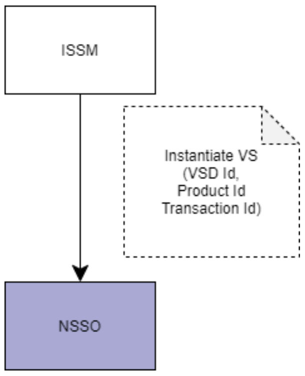


Figure 4-6: Updated ISSM-NSSO Interface

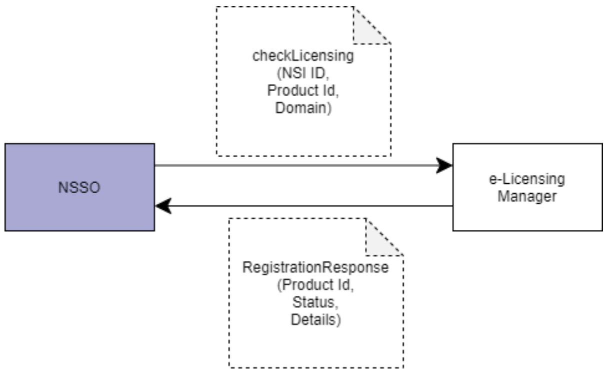

checkLicensing (NSI ID, Product Id, Domain)
NSSO
e-Licensing Manager
RegistrationResponse (Product Id, Status, Details)

Figure 4-7: Updated NSSO-e-Licensing Manager Interface

The NSSO groups several components providing different functionality, as depicted in Figure 4-8.

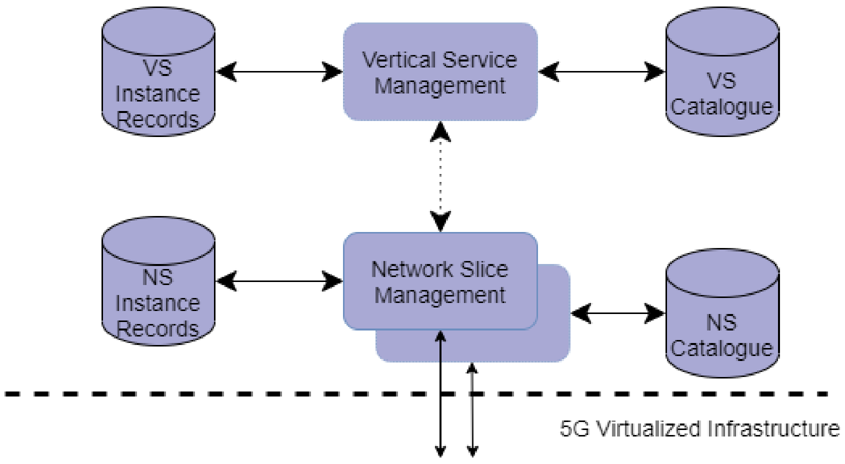


Figure 4-8: Detailed view of the modules composing the Network Slice and Service Orchestrator

The Vertical Service Management is responsible for processing the service lifecycle management requests, while the Network Slice Management modules contain the logic to translate network slices into resource and network service allocations in the 5G Virtualized infrastructure. The multiplicity of the Network Slice Management aims to illustrate that there may be different software modules implementing the logic to allocate the network slices on different segments of the network. In particular, two different software modules have been identified for the initial prototype:

- *Generic-purpose NSMF*: offering standardized network slice management interfaces, and implementing a driver-based approach to interact with the underlying infrastructure and MANO platforms.
- *Slice manager (SM)* evolved from 5GCity project [34]

Each of these modules is accompanied by two different databases, containing the correspondent descriptors, and the information regarding the instances.

As stated in D4.1, the 5GCity slice manager is being extended in the context of 5GZORRO to meet the requirements of a NEST-based NS provisioning. To do so, the following functionalities have been added in the current prototype:

- *Management of Generic Slice Template (GST):* Define the slice attributes together with the set of supported values for each of them in the managed domain. The specific attributes that we are currently using are:
    - Isolation Level (e.g., Logical Isolation)
    - User Data Access (e.g., Direct internet access)
    - Compute Availability Zone (e.g., nexusEdge)
    - Access Technology (e.g., WiFi, Cellular)
    - Area of Service
    - Radio Spectrum (WiFi channel, 4G/5G band, duplex mode)
    - Guaranteed Downlink Throughput per UE
    - Maximum Downlink Throughput per UE
    - Guaranteed Uplink Throughput per UE
    - Maximum Uplink Throughput per UE
- *Management of Network Slice Type (NEST):* Define an instance of the GST with required values for the desired attributes. Note that in order to be feasible, selected values must be a subset of supported values. Together with the registration of every NEST, a network slice blueprint [35] (i.e. required physical and logical resources) is also generated in the SM, which references the resources to be used by this NEST. In the current prototype, this can be done in two different ways:
    - Imperative approach: The resources to be used by the slice are also passed in the payload provided in the NEST creation request.
    - Declarative approach: The internal logic of the SM (as NSMF) and the RAN Controller (as NSSMF) is used to determine the resources to be included in the slice blueprint, based on the requested attributes' values.
- *NEST-based Network Slice Instance (NSI):* Define a network slice instance following the specifications of the blueprint associated with the selected NEST. Additionally, some configurable parameters are also expected at this step for the required slice/service (e.g., PLMN, SSID, etc.)

4.2.2. Prototype implementation

The prototype covers all the modules of the NSSO and is built upon the following software components:

- *Vertical Service Manager*: Containerized deployment of the vertical service management function available in [36]. This module contains the driver to interact with network slice manager from [36] .
- *Generic-purpose NSMF*: Containerized deployment of the network slice management function available in [36]. This module contains the drivers to interact with the MDA and with the e-Licensing Manager, using the APIs established in [40][41].
- *I2CAT Network Slice Manager*: Containerized deployment of the SM. This module contains the driver to interact with the i2CAT's RAN Controller.

4.2.3. Functional tests

The set of functional tests conducted so far on the NSSO are reported in Table 4-3.

Table 4-3: Network Slice and Service Orchestrator functional tests

Name	Description	Passed (Yes/No/Partially)
Service instantiation	A simple service instantiation is requested from the ISSM. The NSSO provisions a network slice with one network service relying on the local domain functionality.	Yes
Configuration of monitoring metrics	The NSSO requests the configuration of the metrics to the MDA	Yes
Configuration of e-Licenses	The NSSO establishes the e-Licenses associated with a particular service instance in the e-License Manager	Yes, details in Sec. 5.3
Creation of GST	A GST is created at the SM specifying the set of supported values for each one of the considered attributes.	Yes

Imperative creation of NEST (and network slice blueprint)	A NEST and associated network slice blueprint is created at the SM following the imperative approach (i.e. references to the desired resources' id is part of the provided payload)	Yes
NEST-based instantiation of RAN slices	A slice instance is created at the SM based on the provided NEST (and associated network slice blueprint)	Yes

4.3. E-Licensing Manager

The 5GZORRO platform enables a rich ecosystem in which Network Functions (NF) vendors and Operators expose and consume services from one another. The e-Licensing system adds a fundamental component to enable a production-grade telecommunication framework which will monitor the utilization of NFs commercialized through the 5GZORRO Marketplace.

4.3.1. Design Updates

The main updates during this development period are mainly due implementation decisions, that have been focused in the internal communications between the centralized part of the e-Licensing Manager and their distributed components. Thus, the e-Licensing Manager consists of a microservice-based application offered in two main subsystems:

- *e-License Manager Core* (eLMC) as a single point of presence inside the 5GZORRO Platform with the objective of centralizing TX and RX communications with the NSSO, the Marketplace, VNF Vendors and Operators. In D4.1 it was introduced as the eLicensing Context and Evaluation Manager, and now has evolved to the eLMC to centralize all the mentioned communications to the external entities, and brokering the requests to the different eLMAs, acting as the brain of the eLicensing Manager.
- *e-License Manager Agent* (eLMA) as a distributed subsystem living closer to the Operator's domain, which oversees the control of the specific Network Function instances registered under a given agreement (referred as TMF ProductOffering Object – PO – according to the Marketplace catalogue[6]).

4.3.2. Prototype implementation

The e-Licensing Manager (eLM) makes use of OpenAPIs to communicate with external services such as the Marketplace and the NSSO. These interfaces are based on REST and are available through the eLMC Swagger console which will be accessible under *<eLMC-URL>:8080/ui*. At this moment, the final deployment environment is not yet defined but an extensive set of tests have been carried out in a ATOS's testbed and are defined in section 4.3.3.

Internal communications between the different microservices that build up eLMC and eLMA are based on AMPQ protocol by making use of a RabbitMQ Broker which is initially deployed along with the eLMC. This choice enables the 5G Platform to seamlessly grow horizontally by including new Operators. The eLMA has three inner components:

1. *eLMA Rest* component is a django based service which will take the control over the local licensing registry, that is a mysql database containing the entire state of watchers and actions.
2. *eLMA MQ*: This is the client part of the MQ broker that deals with communication issues orchestrating the connection of the other sub-modules
3. *Watcher manager*: This component takes care about the lifecycle of the watchers. Each watcher in the watcher list is scheduled for its execution, providing as a result the relevant metrics for the licensing control.

The eLMC is composed by:

1. *eLMC Rest*: This component offers the endpoint to the NSSO.
2. *eLMC MQ*: This is the backend implementation of the MQ broker, managing the deployed eLMAs but also has the functionality of dispatching the POs registration through the deployed eLMAs.

The current deployment is done using docker for both the eLMA and the eLMC. In the next release, the deployment will be performed over Kubernetes.

Figure 4-9 depicts the most important steps in the registration process of a new *productOfferingPrice* (POP), and shows which component of the e-Licensing Manager carries out each part of the process. The POP object is hosted at the marketplace as a part of the Product Offer (PO). POs are used to model resources or services that the providers offer in the 5GZORRO marketplace, describing their characteristics, constraints, and specifications [7].

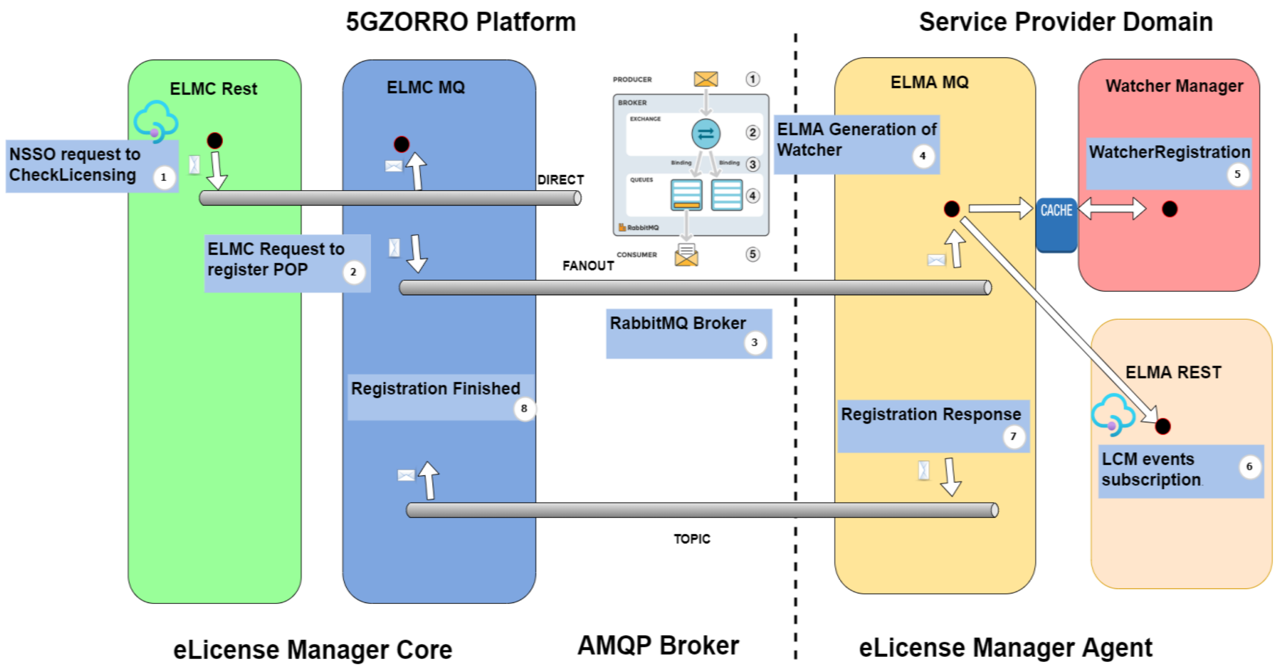


Figure 4-9: eLM Registration process

The entry point for the licensing process is the *CheckLicensing* Message (1) received through the eLMC REST component, that notifies to the eLMC the PO that will be deployed. The incoming PO (2) is translated to a POP and routed to the eLMA deployed in the domain where the PO components will be rolled out (3). The eLMA_MQ component creates the required watcher(s) (4), that are the processes customized to control a specific property of a running NFs inside the Operator's domain. The watcher is registered (5) by the Watcher Manager in the eLMA_Rest component (6). Finally, a message is sent back to the eLMC to inform that the registration is succeeded (7-8).

The first released version of eLMA is capable of monitoring NFs which are commercialized under a time-of-use based pricing. Even though the research work is still raising more flexible and advances techniques to monitor this property, the current implementation is configured to trigger an action in the event of finding out that the NF instance that was being monitored is no longer active or if the agreement has expired.

Figure 4-10 details how the current implementation of the eLM monitors the NF instances over the domains. Watchers are periodically scheduled every certain time to be executed (1) and collect relevant metrics from the running NFs inside every eLMA domain (2). When the watcher manager detects any change related to the licensing agreements of the registered POP, it sends the persist action signal to the eLMC (3).

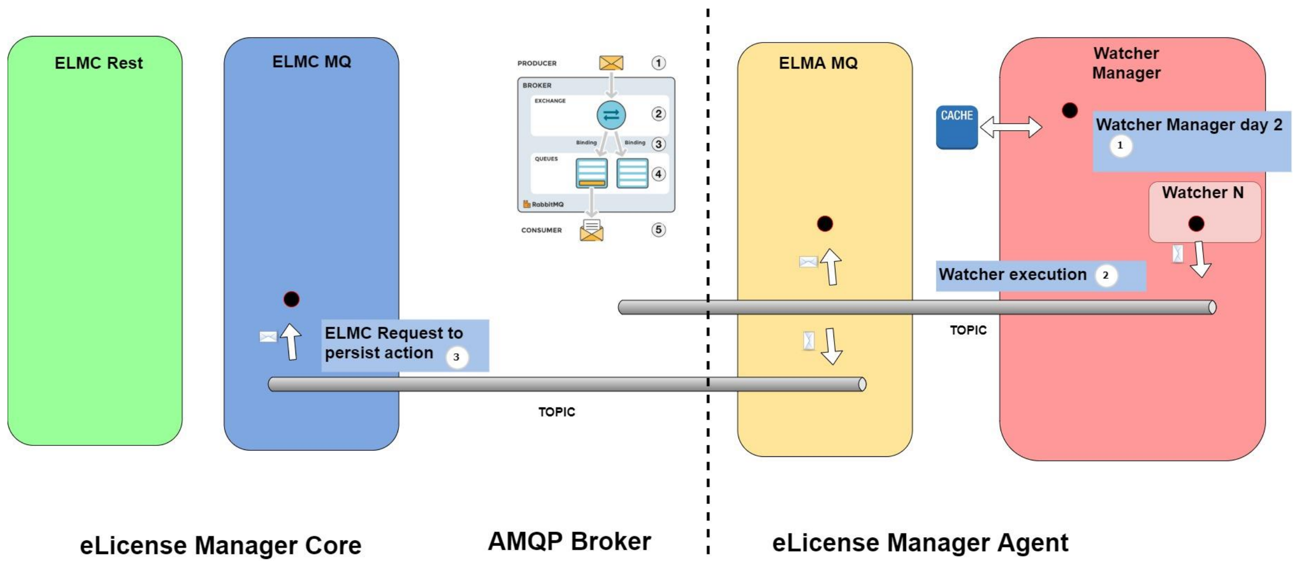


Figure 4-10: eLM Monitoring process

4.3.3. Functional tests

In this section are presented the functional tests that have been performed to the e-Licensing Manager components. These tests are further demonstrated in Section 5.3.1, where a set of screenshots of the ELM behaviour are depicted.

Table 4-4: E-Licensing manager functional tests

Name	Description	Passed (Yes/No/Partially)
Test_osm_interface	Test for the communication between the eLMA and the MANO. It includes: • Login • Get_descriptors • Get_instance_status • Describe_instances • Create_subscription • Remove_subscription	Yes
Test_registration_POP	Test for the registration of a POP in the ELMA.	Yes
Test_registration_POP_error	Registration test of non-existent POP in the marketplace	Yes
Test_registration_PO	Test for the registration of a PO in the ELMC	Yes
Test_watcher	Test for the watcher creation/deletion/update	Yes
Test_mysql	Test for the internal database creation and connection.	Yes
Test_domain	Test for the deployment in non-existing domain.	Yes
Test_NSSO_interface	Integration test with the NSSO	Yes

5. Module integration for intermediate orchestration prototype

Based on the artifacts presented in previous sections 2-4, the integration towards composing the 5GZORRO intermediate orchestration prototype is described in this section.

The integration strategy adopted by 5GZORRO aims to go beyond the classical WP-centric division of the effort, in favour forming proper design and development Teams in charge of specific areas of the 5GZORRO Platform. Such strategy facilitates the implementation of the different parts building the 5GZORRO Platform by fostering a cross-WP approach that includes right away the integration of different components.

The number and the composition of the Teams varies depending on the priorities and the requirements to be satisfied during the different phases of implementation and integration activities. So far 5GZORRO has created six Teams listed below:

- TEAM#1 – GOVERNANCE
- TEAM#2 – MARKETPLACE
- TEAM#3 – ZERO-TOUCH SLICING
- TEAM#4 – E-LICENSING
- TEAM#5 – SLA and DATALAKE
- TEAM#6 – PLATFORM

Except for TEAM#6 that is in charge of coordinating the deployment of the 5GZORRO Platform in the testbeds (5GBarcelona and 5TONIC [42]), all the other Teams are focused on the design and the implementation of different part of the 5GZORRO Platform. In this section we report the integration efforts of Teams 3, 4 and 5 which are more in scope with elements described in this deliverable, although, due to the nature of the Teams, such effort also involves components of the platform that will be reported in D3.2 and are not discussed in this document.

5.1. Integration scenario overview

As part of the integration strategy, one of the base ideas is to elaborate an integration scenario that is common to all the Teams and can be used as storyline to drive the integration of the different pieces of the 5GZORRO platform for what concerns, at least, the base functionalities.

Such a scenario starts from the creation of a resource/service offer and its publication in the marketplace by a stakeholder and terminates with its continuous post-deployment optimization, involving several components at all the levels of abstraction of the 5GZORRO platform. For the sake of readability, this document describes a part of that scenario which is more focused to the scope of the deliverable, i.e., when the ISSM comes in, detailing those pieces that have been already covered by the integration work and reported in Sections 5.2, 5.3, and, 5.4.

In Figure 5-1 the high-level steps of the integration scenario are described. Before describing them, it is important to highlight that, since parts of the platform such as the Marketplace and NFVO/VIM, are not discussed in the document, the integration scenario relies on some assumptions:

1. The target resource/service is already present in the Marketplace and related catalogues
2. Any images and/or descriptors required are onboarded on the NFVO/VIM

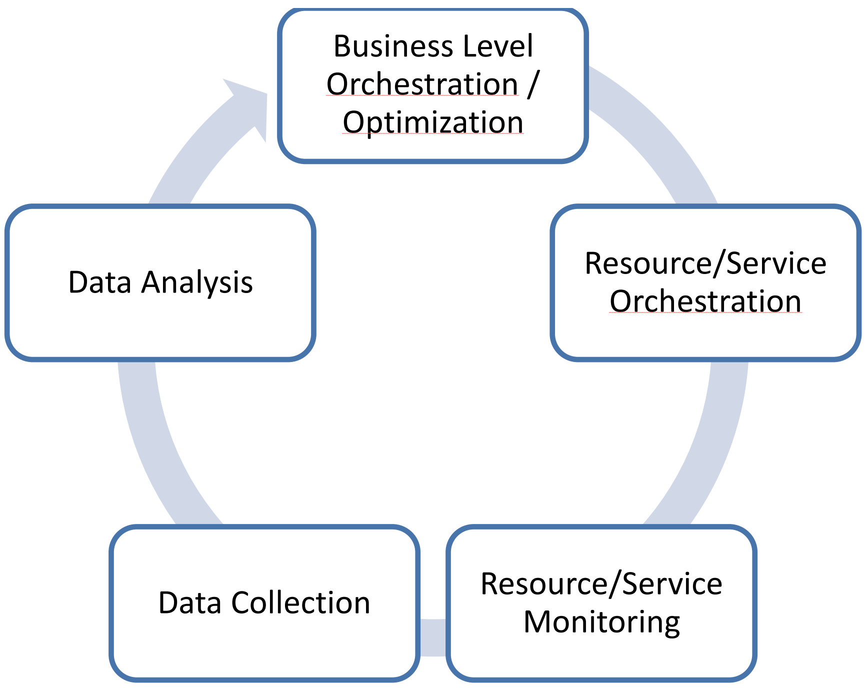

Business Level Orchestration / Optimization
Resource/Service Orchestration
Resource/Service Monitoring
Data Collection
Data Analysis

Figure 5-1: High-level steps followed by the Teams for the integration scenario

At the **Business Level Orchestration – Optimization** phase, the ISSM manages the intent-based request from a stakeholder to find and orchestrate a service/resource with certain characteristics. The ISSM-WFM relies on the ISSM-O to maximize cost-efficiency and cost-trustworthiness and interacts with Smart Resource and Service Discovery module (not reported in this document) to find suitable resource/service candidates among the ones present in the Marketplace.

Then, the ISSM-WFM builds a request to the NSSO to orchestrate the deployment, during the **Resource/Service Orchestration** phase. In this step, the NSSO verifies the service/resource license with the e-Licensing Manager then proceeds with the deployment. After the deployment operation is completed, the NSSO requests the VRM-MDA to start the monitoring of certain parameters of the resource/service so that entering in the **Resource/Service Monitoring** phase.

The VRM-MDA periodically delivers bunch of aggregated monitoring data to the 5GZORRO Data Lake (**Data Collection** phase). The data stored in the Data Lake is available for those components that perform a continuous analysis looking for failures, threats, and possible contact violations in the **Data Analysis** phase. One of these components is the SLA Breach Predictor that, as well as the Data Lake is not in the scope of this deliverable. They are reported in the integration scenario as they implement the mandatory steps that leads to the optimization of the deployed resource/service performed by the ISSM-O, again during the **Business Level Orchestration – Optimization** phase, completing the orchestration loop.

5.2. Zero-touch network slice orchestration

Figure 5-2 shows a simplified integration architecture corresponding to the essential sub-flows common to Flow 3-6 and Flow 3-7, dealing cross-domain slice establishment between two operators, explained in detail in deliverable D2.3.

The integration focused on the following major components: ISSM-WFM, NSSO, MDA, Data Lake (DL), and Identity and Permission Manager. For the rest of the components involved with Workflow 3-6, mockup APIs corresponding to those previously reported in D4.1 as well their more updated versions being reported in this deliverable.

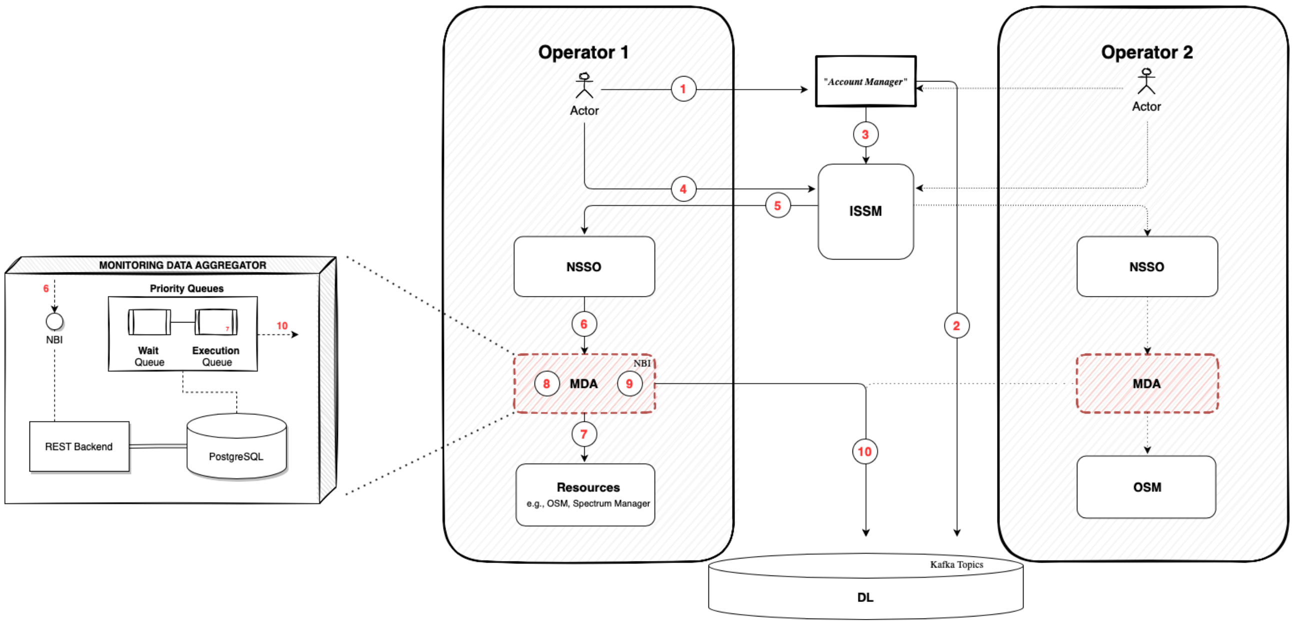


Figure 5-2: Integration Architecture of Flow 3-6 (D2.3)

It has been assumed that Operator 1 has already established a slice subnet (this is not shown in the figure) and, at some point in time, either due to optimization request or due to an explicit operator request, this slice should be scaled out into the domain of Operator 2. For this initial integration, we omit the details of resource selection, optimization, e-licensing, and acquisition (using the mockup APIs that return predefined values) and focus on facilitating the end-to-end orchestration of the workflows. The rationale behind this integration approach was to test all relevant APIs and facilitate an agile DevOps lifecycle by providing a robust workflow skeleton to which each team can independently add business logic of different components as it is being developed and this logic would be automatically tested and iteratively improved.

The slice subnet scale-out was represented by procuring a single resource in the Operator 2 domain. It was not really a slice subnet, but rather a VNF that was pre-onboarded in the respective operators' domains. This VNF has been instantiated and monitored for simple metrics like CPU and memory utilization, but we have deferred the actual 5G slice establishment with 5GC, UPF and SDN stitching to the next stage. Consequently, the product offer we used comprised just a single VNF offered by Operator 2 on the marketplace. The current integrated prototype constitutes a partial implementation of Workflows 3-6, 3-7, 3-10 (D2.3). However, this is a complete end-to end implementation that now progresses very fast to the next release.

The simplified integration workflow that has been implemented includes the following steps.

Table 5-1: ISSM integration steps

Step	Description	Input	Output	Comment
1	5GZORRO actor registers itself "Account Manager"	Operator business details	• operator ID • authToken • inTopic • outTopic • publicKey • privateKey	Account Manager is a mockup component that uses Identity and Permission Manager to uniquely identify an operator and equip it with all the needed identities and permissions to use the

Step	Description	Input	Output	Comment
				5Gzoro platform. In the current integration iteration, this component mocks the life cycle management of a platform user (agent)
2	The operator is registered in the Data Lake	• operatorID • authToken	• nameSpace • availableResources	The DL API is described in D3.1 (p.83--86)
3	The operator details are propagated to ISSM	• operatorID	none	It has been decided by the partners that in the next version of an integrated prototype, ISSM would dynamically verify operatorID against “Account Manager”. Currently, it is an implementation shortcut
4	ISSM-WFM receives a “slice instantiation” request	• NSSO format slice blueprint	ISSM executes steps of Workflow 3-6 (D2.3) related to resource discovery, optimization, and acquisition at the marketplace	For the sake of simplicity, at this stage, we avoided semantic transformations across ISSM-WFM and NSSO Another simplification was considering only the part of the workflow related to the domain of Operator 2 assuming that Operator 1 already established its slice subnet. This is justified by the fact that this is an actual common scenario when scale-out is driven by an optimization or an operator wants to extend its service area footprint. Furthermore, instantiating a slice subnet in a single domain is a standard mainstream operation while the focus of this

Step	Description	Input	Output	Comment
				project is on the cross-operator slices.
5	ISSM -WFM requests resource instantiation from NSSO	• Business Flow ID • Product Offer ID	• resourceID	Business transaction ID is automatically generated by ISSM-WFM Product Offer ID is generated by the marketplace at the time of a product offer creation resourceID is generated by NSSO upon instantiation of a product offer
6	NSSO propagates configuration with dynamic variables to MDA	• Operator ID • Business Flow ID • DL Kafka Topic • Monitoring Endpoint • Context (object) • Resource ID • Network Slice ID • Parent Resource ID • Reference ID • Metric Name/metricID • Metric Type • Aggregation Method (sum, average. etc.) • Step • Step Aggregation	none	
7	MDA fetches metric values from OSM	• metricName/metricID	• metricValue	
8	Aggregate metric by MDA component	• metricName/metricID • metricValue • aggregationMethod • step • step_aggregation	• Should something be here, e.g. 'aggregated metricValue'	
9	Hash/signing data with PK from the Operator with SHA256 and RSA algorithm	• Input keys	none	

Step	Description	Input	Output	Comment
10	Post data into Data Lake (DL Kafka topics / REST API)	• operatorID • businessID • networkID • monitoringData • resourceID • referenceID • metricName/metricID • metricValue • aggregationMethod (if the case) • timestamp • hashedData	none	At this point, monitoring data enters DL and the Intelligent SLA Breach Prediction component comprising pipelines executing in the DL can consume the data and analyse it.

5.2.1. Integration tests and results

The actual testing took place in a distributed environment bringing together premises of Nextworks, Altice Labs and IBM interconnected by VPN. The ISSM-WFM and Data Lake components have been deployed locally at IBM. The NSSO, OSM, Identity and Permission Manager and Account Manager have been deployed in Nextworks.

Since for the integration we did not possess an environment for the real instantiation with the source of the data defined, Altice Labs used a dummy orchestrator (OSM) to produce/collect metric values and, consequently, have the capability to push the encrypted hash of data into the ingestion pipeline of Data Lake for a given IN topic. This Kafka topic, along with the Business Flow ID, is obtained when the NSSO receives a blueprint describing what should be instantiated (VSB).

Altice Labs provided a REST API for the definition of the monitoring specs and this way facilitated this integration phase that endeavours to demonstrate that MDA, indeed, receives, from Vertical Slicer, a configuration stating which metrics should be monitored and aggregated.

Moreover, for testing purposes, the MDA component was deployed into the Nextworks environment, and due to this reason, ALB made available software packages to use as dependencies. Furthermore, a text configuration file was generated to declare the environment variables concerning the database applied and the operator's private key.

5.3. E-Licensing control

Every software component onboarded in the 5GZORRO Marketplace may include the tied licensing agreements associated, and with them, the definition of the costs derived from the use of this software. This integration work aims to demonstrate the process that begins after the software acquisition by any customer. From this moment, the resource is ready to be onboarded, deployed and managed by the platform.

The e-Licensing control configuration is triggered with the purpose of obtaining the relevant information about the use of this software, in accordance with the agreements that have been signed. Stamping these uses in the blockchain.

Below two workflows are depicted for illustrating the license configuration and the control of the use of the software deployed in 5GZORRO. It is worth to notice that we detected two misprints in workflows presented in deliverable D2.3 regarding the trigger of the licensing control (see D2.3). The correct sequence for the licensing check message is always from the NSSO, and not triggered by the ISSM.

The workflow of the licensing configuration is depicted in Figure 5-3.

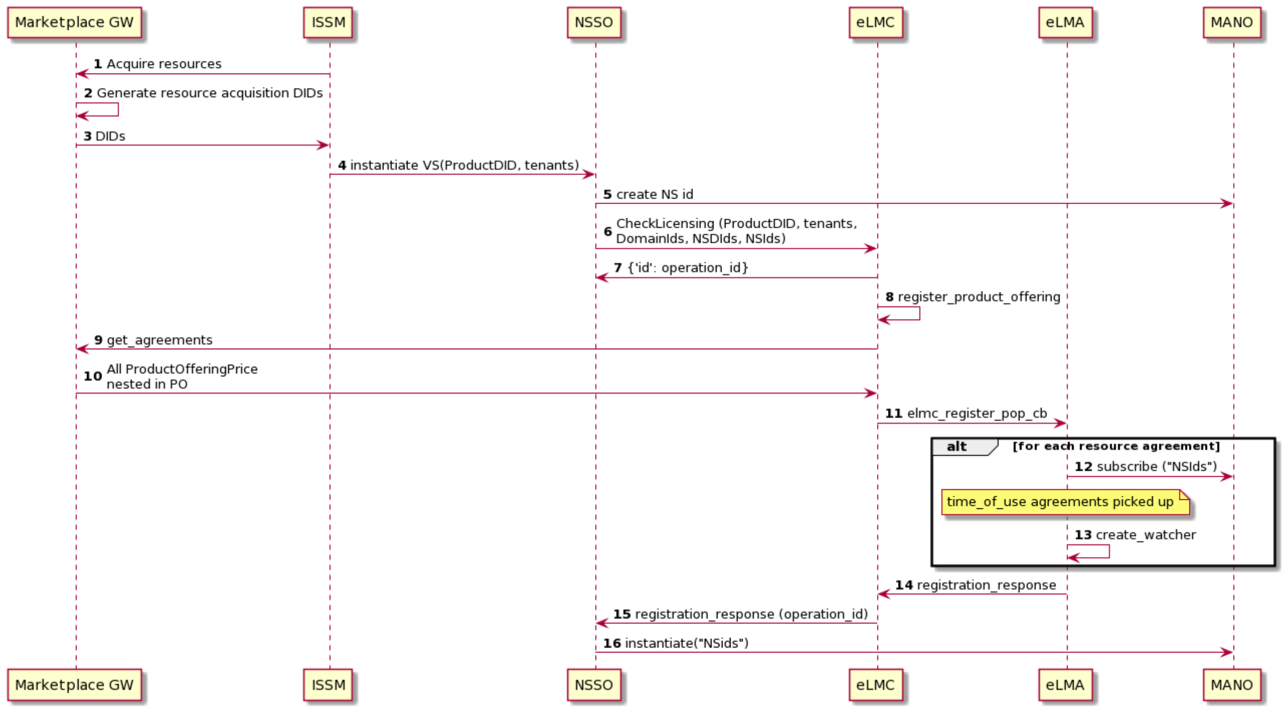

Marketplace GW
ISSM
NSSO
eLMC
eLMA
MANO
1 Acquire resources
2 Generate resource acquisition DIDs
3 DIDs
4 instantiate VS(ProductDID, tenants)
5 create NS id
6 CheckLicensing (ProductDID, tenants, DomainIds, NSDIds, NSIds)
7 {'id': operation_id}
8 register_product_offering
9 get_agreements
10 All ProductOfferingPrice nested in PO
11 elmc_register_pop_cb
alt [for each resource agreement]
12 subscribe ("NSIds")
time_of_use agreements picked up
13 create_watcher
14 registration_response
15 registration_response (operation_id)
16 instantiate("NSids")

Figure 5-3: E-License configuration workflow

The steps that are followed in the pictures are:

1. The ISSM request resources to the Marketplace. From this list of potentially heterogeneous resources, we are supposing that we have software resources to illustrate this workflow.

2-3. The Marketplace performs the whole process of the resource acquisition, generating the distributed identifiers of the products and resources.

4. The ISSM request the instantiation of the vertical slice, in certain tenants.

5. The NSSO registers the instance in the MANO, retrieving the ID of the NSI that has been registered.

6. The NSSO request the licensing checking to the eLMC, with all the information about the deployment that is processing: Product DID, domain IDs, tenant IDs, NSD IDs and NSI IDs. The NSSO waits for the service instantiation until the e-Licensing Manager verifies the NS license and process all the configurations for the software control.

7. The eLMC replies to the NSSO with an operation code that identifies the deployment intent.

8. The eLMC processes the registration of the product offer.

9-10. The eLMC request the related licensing agreements. These agreements are reflected in the nested productOfferingPrice object inside the productOffer.

11. The eLMC register the productOfferingPrice objects in the eLMAs hosted in each domain involved in the productOffer

12-13. For each resource agreement, the eLMA subscribes to the MANO instances through the OSM NBI. This subscription allows the eLMA to be aware of any event related to the instance. The watcher, who is the entity devoted to managing this subscription is created in the eLMA.

14. The eLMAs in the domains involved notifies the eLMC the result of the configuration process.

15. The eLMC notifies the response of the configuration of the licensing control to the NSSO

16. The NSSO triggers the instantiation of the NSIs in the MANO.

The workflow of the licensing working procedure is depicted in Figure 5-4.

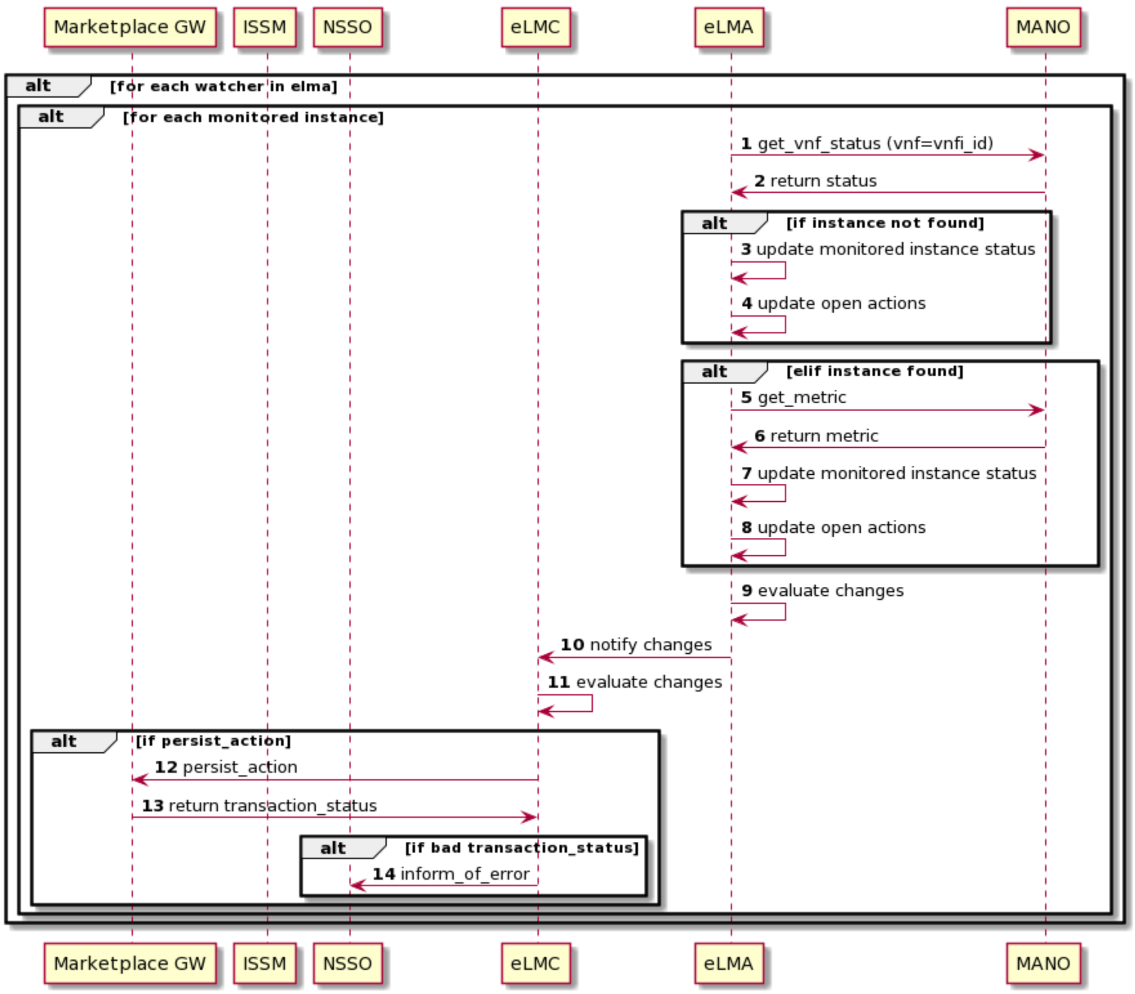

Marketplace GW
ISSM
NSSO
eLMC
eLMA
MANO
alt [for each watcher in elma]
alt [for each monitored instance]
1 get_vnf_status (vnf=vnfi_id)
2 return status
alt [if instance not found]
3 update monitored instance status
4 update open actions
alt [elif instance found]
5 get_metric
6 return metric
7 update monitored instance status
8 update open actions
9 evaluate changes
10 notify changes
11 evaluate changes
alt [if persist_action]
12 persist_action
13 return transaction_status
alt [if bad transaction_status]
14 inform_of_error

Figure 5-4: E-License control workflow

In particular,

1-2. For each monitored instance, the eLMA request the status in the MANO

3-4. If the instance is not running in the MANO, the monitored instance object and the open actions in the eLMA are updated.

5-8. If the instance is running, the eLMA retrieves the metric or set of metrics related to the licensing agreements to the MANO, this metric is updated in the status of the monitoring instance and in the open actions.

9. Depending on the status of the monitored instance and the actions, the eLMA evaluates comparing with the previous status.

10. The eLMA notifies to the eLMC the changes of the monitored instances in his domain.

11. Since the eLMC has the global view of all the eLMAs and can compare the status of all the deployments, decide if an action needs to be submitted for persisting.

12-14. If the eLMC decides to submit an action for storing, this action is sent to the Marketplace that will create a transaction with the action in the DLT. In case of error in the transaction, the NSSO is notified.

5.3.1. Integration tests and results

The environment selected for the early integration is ATOS testbed. Once these tests are passed, the 5GZORRO components will be deployed to the 5GBarcelona testbed for the integration with further components. The VNF used in this integration is not functional, since the objective of this effort is to test the interfaces and the behaviour of both NSSO and ELM.

The interaction with the Marketplace is mocked for the moment for two workflows (Figure 5-3 and Figure 5-4). At this stage of integration, the licensing agreements (steps 9-10) are pre-created and hosted inside the e-Licensing Manager. The work in progress outcomes for the next period are the Licensing agreements creation in the 5GZORRO marketplace through the portal GUI, and the interface between the e-Licensing manager to request the signed agreements. Besides, the interface between the e-Licensing Manager and the Marketplace for the action persisting is work in progress, so it is mocked at this stage (steps 12-13 in Figure 5-4).

A set of HTTP REST calls are used to configure the NSSO externally described in Table 5-2:

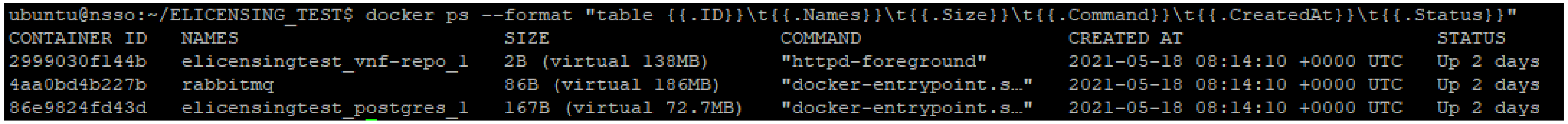
Table 5-2: NSSO HTTP rest requests used

HTTP REST call	Description
Login_domain	Login in the NSSO platform
Onboard_vsb_mysql	Onboard of the vertical slice blueprint. That is: VNFDs, NSDs, etc
Onboard_vsd_mysql	Onboard the vertical slice descriptor
Instantiate_mysql	Covers step 5 and step 6 in Figure 5-3, registering the NSI in the MANO and triggering the licensing checking

A set of screenshots are presented to illustrate the integration with the NSSO and ELM software components. They are presented in three groups to facilitate the comprehension of the thread in each component:

(1) The first group of screenshots demonstrates the licensing verification before the NS instantiation, and also the configuration of the watchers in the e-Licensing Manager to control the use of the NS.

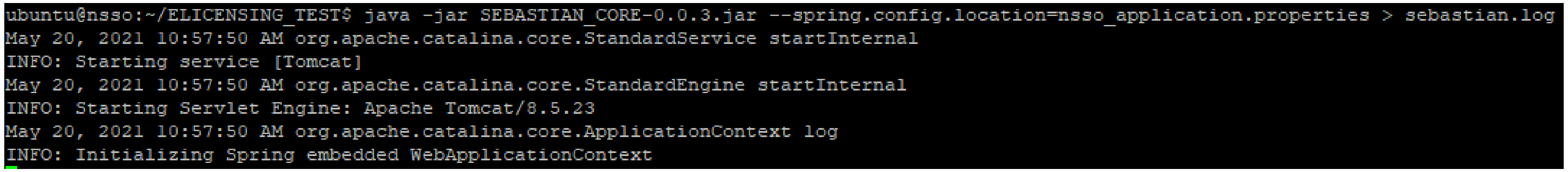
The NSSO deployment is composed by 3 docker containers in Figure 5-5 with the postgres database, the rabbitmq broker instance and the apache webserver that hosts the VNF metadata:



```
ubuntu@nsso:~/ELICENSING_TEST$ docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Size}}\t{{.Command}}\t{{.CreatedAt}}\t{{.Status}}"
CONTAINER ID   NAMES                        SIZE                    COMMAND                  CREATED AT                        STATUS
2999030f144b   elicensingtest_vnf-repo_1    2B (virtual 138MB)      "httpd-foreground"       2021-05-18 08:14:10 +0000 UTC     Up 2 days
4aa0bd4b227b   rabbitmq                     86B (virtual 186MB)     "docker-entrypoint.s…"   2021-05-18 08:14:10 +0000 UTC     Up 2 days
86e9824fd43d   elicensingtest_postgres_1    167B (virtual 72.7MB)   "docker-entrypoint.s…"   2021-05-18 08:14:10 +0000 UTC     Up 2 days
```

Figure 5-5: NSSO docker components deployment

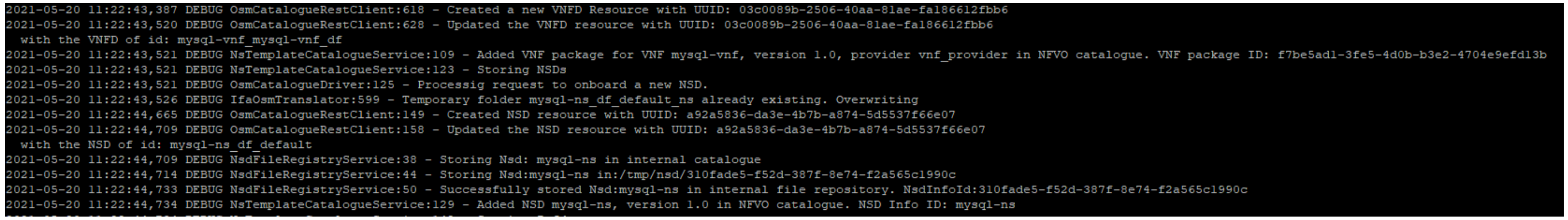
Also, the NSSO logs are presented in Figure 5-6



```
ubuntu@nsso:~/ELICENSING_TEST$ java -jar SEBASTIAN_CORE-0.0.3.jar --spring.config.location=nsso_application.properties > sebastian.log
May 20, 2021 10:57:50 AM org.apache.catalina.core.StandardService startInternal
INFO: Starting service [Tomcat]
May 20, 2021 10:57:50 AM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: Apache Tomcat/8.5.23
May 20, 2021 10:57:50 AM org.apache.catalina.core.ApplicationContext log
INFO: Initializing Spring embedded WebApplicationContext
```

Figure 5-6: NSSO deployment

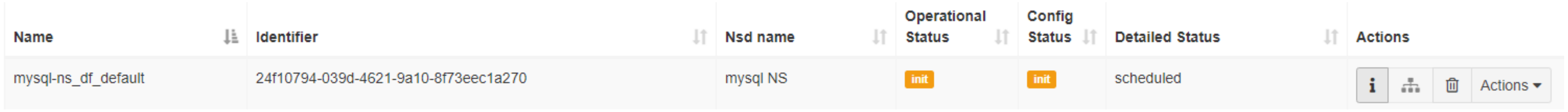
Using the Onboard_vsb_mysql HTTP call, the VNF/NS descriptors are correctly onboarded in the MANO, as shown in Figure 5-7:



```
2021-05-20 11:22:43,387 DEBUG OsmCatalogueRestClient:618 - Created a new VNFD Resource with UUID: 03c0089b-2506-40aa-81ae-fa186612fbb6
2021-05-20 11:22:43,520 DEBUG OsmCatalogueRestClient:628 - Updated the VNFD resource with UUID: 03c0089b-2506-40aa-81ae-fa186612fbb6
  with the VNFD of id: mysql-vnf_mysql-vnf_df
2021-05-20 11:22:43,521 DEBUG NsTemplateCatalogueService:109 - Added VNF package for VNF mysql-vnf, version 1.0, provider vnf_provider in NFVO catalogue. VNF package ID: f7be5ad1-3fe5-4d0b-b3e2-4704e9efd13b
2021-05-20 11:22:43,521 DEBUG NsTemplateCatalogueService:123 - Storing NSDs
2021-05-20 11:22:43,521 DEBUG OsmCatalogueDriver:125 - Processig request to onboard a new NSD.
2021-05-20 11:22:43,526 DEBUG IfaOsmTranslator:599 - Temporary folder mysql-ns_df_default_ns already existing. Overwriting
2021-05-20 11:22:44,665 DEBUG OsmCatalogueRestClient:149 - Created NSD resource with UUID: a92a5836-da3e-4b7b-a874-5d5537f66e07
2021-05-20 11:22:44,709 DEBUG OsmCatalogueRestClient:158 - Updated the NSD resource with UUID: a92a5836-da3e-4b7b-a874-5d5537f66e07
  with the NSD of id: mysql-ns_df_default
2021-05-20 11:22:44,709 DEBUG NsdFileRegistryService:38 - Storing Nsd: mysql-ns in internal catalogue
2021-05-20 11:22:44,714 DEBUG NsdFileRegistryService:44 - Storing Nsd:mysql-ns in:/tmp/nsd/310fade5-f52d-387f-8e74-f2a565c1990c
2021-05-20 11:22:44,733 DEBUG NsdFileRegistryService:50 - Successfully stored Nsd:mysql-ns in internal file repository. NsdInfoId:310fade5-f52d-387f-8e74-f2a565c1990c
2021-05-20 11:22:44,734 DEBUG NsTemplateCatalogueService:129 - Added NSD mysql-ns, version 1.0 in NFVO catalogue. NSD Info ID: mysql-ns
```

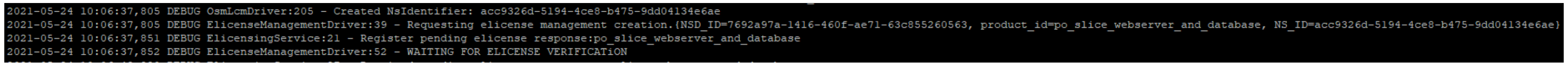
Figure 5-7: Onboard of VNF and NS descriptors in the OSM NFVO by the NSSO

Once HTTP REST requests in Table 5-2 are completed, the NSI is registered in OSM (Figure 5-8). This is step 5 at Figure 5-3. At this point the NSSO request the ELM to process checkLicensing message that is displayed in Figure 5-9 and waits for the license verification and configuration. Once the NSSO is notified about the license verification, sends the request to the NFVO to instantiate the NSi (Figure 5-10). In Figure 5-11 is displayed the NSi instance running.



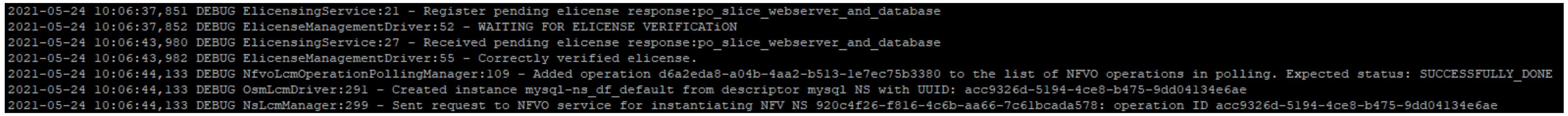
Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
mysql-ns_df_default	24f10794-039d-4621-9a10-8f73eec1a270	mysql NS	init	init	scheduled	Actions

Figure 5-8: NSI registered is OSM NFVO



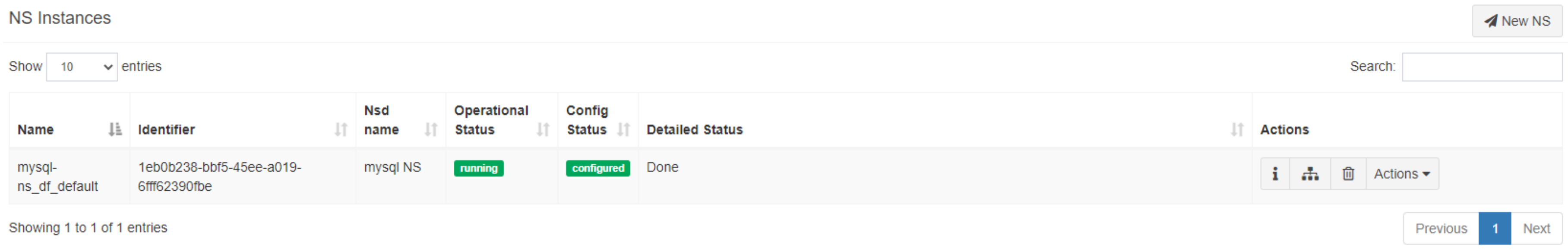
```
2021-05-24 10:06:37,805 DEBUG OsmLcmDriver:205 - Created NsIdentifier: acc9326d-5194-4ce8-b475-9dd04134e6ae
2021-05-24 10:06:37,805 DEBUG ElicenseManagementDriver:39 - Requesting elicense management creation.{NSD_ID=7692a97a-1416-460f-ae71-63c855260563, product_id=po_slice_webserver_and_database, NS_ID=acc9326d-5194-4ce8-b475-9dd04134e6ae}
2021-05-24 10:06:37,851 DEBUG ElicensingService:21 - Register pending elicense response:po_slice_webserver_and_database
2021-05-24 10:06:37,852 DEBUG ElicenseManagementDriver:52 - WAITING FOR ELICENSE VERIFICATiON
```

Figure 5-9: CheckLicensing request from the NSSO to the ELM



```
2021-05-24 10:06:37,851 DEBUG ElicensingService:21 - Register pending elicense response:po_slice_webserver_and_database
2021-05-24 10:06:37,852 DEBUG ElicenseManagementDriver:52 - WAITING FOR ELICENSE VERIFICATiON
2021-05-24 10:06:43,980 DEBUG ElicensingService:27 - Received pending elicense response:po_slice_webserver_and_database
2021-05-24 10:06:43,982 DEBUG ElicenseManagementDriver:55 - Correctly verified elicense.
2021-05-24 10:06:44,133 DEBUG NfvoLcmOperationPollingManager:109 - Added operation d6a2eda8-a04b-4aa2-b513-1e7ec75b3380 to the list of NFVO operations in polling. Expected status: SUCCESSFULLY_DONE
2021-05-24 10:06:44,133 DEBUG OsmLcmDriver:291 - Created instance mysql-ns_df_default from descriptor mysql NS with UUID: acc9326d-5194-4ce8-b475-9dd04134e6ae
2021-05-24 10:06:44,133 DEBUG NsLcmManager:299 - Sent request to NFVO service for instantiating NFV NS 920c4f26-f816-4c6b-aa66-7c61bcada578: operation ID acc9326d-5194-4ce8-b475-9dd04134e6ae
```

Figure 5-10: License verified and service instantiation



NS Instances

New NS

Show 10 entries

Search:

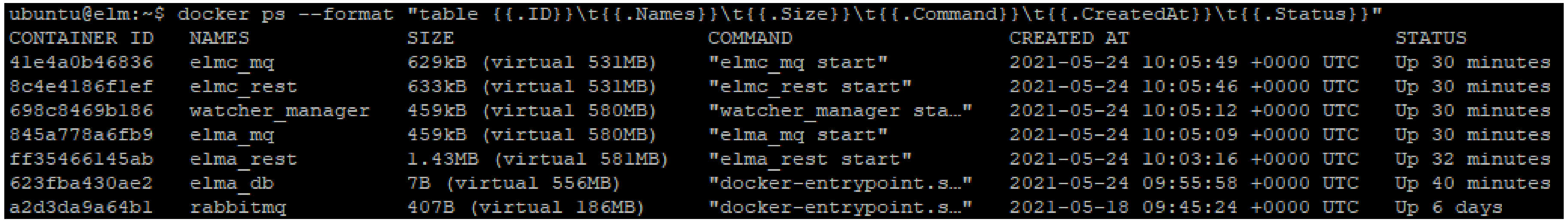
Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
mysql-ns_df_default	1eb0b238-bbf5-45ee-a019-6fff62390fbe	mysql NS	running	configured	Done	Actions

Showing 1 to 1 of 1 entries

Previous 1 Next

Figure 5-11: NS instance deployed at OSM after the license verification.

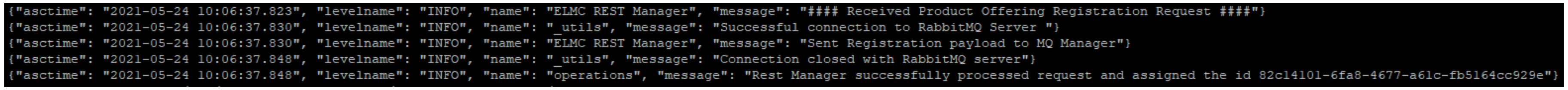
From the e-Licensing manager perspective, the ELM deployment is dockerized and composed by 7 containers as described in Figure 5-12:



```
ubuntu@elm:~$ docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Size}}\t{{.Command}}\t{{.CreatedAt}}\t{{.Status}}"
CONTAINER ID   NAMES             SIZE                   COMMAND                    CREATED AT                      STATUS
41e4a0b46836   elmc_mq           629kB (virtual 531MB)  "elmc_mq start"            2021-05-24 10:05:49 +0000 UTC   Up 30 minutes
8c4e4186f1ef   elmc_rest         633kB (virtual 531MB)  "elmc_rest start"          2021-05-24 10:05:46 +0000 UTC   Up 30 minutes
698c8469b186   watcher_manager   459kB (virtual 580MB)  "watcher_manager sta…"     2021-05-24 10:05:12 +0000 UTC   Up 30 minutes
845a778a6fb9   elma_mq           459kB (virtual 580MB)  "elma_mq start"            2021-05-24 10:05:09 +0000 UTC   Up 30 minutes
ff35466145ab   elma_rest         1.43MB (virtual 581MB) "elma_rest start"          2021-05-24 10:03:16 +0000 UTC   Up 32 minutes
623fba430ae2   elma_db           7B (virtual 556MB)     "docker-entrypoint.s…"     2021-05-24 09:55:58 +0000 UTC   Up 40 minutes
a2d3da9a64b1   rabbitmq          407B (virtual 186MB)   "docker-entrypoint.s…"     2021-05-18 09:45:24 +0000 UTC   Up 6 days
```

Figure 5-12: ELM docker container deployment

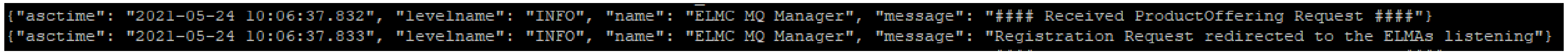
Once the NSSO triggers the checkLicensing request, the ELMC_REST receives the message and extracts the product offer and delegates to ELMC_MQ the distribution of the request.



```
{"asctime": "2021-05-24 10:06:37.823", "levelname": "INFO", "name": "ELMC REST Manager", "message": "#### Received Product Offering Registration Request ####"}
{"asctime": "2021-05-24 10:06:37.830", "levelname": "INFO", "name": "_utils", "message": "Successful connection to RabbitMQ Server "}
{"asctime": "2021-05-24 10:06:37.830", "levelname": "INFO", "name": "ELMC REST Manager", "message": "Sent Registration payload to MQ Manager"}
{"asctime": "2021-05-24 10:06:37.848", "levelname": "INFO", "name": "_utils", "message": "Connection closed with RabbitMQ server"}
{"asctime": "2021-05-24 10:06:37.848", "levelname": "INFO", "name": "operations", "message": "Rest Manager successfully processed request and assigned the id 82c14101-6fa8-4677-a61c-fb5164cc929e"}
```

Figure 5-13: ELMC_REST process for the PO registration

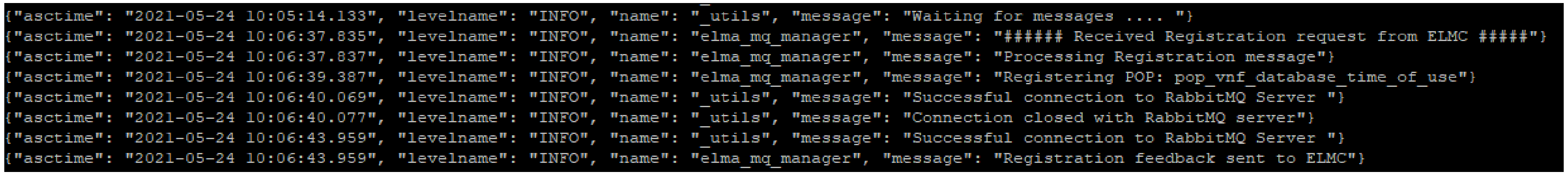
ELMC_MQ receives the PO requests from the ELMC_REST and redirects the registration message to the ELMAs registered in the system.



```
{"asctime": "2021-05-24 10:06:37.832", "levelname": "INFO", "name": "ELMC MQ Manager", "message": "#### Received ProductOffering Request ####"}
{"asctime": "2021-05-24 10:06:37.833", "levelname": "INFO", "name": "ELMC MQ Manager", "message": "Registration Request redirected to the ELMAs listening"}
```

Figure 5-14: ELMC_MQ redirect the registration message

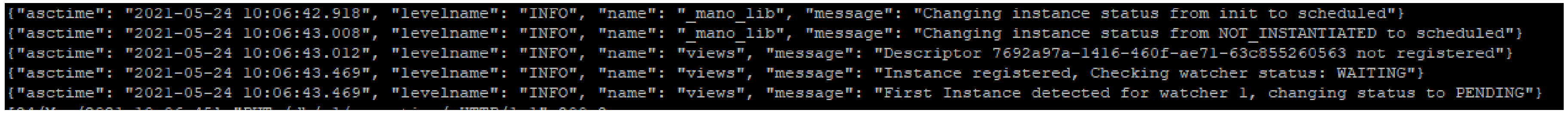
ELMA_MQ receives the registration request and corroborates the POP existence in the marketplace. Once the POP has been registered and watchers are configured, sends the message of registration successful to the ELMC.



```
{"asctime": "2021-05-24 10:05:14.133", "levelname": "INFO", "name": "_utils", "message": "Waiting for messages .... "}
{"asctime": "2021-05-24 10:06:37.835", "levelname": "INFO", "name": "elma_mq_manager", "message": "###### Received Registration request from ELMC #####"}
{"asctime": "2021-05-24 10:06:37.837", "levelname": "INFO", "name": "elma_mq_manager", "message": "Processing Registration message"}
{"asctime": "2021-05-24 10:06:39.387", "levelname": "INFO", "name": "elma_mq_manager", "message": "Registering POP: pop_vnf_database_time_of_use"}
{"asctime": "2021-05-24 10:06:40.069", "levelname": "INFO", "name": "_utils", "message": "Successful connection to RabbitMQ Server "}
{"asctime": "2021-05-24 10:06:40.077", "levelname": "INFO", "name": "_utils", "message": "Connection closed with RabbitMQ server"}
{"asctime": "2021-05-24 10:06:43.959", "levelname": "INFO", "name": "_utils", "message": "Successful connection to RabbitMQ Server "}
{"asctime": "2021-05-24 10:06:43.959", "levelname": "INFO", "name": "elma_mq_manager", "message": "Registration feedback sent to ELMC"}
```

Figure 5-15: ELMA_MQ registers the POP and send ACK to ELMC_MQ

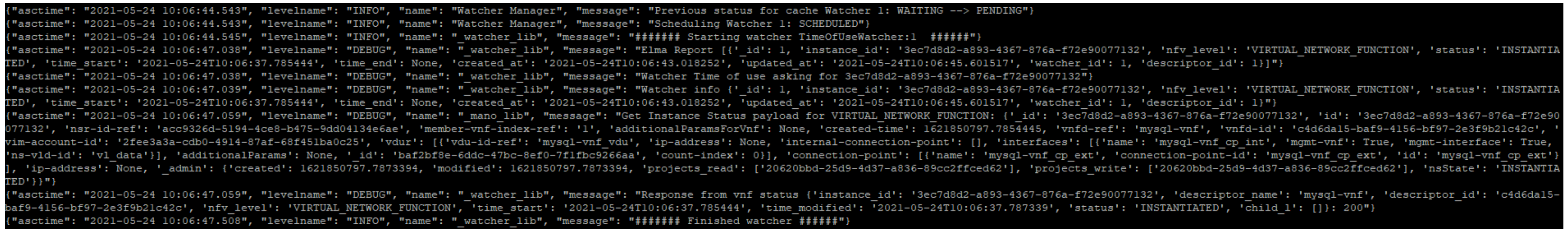
ELMA Rest detects the instances that are scheduled in the NFVO that are being evaluated for the licensing check. ELMA_REST request the watcher registration:



```
{"asctime": "2021-05-24 10:06:42.918", "levelname": "INFO", "name": "_mano_lib", "message": "Changing instance status from init to scheduled"}
{"asctime": "2021-05-24 10:06:43.008", "levelname": "INFO", "name": "_mano_lib", "message": "Changing instance status from NOT_INSTANTIATED to scheduled"}
{"asctime": "2021-05-24 10:06:43.012", "levelname": "INFO", "name": "views", "message": "Descriptor 7692a97a-1416-460f-ae71-63c855260563 not registered"}
{"asctime": "2021-05-24 10:06:43.469", "levelname": "INFO", "name": "views", "message": "Instance registered, Checking watcher status: WAITING"}
{"asctime": "2021-05-24 10:06:43.469", "levelname": "INFO", "name": "views", "message": "First Instance detected for watcher 1, changing status to PENDING"}
```

Figure 5-16: ELMA_REST OSM scheduled NSIs detection

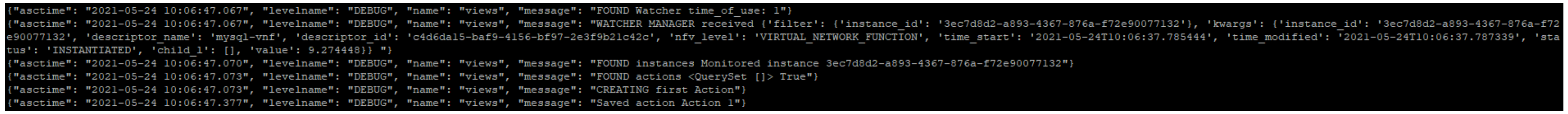
The watcher manager detects the instantiation of the NSI and register the time when it has been deployed



```
{"asctime": "2021-05-24 10:06:44.543", "levelname": "INFO", "name": "Watcher Manager", "message": "Previous status for cache Watcher 1: WAITING --> PENDING"}
{"asctime": "2021-05-24 10:06:44.543", "levelname": "INFO", "name": "Watcher Manager", "message": "Scheduling Watcher 1: SCHEDULED"}
{"asctime": "2021-05-24 10:06:44.545", "levelname": "INFO", "name": "_watcher_lib", "message": "####### Starting watcher TimeOfUseWatcher:1  ######"}
{"asctime": "2021-05-24 10:06:47.038", "levelname": "DEBUG", "name": "_watcher_lib", "message": "Elma Report [{'_id': 1, 'instance_id': '3ec7d8d2-a893-4367-876a-f72e90077132', 'nfv_level': 'VIRTUAL_NETWORK_FUNCTION', 'status': 'INSTANTIA
TED', 'time_start': '2021-05-24T10:06:37.785444', 'time_end': None, 'created_at': '2021-05-24T10:06:43.018252', 'updated_at': '2021-05-24T10:06:45.601517', 'watcher_id': 1, 'descriptor_id': 1}]"}
{"asctime": "2021-05-24 10:06:47.038", "levelname": "DEBUG", "name": "_watcher_lib", "message": "Watcher Time of use asking for 3ec7d8d2-a893-4367-876a-f72e90077132"}
{"asctime": "2021-05-24 10:06:47.039", "levelname": "DEBUG", "name": "_watcher_lib", "message": "Watcher info {'_id': 1, 'instance_id': '3ec7d8d2-a893-4367-876a-f72e90077132', 'nfv_level': 'VIRTUAL_NETWORK_FUNCTION', 'status': 'INSTANTIA
TED', 'time_start': '2021-05-24T10:06:37.785444', 'time_end': None, 'created_at': '2021-05-24T10:06:43.018252', 'updated_at': '2021-05-24T10:06:45.601517', 'watcher_id': 1, 'descriptor_id': 1}"}
{"asctime": "2021-05-24 10:06:47.059", "levelname": "DEBUG", "name": "_mano_lib", "message": "Get Instance Status payload for VIRTUAL_NETWORK_FUNCTION: {'_id': '3ec7d8d2-a893-4367-876a-f72e90077132', 'id': '3ec7d8d2-a893-4367-876a-f72e90
077132', 'nsr-id-ref': 'acc9326d-5194-4ce8-b475-9dd04134e6ae', 'member-vnf-index-ref': '1', 'additionalParamsForVnf': None, 'created-time': 1621850797.7854445, 'vnfd-ref': 'mysql-vnf', 'vnfd-id': 'c4d6da15-baf9-4156-bf97-2e3f9b21c42c', '
vim-account-id': '2fee3a3a-cdb0-4914-87af-68f451ba0c25', 'vdur': [{'vdu-id-ref': 'mysql-vnf_vdu', 'ip-address': None, 'internal-connection-point': [], 'interfaces': [{'name': 'mysql-vnf_cp_int', 'mgmt-vnf': True, 'mgmt-interface': True,
'ns-vld-id': 'vl_data'}], 'additionalParams': None, '_id': 'baf2bf8e-6ddc-47bc-8ef0-7f1fbc9266aa', 'count-index': 0}], 'connection-point': [{'name': 'mysql-vnf_cp_ext', 'connection-point-id': 'mysql-vnf_cp_ext', 'id': 'mysql-vnf_cp_ext'}
], 'ip-address': None, '_admin': {'created': 1621850797.7873394, 'modified': 1621850797.7873394, 'projects_read': ['20620bbd-25d9-4d37-a836-89cc2ffced62'], 'projects_write': ['20620bbd-25d9-4d37-a836-89cc2ffced62'], 'nsState': 'INSTANTIA
TED'}}"}
{"asctime": "2021-05-24 10:06:47.059", "levelname": "DEBUG", "name": "_watcher_lib", "message": "Response from vnf status {'instance_id': '3ec7d8d2-a893-4367-876a-f72e90077132', 'descriptor_name': 'mysql-vnf', 'descriptor_id': 'c4d6da15-
baf9-4156-bf97-2e3f9b21c42c', 'nfv_level': 'VIRTUAL_NETWORK_FUNCTION', 'time_start': '2021-05-24T10:06:37.785444', 'time_modified': '2021-05-24T10:06:37.787339', 'status': 'INSTANTIATED', 'child_1': []}: 200"}
{"asctime": "2021-05-24 10:06:47.508", "levelname": "INFO", "name": "_watcher_lib", "message": "####### Finished watcher ######"}
```

Figure 5-17: Watcher Manager execution

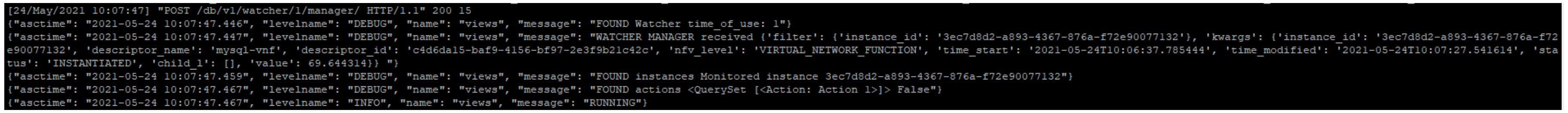
The ELMA_REST detects the watcher and creates an action. In this action will be registered the use that is being performed (time of use in this case).



```
{"asctime": "2021-05-24 10:06:47.067", "levelname": "DEBUG", "name": "views", "message": "FOUND Watcher time_of_use: 1"}
{"asctime": "2021-05-24 10:06:47.067", "levelname": "DEBUG", "name": "views", "message": "WATCHER MANAGER received {'filter': {'instance_id': '3ec7d8d2-a893-4367-876a-f72e90077132'}, 'kwargs': {'instance_id': '3ec7d8d2-a893-4367-876a-f72
e90077132', 'descriptor_name': 'mysql-vnf', 'descriptor_id': 'c4d6da15-baf9-4156-bf97-2e3f9b21c42c', 'nfv_level': 'VIRTUAL_NETWORK_FUNCTION', 'time_start': '2021-05-24T10:06:37.785444', 'time_modified': '2021-05-24T10:06:37.787339', 'sta
tus': 'INSTANTIATED', 'child_1': [], 'value': 9.274448}} "}
{"asctime": "2021-05-24 10:06:47.070", "levelname": "DEBUG", "name": "views", "message": "FOUND instances Monitored instance 3ec7d8d2-a893-4367-876a-f72e90077132"}
{"asctime": "2021-05-24 10:06:47.073", "levelname": "DEBUG", "name": "views", "message": "FOUND actions <QuerySet []> True"}
{"asctime": "2021-05-24 10:06:47.073", "levelname": "DEBUG", "name": "views", "message": "CREATING first Action"}
{"asctime": "2021-05-24 10:06:47.377", "levelname": "DEBUG", "name": "views", "message": "Saved action Action 1"}
```

Figure 5-18: Action creation in ELMA_REST component

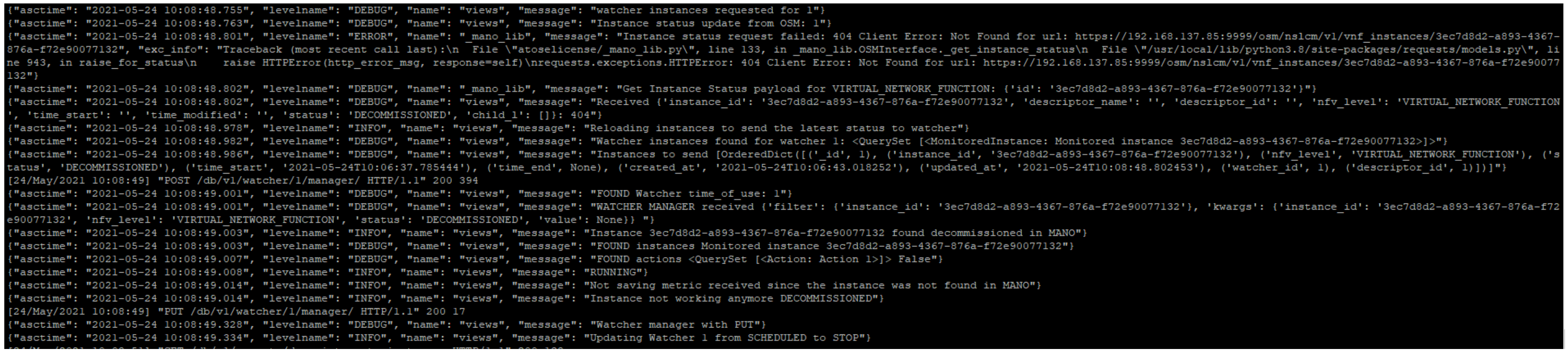
(2) The second group of screenshots describe how the control of the software is performed. The process starts in the ELMA_REST component. Each 5 seconds, all the watchers configured in the watcher manager are scheduled for execution. Thus, in the ELMA_REST component retrieves this execution, verifies that the instance is at RUNNING state and updates the time of use metric.



```
[24/May/2021 10:07:47] "POST /db/v1/watcher/1/manager/ HTTP/1.1" 200 15
{"asctime": "2021-05-24 10:07:47.446", "levelname": "DEBUG", "name": "views", "message": "FOUND Watcher time_of_use: 1"}
{"asctime": "2021-05-24 10:07:47.447", "levelname": "DEBUG", "name": "views", "message": "WATCHER MANAGER received {'filter': {'instance_id': '3ec7d8d2-a893-4367-876a-f72e90077132'}, 'kwargs': {'instance_id': '3ec7d8d2-a893-4367-876a-f72
e90077132', 'descriptor_name': 'mysql-vnf', 'descriptor_id': 'c4d6da15-baf9-4156-bf97-2e3f9b21c42c', 'nfv_level': 'VIRTUAL_NETWORK_FUNCTION', 'time_start': '2021-05-24T10:06:37.785444', 'time_modified': '2021-05-24T10:07:27.541614', 'sta
tus': 'INSTANTIATED', 'child_1': [], 'value': 69.644314}} "}
{"asctime": "2021-05-24 10:07:47.459", "levelname": "DEBUG", "name": "views", "message": "FOUND instances Monitored instance 3ec7d8d2-a893-4367-876a-f72e90077132"}
{"asctime": "2021-05-24 10:07:47.467", "levelname": "DEBUG", "name": "views", "message": "FOUND actions <QuerySet [<Action: Action 1>]> False"}
{"asctime": "2021-05-24 10:07:47.467", "levelname": "INFO", "name": "views", "message": "RUNNING"}
```

Figure 5-19: ELMA_REST watcher control.

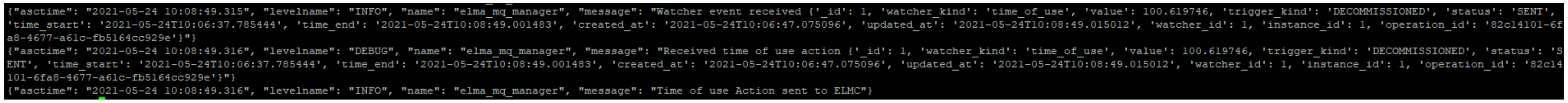
In this case we have manually removed the deployed NSi from OSM. This operation is detected by the ELMA_REST as displayed in Figure 5-20, so the watcher is stopped and a notification is sent to the ELMA_MQ:



```
{"asctime": "2021-05-24 10:08:48.755", "levelname": "DEBUG", "name": "views", "message": "watcher instances requested for 1"}
{"asctime": "2021-05-24 10:08:48.763", "levelname": "DEBUG", "name": "views", "message": "Instance status update from OSM: 1"}
{"asctime": "2021-05-24 10:08:48.801", "levelname": "ERROR", "name": "_mano_lib", "message": "Instance status request failed: 404 Client Error: Not Found for url: https://192.168.137.85:9999/osm/nslcm/v1/vnf_instances/3ec7d8d2-a893-4367-
876a-f72e90077132", "exc_info": "Traceback (most recent call last):\n  File \"atoselicense/_mano_lib.py\", line 133, in _mano_lib.OSMInterface._get_instance_status\n  File \"/usr/local/lib/python3.8/site-packages/requests/models.py\", li
ne 943, in raise_for_status\n    raise HTTPError(http_error_msg, response=self)\nrequests.exceptions.HTTPError: 404 Client Error: Not Found for url: https://192.168.137.85:9999/osm/nslcm/v1/vnf_instances/3ec7d8d2-a893-4367-876a-f72e90077
132"}
{"asctime": "2021-05-24 10:08:48.802", "levelname": "DEBUG", "name": "_mano_lib", "message": "Get Instance Status payload for VIRTUAL_NETWORK_FUNCTION: {'id': '3ec7d8d2-a893-4367-876a-f72e90077132'}"}
{"asctime": "2021-05-24 10:08:48.802", "levelname": "DEBUG", "name": "views", "message": "Received {'instance_id': '3ec7d8d2-a893-4367-876a-f72e90077132', 'descriptor_name': '', 'descriptor_id': '', 'nfv_level': 'VIRTUAL_NETWORK_FUNCTION
', 'time_start': '', 'time_modified': '', 'status': 'DECOMMISSIONED', 'child_1': []}: 404"}
{"asctime": "2021-05-24 10:08:48.978", "levelname": "INFO", "name": "views", "message": "Reloading instances to send the latest status to watcher"}
{"asctime": "2021-05-24 10:08:48.982", "levelname": "DEBUG", "name": "views", "message": "Watcher instances found for watcher 1: <QuerySet [<MonitoredInstance: Monitored instance 3ec7d8d2-a893-4367-876a-f72e90077132>]>"}
{"asctime": "2021-05-24 10:08:48.986", "levelname": "DEBUG", "name": "views", "message": "Instances to send [OrderedDict([('_id', 1), ('instance_id', '3ec7d8d2-a893-4367-876a-f72e90077132'), ('nfv_level', 'VIRTUAL_NETWORK_FUNCTION'), ('s
tatus', 'DECOMMISSIONED'), ('time_start', '2021-05-24T10:06:37.785444'), ('time_end', None), ('created_at', '2021-05-24T10:06:43.018252'), ('updated_at', '2021-05-24T10:08:48.802453'), ('watcher_id', 1), ('descriptor_id', 1)])]"}
[24/May/2021 10:08:49] "POST /db/v1/watcher/1/manager/ HTTP/1.1" 200 394
{"asctime": "2021-05-24 10:08:49.001", "levelname": "DEBUG", "name": "views", "message": "FOUND Watcher time_of_use: 1"}
{"asctime": "2021-05-24 10:08:49.001", "levelname": "DEBUG", "name": "views", "message": "WATCHER MANAGER received {'filter': {'instance_id': '3ec7d8d2-a893-4367-876a-f72e90077132'}, 'kwargs': {'instance_id': '3ec7d8d2-a893-4367-876a-f72
e90077132', 'nfv_level': 'VIRTUAL_NETWORK_FUNCTION', 'status': 'DECOMMISSIONED', 'value': None}} "}
{"asctime": "2021-05-24 10:08:49.003", "levelname": "INFO", "name": "views", "message": "Instance 3ec7d8d2-a893-4367-876a-f72e90077132 found decommissioned in MANO"}
{"asctime": "2021-05-24 10:08:49.003", "levelname": "DEBUG", "name": "views", "message": "FOUND instances Monitored instance 3ec7d8d2-a893-4367-876a-f72e90077132"}
{"asctime": "2021-05-24 10:08:49.007", "levelname": "DEBUG", "name": "views", "message": "FOUND actions <QuerySet [<Action: Action 1>]> False"}
{"asctime": "2021-05-24 10:08:49.008", "levelname": "INFO", "name": "views", "message": "RUNNING"}
{"asctime": "2021-05-24 10:08:49.014", "levelname": "INFO", "name": "views", "message": "Not saving metric received since the instance was not found in MANO"}
{"asctime": "2021-05-24 10:08:49.014", "levelname": "INFO", "name": "views", "message": "Instance not working anymore DECOMMISSIONED"}
[24/May/2021 10:08:49] "PUT /db/v1/watcher/1/manager/ HTTP/1.1" 200 17
{"asctime": "2021-05-24 10:08:49.328", "levelname": "DEBUG", "name": "views", "message": "Watcher manager with PUT"}
{"asctime": "2021-05-24 10:08:49.334", "levelname": "INFO", "name": "views", "message": "Updating Watcher 1 from SCHEDULED to STOP"}
```

Figure 5-20: ELMA_REST decommissioned VNF detection

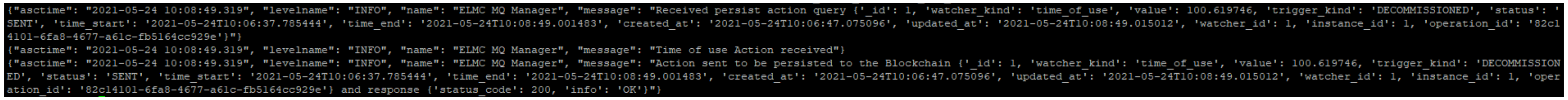
The ELMA_MQ received the watcher event that informs about the decommissioned VNF, and the value informs that the VNF has been used for 100,619746 seconds. This action is sent to the ELMC



```
{"asctime": "2021-05-24 10:08:49.315", "levelname": "INFO", "name": "elma_mq_manager", "message": "Watcher event received {'_id': 1, 'watcher_kind': 'time_of_use', 'value': 100.619746, 'trigger_kind': 'DECOMMISSIONED', 'status': 'SENT',
'time_start': '2021-05-24T10:06:37.785444', 'time_end': '2021-05-24T10:08:49.001483', 'created_at': '2021-05-24T10:06:47.075096', 'updated_at': '2021-05-24T10:08:49.015012', 'watcher_id': 1, 'instance_id': 1, 'operation_id': '82c14101-6f
a8-4677-a61c-fb5164cc929e'}"}
{"asctime": "2021-05-24 10:08:49.316", "levelname": "DEBUG", "name": "elma_mq_manager", "message": "Received time of use action {'_id': 1, 'watcher_kind': 'time_of_use', 'value': 100.619746, 'trigger_kind': 'DECOMMISSIONED', 'status': 'S
ENT', 'time_start': '2021-05-24T10:06:37.785444', 'time_end': '2021-05-24T10:08:49.001483', 'created_at': '2021-05-24T10:06:47.075096', 'updated_at': '2021-05-24T10:08:49.015012', 'watcher_id': 1, 'instance_id': 1, 'operation_id': '82c14
101-6fa8-4677-a61c-fb5164cc929e'}"}
{"asctime": "2021-05-24 10:08:49.316", "levelname": "INFO", "name": "elma_mq_manager", "message": "Time of use Action sent to ELMC"}
```

Figure 5-21: ELMA_MQ Watcher event received

Finally, the ELMC_MQ receives the action and informs the marketplace.



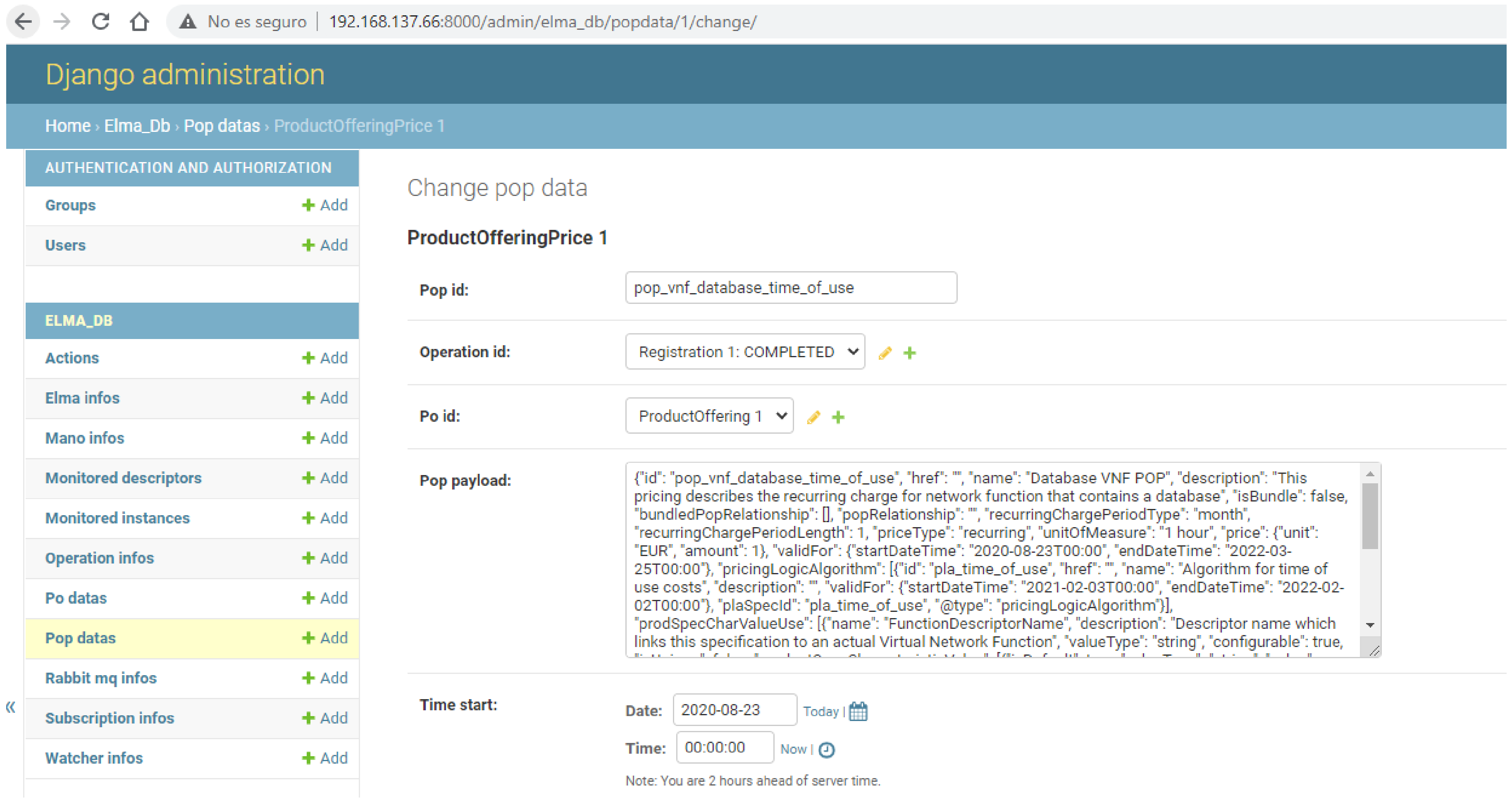
```
{"asctime": "2021-05-24 10:08:49.319", "levelname": "INFO", "name": "ELMC MQ Manager", "message": "Received persist action query {'_id': 1, 'watcher_kind': 'time_of_use', 'value': 100.619746, 'trigger_kind': 'DECOMMISSIONED', 'status': '
SENT', 'time_start': '2021-05-24T10:06:37.785444', 'time_end': '2021-05-24T10:08:49.001483', 'created_at': '2021-05-24T10:06:47.075096', 'updated_at': '2021-05-24T10:08:49.015012', 'watcher_id': 1, 'instance_id': 1, 'operation_id': '82c1
4101-6fa8-4677-a61c-fb5164cc929e'}"}
{"asctime": "2021-05-24 10:08:49.319", "levelname": "INFO", "name": "ELMC MQ Manager", "message": "Time of use Action received"}
{"asctime": "2021-05-24 10:08:49.319", "levelname": "INFO", "name": "ELMC MQ Manager", "message": "Action sent to be persisted to the Blockchain {'_id': 1, 'watcher_kind': 'time_of_use', 'value': 100.619746, 'trigger_kind': 'DECOMMISSION
ED', 'status': 'SENT', 'time_start': '2021-05-24T10:06:37.785444', 'time_end': '2021-05-24T10:08:49.001483', 'created_at': '2021-05-24T10:06:47.075096', 'updated_at': '2021-05-24T10:08:49.015012', 'watcher_id': 1, 'instance_id': 1, 'oper
ation_id': '82c14101-6fa8-4677-a61c-fb5164cc929e'} and response {'status_code': 200, 'info': 'OK'}"}
```

Figure 5-22: ELMC_MQ time of use action reception

(3) The third group of screenshots are related to functional test that have been described in Table 4-4 and are related to this integration effort.

a) Test Registration POP:

The POP registration has been demonstrated in Figure 5-15. Using the django administrator console, it is possible to access the data gathered in ELMA_REST database and verify the correct POP registration:


No es seguro | 192.168.137.66:8000/admin/elma_db/popdata/1/change/

Django administration

Home › Elma_Db › Pop datas › ProductOfferingPrice 1

AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add

ELMA_DB
Actions + Add
Elma infos + Add
Mano infos + Add
Monitored descriptors + Add
Monitored instances + Add
Operation infos + Add
Po datas + Add
Pop datas + Add
Rabbit mq infos + Add
Subscription infos + Add
Watcher infos + Add

Change pop data

ProductOfferingPrice 1

Pop id: pop_vnf_database_time_of_use

Operation id: Registration 1: COMPLETED

Po id: ProductOffering 1

Pop payload: {"id": "pop_vnf_database_time_of_use", "href": "", "name": "Database VNF POP", "description": "This pricing describes the recurring charge for network function that contains a database", "isBundle": false, "bundledPopRelationship": [], "popRelationship": "", "recurringChargePeriodType": "month", "recurringChargePeriodLength": 1, "priceType": "recurring", "unitOfMeasure": "1 hour", "price": {"unit": "EUR", "amount": 1}, "validFor": {"startDateTime": "2020-08-23T00:00", "endDateTime": "2022-03-25T00:00"}, "pricingLogicAlgorithm": [{"id": "pla_time_of_use", "href": "", "name": "Algorithm for time of use costs", "description": "", "validFor": {"startDateTime": "2021-02-03T00:00", "endDateTime": "2022-02-02T00:00"}, "plaSpecId": "pla_time_of_use", "@type": "pricingLogicAlgorithm"}], "prodSpecCharValueUse": [{"name": "FunctionDescriptorName", "description": "Descriptor name which links this specification to an actual Virtual Network Function", "valueType": "string", "configurable": true,

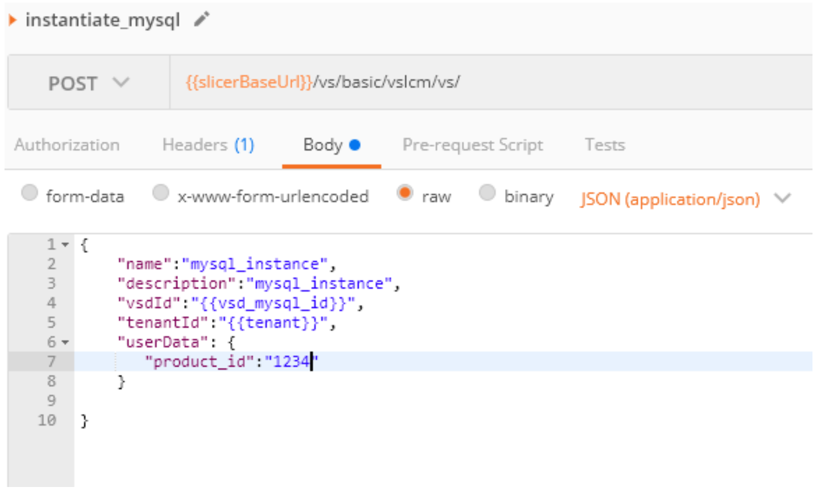
Time start:
Date: 2020-08-23 Today |
Time: 00:00:00 Now |
Note: You are 2 hours ahead of server time.

Figure 5-23: Django Administration Console shows POP registration

b) Test_registration_POP_error

In this case, we try to register a PO that is not signed in the mocked marketplace. We will use Postman to make the NSSO to create a request to the ELMC with a wrong product_id and verify that the instantiation of the service is not successful since there is not a license purchased related to this this product id.

The HTTP rest call is:



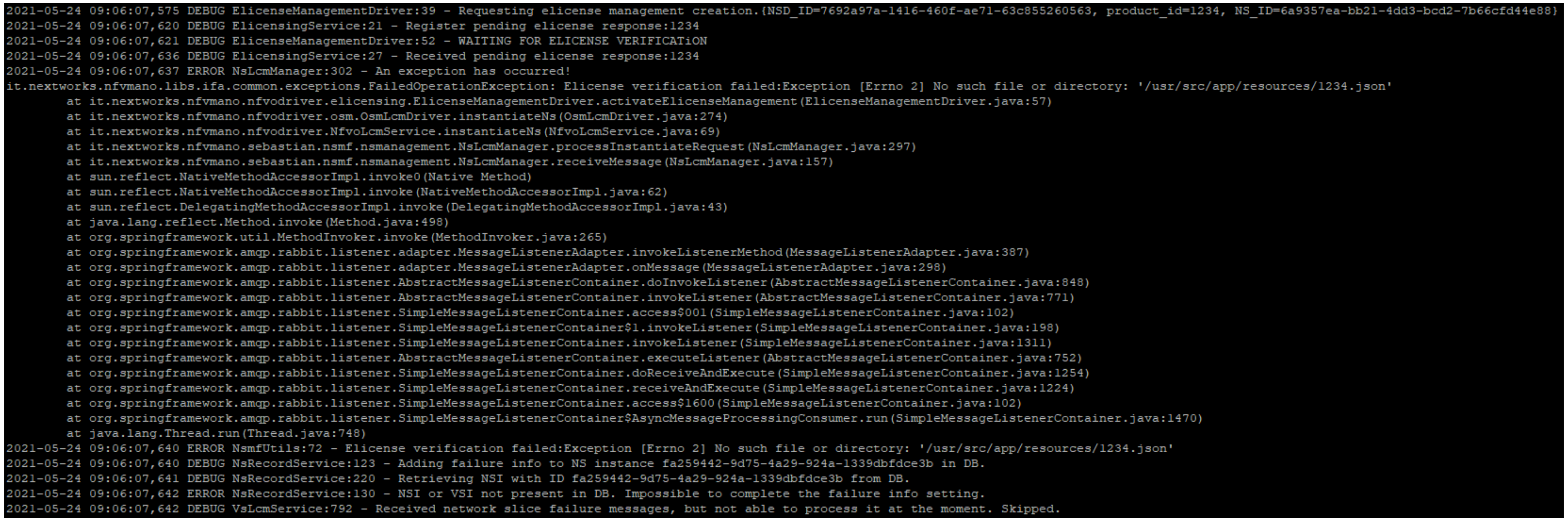
```
{{slicerBaseUrl}}/vs/basic/vslcm/vs/
```

```
{
    "name":"mysql_instance",
    "description":"mysql_instance",
    "vsdId":"{{vsd_mysql_id}}",
    "tenantId":"{{tenant}}",
    "userData": {
       "product_id":"1234"
    }

}
```

Figure 5-24: Request with wrong Product_ID

The resulting response in the NSSO is an error in the license verification, so as expected the NSI it is not instantiated:

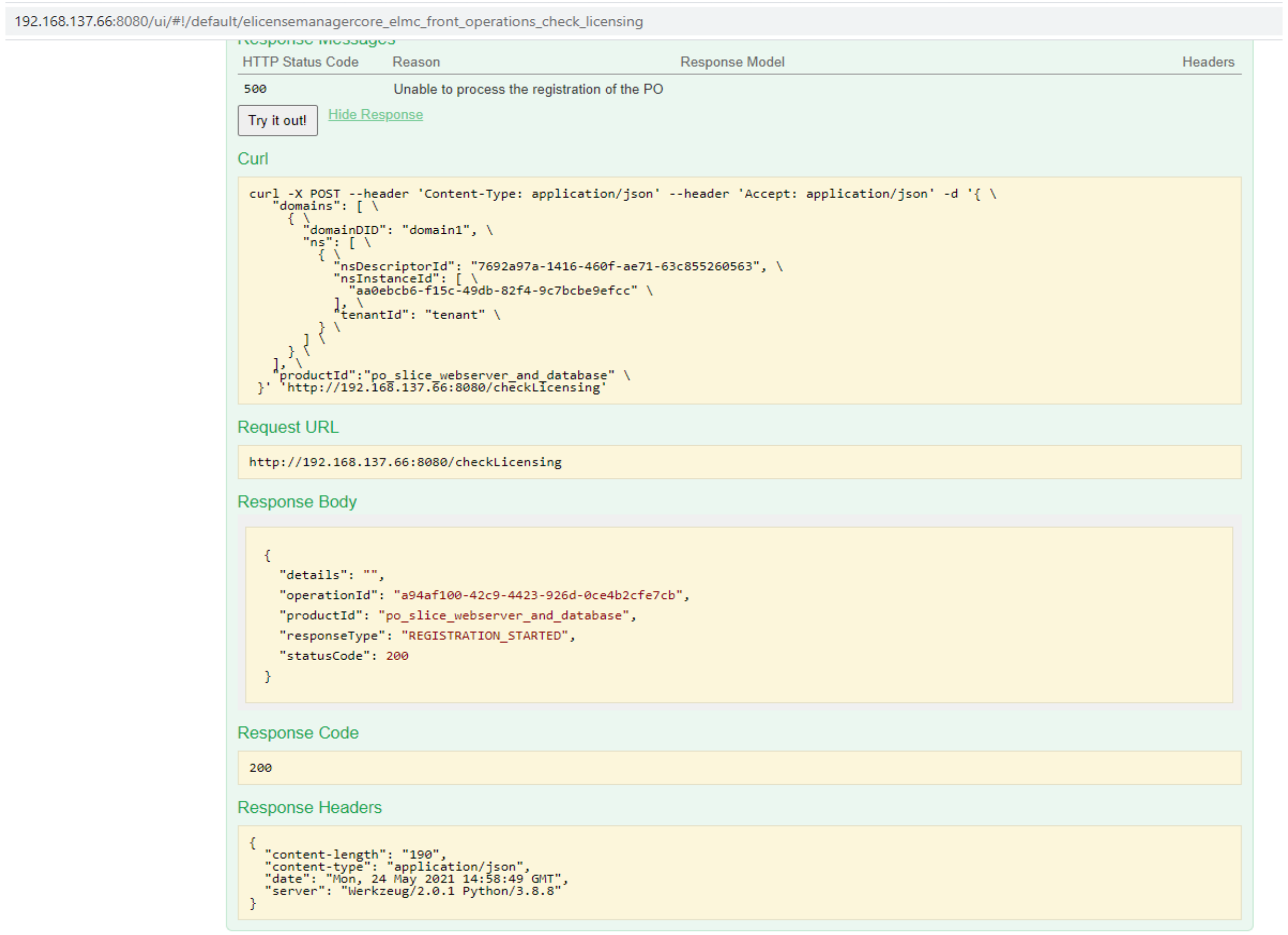


```
2021-05-24 09:06:07,575 DEBUG ElicenseManagementDriver:39 - Requesting elicense management creation.{NSD_ID=7692a97a-1416-460f-ae71-63c855260563, product_id=1234, NS_ID=6a9357ea-bb21-4dd3-bcd2-7b66cfd44e88}
2021-05-24 09:06:07,620 DEBUG ElicensingService:21 - Register pending elicense response:1234
2021-05-24 09:06:07,621 DEBUG ElicenseManagementDriver:52 - WAITING FOR ELICENSE VERIFICATiON
2021-05-24 09:06:07,636 DEBUG ElicensingService:27 - Received pending elicense response:1234
2021-05-24 09:06:07,637 ERROR NsLcmManager:302 - An exception has occurred!
it.nextworks.nfvmano.libs.ifa.common.exceptions.FailedOperationException: Elicense verification failed:Exception [Errno 2] No such file or directory: '/usr/src/app/resources/1234.json'
        at it.nextworks.nfvmano.nfvodriver.elicensing.ElicenseManagementDriver.activateElicenseManagement(ElicenseManagementDriver.java:57)
        at it.nextworks.nfvmano.nfvodriver.osm.OsmLcmDriver.instantiateNs(OsmLcmDriver.java:274)
        at it.nextworks.nfvmano.nfvodriver.NfvoLcmService.instantiateNs(NfvoLcmService.java:69)
        at it.nextworks.nfvmano.sebastian.nsmf.nsmanagement.NsLcmManager.processInstantiateRequest(NsLcmManager.java:297)
        at it.nextworks.nfvmano.sebastian.nsmf.nsmanagement.NsLcmManager.receiveMessage(NsLcmManager.java:157)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.springframework.util.MethodInvoker.invoke(MethodInvoker.java:265)
        at org.springframework.amqp.rabbit.listener.adapter.MessageListenerAdapter.invokeListenerMethod(MessageListenerAdapter.java:387)
        at org.springframework.amqp.rabbit.listener.adapter.MessageListenerAdapter.onMessage(MessageListenerAdapter.java:298)
        at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.doInvokeListener(AbstractMessageListenerContainer.java:848)
        at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.invokeListener(AbstractMessageListenerContainer.java:771)
        at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.access$001(SimpleMessageListenerContainer.java:102)
        at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer$1.invokeListener(SimpleMessageListenerContainer.java:198)
        at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.invokeListener(SimpleMessageListenerContainer.java:1311)
        at org.springframework.amqp.rabbit.listener.AbstractMessageListenerContainer.executeListener(AbstractMessageListenerContainer.java:752)
        at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.doReceiveAndExecute(SimpleMessageListenerContainer.java:1254)
        at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.receiveAndExecute(SimpleMessageListenerContainer.java:1224)
        at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer.access$1600(SimpleMessageListenerContainer.java:102)
        at org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer$AsyncMessageProcessingConsumer.run(SimpleMessageListenerContainer.java:1470)
        at java.lang.Thread.run(Thread.java:748)
2021-05-24 09:06:07,640 ERROR NsmfUtils:72 - Elicense verification failed:Exception [Errno 2] No such file or directory: '/usr/src/app/resources/1234.json'
2021-05-24 09:06:07,640 DEBUG NsRecordService:123 - Adding failure info to NS instance fa259442-9d75-4a29-924a-1339dbfdce3b in DB.
2021-05-24 09:06:07,641 DEBUG NsRecordService:220 - Retrieving NSI with ID fa259442-9d75-4a29-924a-1339dbfdce3b from DB.
2021-05-24 09:06:07,642 ERROR NsRecordService:130 - NSI or VSI not present in DB. Impossible to complete the failure info setting.
2021-05-24 09:06:07,642 DEBUG VsLcmService:792 - Received network slice failure messages, but not able to process it at the moment. Skipped.
```

Figure 5-25: Error in license verification shown at NSSO logs

c) Test NSSO interface

Before the integration effort we used our swagger API: http://ELMC_URL:8080/ui for the first functional tests regarding the interface between the NSSO and the eLMC. As showed in the swagger API, the response from the eLMC is notifying us that the PO registration process has started, so the interface is well defined:


192.168.137.66:8080/ui/#!/default/elicensemanagercore_elmc_front_operations_check_licensing

HTTP Status Code	Reason	Response Model	Headers
500	Unable to process the registration of the PO		

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
   "domains": [ \
     { \
       "domainDID": "domain1", \
       "ns": [ \
         { \
           "nsDescriptorId": "7692a97a-1416-460f-ae71-63c855260563", \
           "nsInstanceId": [ \
             "aa0ebcb6-f15c-49db-82f4-9c7bcbe9efcc" \
           ], \
           "tenantId": "tenant" \
         } \
       ] \
     } \
   ], \
   "productId":"po_slice_webserver_and_database" \
 }' 'http://192.168.137.66:8080/checkLicensing'
```

Request URL

```
http://192.168.137.66:8080/checkLicensing
```

Response Body

```
{
  "details": "",
  "operationId": "a94af100-42c9-4423-926d-0ce4b2cfe7cb",
  "productId": "po_slice_webserver_and_database",
  "responseType": "REGISTRATION_STARTED",
  "statusCode": 200
}
```

Response Code

```
200
```

Response Headers

```
{
  "content-length": "190",
  "content-type": "application/json",
  "date": "Mon, 24 May 2021 14:58:49 GMT",
  "server": "Werkzeug/2.0.1 Python/3.8.8"
}
```

Figure 5-26: Example of PO registration using eLMC API

5.4. Data-driven actuation

As discussed in previous sections, the provisioning of a network slice to an operator through 5GZORRO components, creates a Service Level Agreement (SLA) contract. The SLA contains the quantifiable metrics of the virtual resource that have been agreed upon, called Service Level Objectives (SLO). The creation of the smart contract initiates a workflow whose purpose is to trigger a data pipeline that predicts the future values of those SLOs throughout the lifecycle of the SLA, using Machine Learning algorithms. To that end, the following 5GZORRO components are involved:

- Monitoring Data Aggregator (MDA)
- Intelligent SLA Breach Prediction module (ISBP)
- Smart Contract Lifecycle Manager (SCLM)
- Data Lake

The workflow is visible in Figure 5-27.

Once the SLA is created, the MDA starts collecting monitoring data related to the network slice that has been provisioned, and sends them to the Storage Service located in the Data Lake of 5GZORRO. Concurrently, SCLM sends the SLA to a different message queue also located in the Data Lake. The ISBP, which is part of the Data Lake and awaits messages from these message queues, parses the message containing the SLA and creates a pipeline that generates future predictions of the SLO metric, based on the monitoring data of the slice. The predictions are generated by an AI model that has been trained on data similar to the ones it is predicting. The ISBP also includes functionality that enables the re-training of the AI model if the accuracy of its predictions falls under a given threshold. Finally, if the generated prediction is above the threshold dictated by the SLA, it is packaged in a notification message and pushed to a message queue. This notification can be then picked up by the Intelligent Slice and Service Manager component of 5GZORRO that can then take action to prevent contract breach.

The architecture of the components mentioned above can be seen in Figure 5-28. The MDA and the SCLM, are not part of the Data Lake and interface with the Storage Services and a message queue to push monitoring data and the SLA events respectively. Accordingly, the Data Lake maintains several message queues in order to implement a stream-based service. As can be seen, each queue stores different kind of data.

The ISBP is located within a Docker container and has connections to the message queues, but also exposes a REST API, implementing a limited number of functions. The Prediction and Training modules are ephemeral containers whose lifecycle depends only on the duration of the task they have to complete. The launching of these containers is triggered by events fired by the creation of the files containing the data for the prediction and model training respectively. Upon completion, the Prediction module sends the generated prediction to the ISBP using a simple HTTP request, whereas the Training module stores the new model to the Data Lake storage.

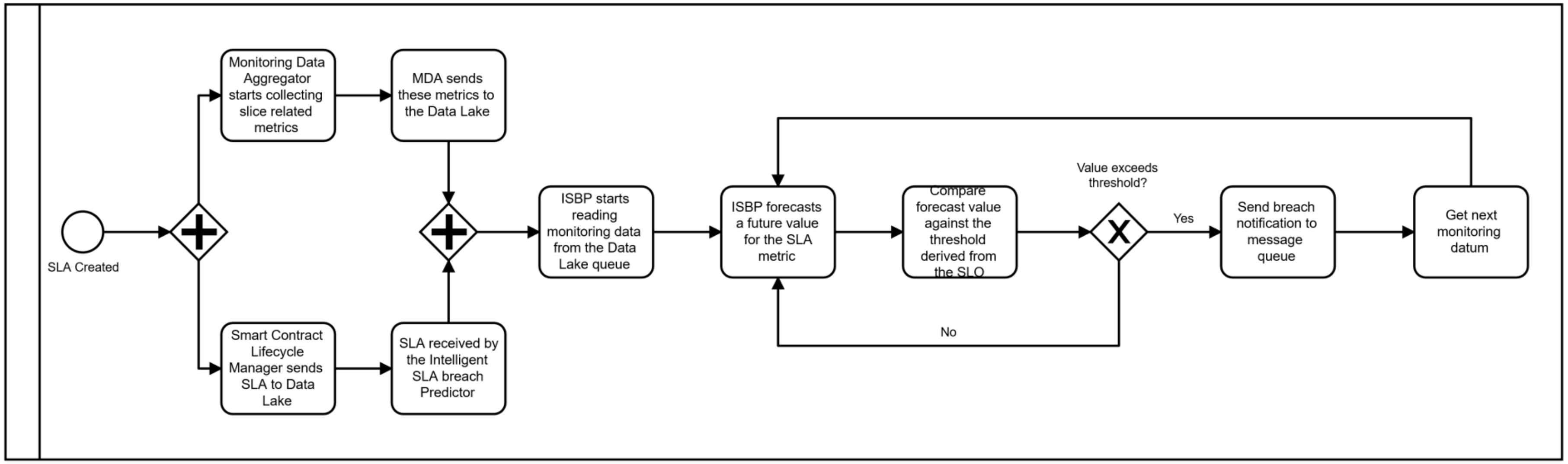

SLA Created
Monitoring Data Aggregator starts collecting slice related metrics
MDA sends these metrics to the Data Lake
Smart Contract Lifecycle Manager sends SLA to Data Lake
SLA received by the Intelligent SLA breach Predictor
ISBP starts reading monitoring data from the Data Lake queue
ISBP forecasts a future value for the SLA metric
Compare forecast value against the threshold derived from the SLO
Value exceeds threshold?
Yes
No
Send breach notification to message queue
Get next monitoring datum

Figure 5-27: SLA Breach prediction workflow

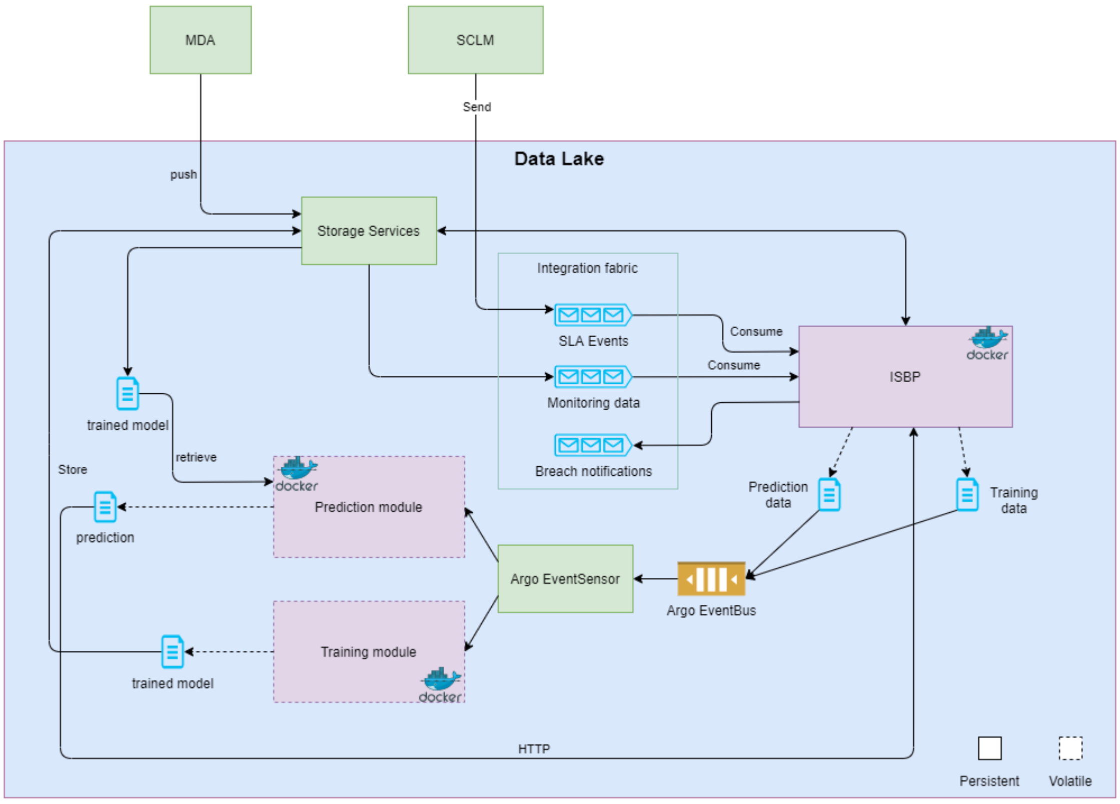


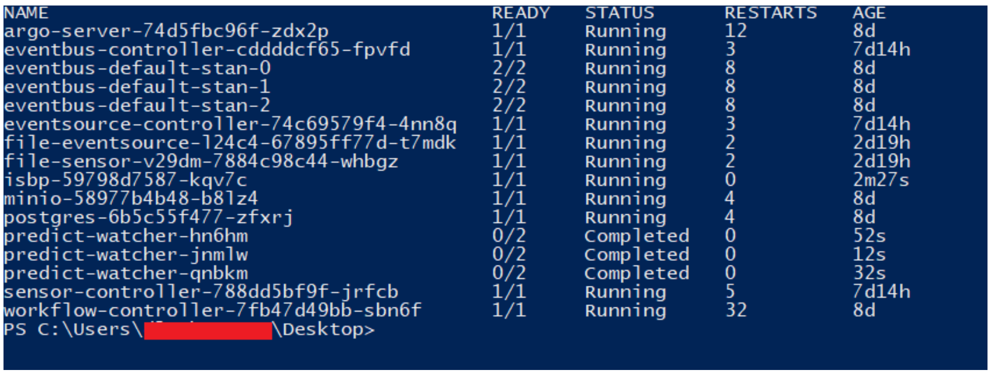
Figure 5-28: ISBP Architecture

5.4.1. Integration tests and results

The components ISBP and Prediction/Training modules have been developed and tested in local ICOM premises with a local installation of Apache Kafka as the message queue ISBP connects to. The tests involve an existing dataset consisting of metrics acquired from one of ICOM's servers. These metrics include the server bandwidth, cache hits and request per minute. The data is pushed to Kafka which simulates the workflow discussed above. The results with the ephemeral containers mentioned previously can be seen in Figure 5-29.

The integration of these components with the rest of the components of the Data Lake is underway, with the main focus being:

- The deployment of these components to the infrastructure of the Data Lake.
- The connectivity of ISBP with the message queues.
- The connectivity of the components related to other aspects of the Storage Services of the Data Lake.



```
NAME                                         READY   STATUS      RESTARTS   AGE
argo-server-74d5fbc96f-zdx2p                 1/1     Running     12         8d
eventbus-controller-cddddcf65-fpvfd          1/1     Running     3          7d14h
eventbus-default-stan-0                      2/2     Running     8          8d
eventbus-default-stan-1                      2/2     Running     8          8d
eventbus-default-stan-2                      2/2     Running     8          8d
eventsource-controller-74c69579f4-4nn8q      1/1     Running     3          7d14h
file-eventsource-124c4-67895ff77d-t7mdk      1/1     Running     2          2d19h
file-sensor-v29dm-7884c98c44-whbgz           1/1     Running     2          2d19h
isbp-59798d7587-kqv7c                        1/1     Running     0          2m27s
minio-58977b4b48-b81z4                       1/1     Running     4          8d
postgres-6b5c55f477-zfxrj                    1/1     Running     4          8d
predict-watcher-hn6hm                        0/2     Completed   0          52s
predict-watcher-jnmlw                        0/2     Completed   0          12s
predict-watcher-qnbkm                        0/2     Completed   0          32s
sensor-controller-788dd5bf9f-jrfcb           1/1     Running     5          7d14h
workflow-controller-7fb47d49bb-sbn6f         1/1     Running     32         8d
PS C:\Users\[redacted]\Desktop>
```

Figure 5-29: ISBP containers running at ICOM premises

6. Conclusions

This deliverable reports the status of the implementation of the set of modules building the 5GZORRO platform that are being implemented in the context of WP4. In this sense, the document follows the same structure of D4.1 and provides an overview of three of main features of the 5GZORRO Platform the WP4 aims to realise: Security and Trust Orchestration, Intelligent and Automated Slice and Service Management and MANO and Slicing Enhancements. Per each of such feature is discussed the set of modules it encompasses and per each module are reported any deviation from the original design (see D4.1), the current implementation status and a list of the main functional tests performed.

The second part of the document discusses the effort in the integration of modules presented and the strategy adopted by 5GZORRO to perform such integration. Several Platform Development Teams have been created, with the aim of going beyond the limitation imposed by the classical WP-based division of the effort, thus facilitating the development of the components by following an integration-driven approach. In this sense, the integration of the components, as result of the activities of Teams 3 (Zero-touch slicing), 4 (e-Licensing Management) and 6 (SLA-Datalake) are reported and detailed, in terms of storylines and integration tests, which also include the interaction with other 5GZORRO modules not discussed in this document whose implementation will be reported in D3.2.

The modules and the integration work reported in this document does not represent a stable implementation but rather a snapshot of the current status of some parts of the 5GZORRO platform, which will be continuously improved and refined in the following months and reported in the deliverable D4.3.

The list of KPIs and object covered by this deliverable is reported in Table 6-1.

Table 6-1: D4.2 contribution to 5GZORRO objectives and KPIs.

OBJECTIVE	Target KPIs	Applicable Prototype Artifact
OBJ-2. Design and prototype a security and trust framework, integrated with 5G service management platforms, to demonstrate Zero-Day trust establishment in distributed multi-stakeholder environments and automated security management to ensure trusted and secure execution of offloaded workloads across domains in 5G networks	• *Provide mechanisms for zero touch trust automation in multi-domain scenarios on top of a 5G service management framework (KPI target: to cover up to 4 different stakeholders as part of the automated trust establishment process and to enable its automatic renegotiation when a stakeholder is joining or leaving the trust link).*	See Sec. 2 for Security and Trust Orchestration
	• *Enhance a 5G service management framework enabling the detection of security vulnerabilities and compromises and the provision of a set of potential countermeasures to mitigate them using a zero-touch approach (KPI target: identifying 6 different types of common attacks to software infrastructures and provide a complete set of countermeasures -filter traffic, divert it to a honeynet, send an alert to the system admin, etc.- for each of them).*	See Sec. 2.3 Intra-domain Security enablers
	• *Support the integration of zero trust hardware platforms (TEE - Trusted Execution Environments) as a root of trust for the monitoring of information and the establishment of end-to-end secure communications enabling critical workloads to go across different tenants and different stakeholders (KPI target: research on the integration evolution of three TEE platforms --one provided by a project partner-- and two other commercial ones to support a fast and secure establishment of end-to-end cross-slice communications for critical workloads).*	See Sec. 2.2 Trusted Execution Environment Security Management
OBJ-5. Define and prototype a secure shared spectrum market to enable real-time trading of spectrum allocations between parties that do not have a pre-established trust relationship.	• *Agnostic support of various radio technologies, to ensure that the market will work regardless of the considered radio technology (KPI target: 5GNR, LTE and WiFi will be supported).*	See Sec. 4.2 Network Slice and Service Orchestrator
OBJ-6. Realize a cloud-friendly network software licensing framework for location	• *Enable the creation of license agreement templates associated to VNF/NS instances (KPI target: create templates attached to eContract detailing name, context, license conditions, negotiation goal and constraints).*	See Sec. 4.3 E-Licensing Manager .

OBJECTIVE	Target KPIs	Applicable Prototype Artifact
independent network appliances execution.	• *Generate vendor independent license token to manage location independent VNFs from 3rd party edge to core datacenter (KPI target: license service creates generic tokens to latter run any vendor VNF across at least 2 network segments).*	See Sec. 4.3 E-Licensing Manager .
	• *Instantiate Network Services with VNFs from diverse providers (KPI target: use eContract to include VNF licensed by at least 3 different providers).*	See, Sec. 3.1, ISSM Workflow Manager (ISSM-WFM, Sec. 4.2 Network Slice and Service Orchestrator , Sec. 4.3 E-Licensing Manager
OBJ-8. Ensure the long-term success of the project through standardization and dissemination in scientific, industrial, and commercial fora, and by contributing to relevant open-source communities & SDOs also exploring synergies with other EU initiatives and projects.	*No specific target to be covered by architecture design*	5GZORRO Platform prototypes are available on GitHub under the 5GZORRO Project space [42]

References

[1] 5GZORRO Consortium, Deliverable D4.1 - Intermediate prototype of Zero Touch Service Mgmt with Security and Trust.

[2] Xiong, L., & Liu, L. (2004) - Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. IEEE transactions on Knowledge and Data Engineering, 16(7), 843-857.

[3] Haga, S., Esmaeily, A., Kralevska, K., & Gligoroski, D. (2020, November) - 5G Network Slice Isolation with WireGuard and Open Source MANO: A VPNaaS Proof-of-Concept. In 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) (pp. 181-187). IEEE.

[4] Bollapragada, V., Khalid, M., & Wainner, S. (2005) - IPSec VPN Design. Cisco Press.

[5] Feilner, M. (2006) - OpenVPN: Building and integrating virtual private networks. Packt Publishing Ltd.

[6] Product Catalog Management API REST Specification, TM Forum Specification, TMF620, Release 19.0.0, July 2019.

[7] 5GZORRO Consortium, Deliverable D3.1 – Design of the evolved 5G Service layer solutions, Jan 2021

[8] SCONE – A secure container environment - https://scontain.com/index.html?lang=en

[9] Intel Software Guard Extensions –https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions.html

[10] Azure Cloud – SGX powered Servers - https://azure.microsoft.com/en-us/blog/dcsv2series-vm-now-generally-available-from-azure-confidential-computing/ Accessed 27 January 2021.

[11] Zeek network security monitor - https://zeek.org/

[12] Elasticsearch search and analytics engine - https://www.elastic.co/elasticsearch/

[13] Kibana - https://www.elastic.co/kibana

[14] Filebeat log shipper - https://www.elastic.co/beats/filebeat.

[15] Flask - https://flask.palletsprojects.com/en/1.1.x

[16] Gevent Python networking library - http://www.gevent.org/

[17] Werkzeug WSGI server - https://werkzeug.palletsprojects.com/en/1.0.x/

[18] 5GZORRO Consortium, Deliverable D2.3 – Update Design of the 5GZORRO Platform for Security & Trust, April 2021

[19] ETSI GS NFV-SOL 006 V3.3.1 (2020-08): Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on YANG Specification

[20] Resource Catalog Management API REST Specification" (2020-11), TM Forum Specification, TMF634, Release 17.0.1, December 2017

[21] Service Catalog Management API REST Specification, TM Forum Specification, TMF633, Release 18.5.0, January 2019.

[22] Spring Boot application framework - https://spring.io/projects/spring-boot

[23] Nextworks 5G Catalogue - https://github.com/nextworks-it/5g-catalogue

[24] Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on TOSCA specification https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/001/03.03.01_60/gs_NFV-SOL001v030301p.pdf

[25] PostgreSQL - https://www.postgresql.org/

[26] J-OSM - A Java-based client for Open Source MANO (OSM) - https://github.com/girtel/J-OSMClient

[27] OpenStack4j - A fluent OpenStack client - https://github.com/ContainX/openstack4j

[28] FastAPI - https://fastapi.tiangolo.com

[29] Python client for Apche Kafka - https://kafka-python.readthedocs.io/en/master/index.html

[30] Psycopg – PostgreSQL database adapter for Python - https://www.psycopg.org/docs/

[31] Python-RSA - https://stuvel.eu/python-rsa-doc/

[32] Timeloop - https://github.com/sankalpjonn/timeloop

[33] SQLAlchemy - Python SQL toolkit and Object Relational Mapper - https://www.sqlalchemy.org/

[34] 5GCity – A distributed cloud & radio platform for 5G Neutral Hosts - https://www.5gcity.eu/

[35] Network Slice Blueprint Definition - https://www.ngmn.org/wp-content/uploads/160113_NGMN_Network_Slicing_v1_0.pdf, page 6

[36] NXW slicer, 5GZORRO-core-1.0-alfa release, https://github.com/nextworks-it/slicer/tree/5gzorro-core-1.0-alfa

[37] NXW slicer-catalogue, 5GZORRO-core-1.0-alfa release, https://github.com/nextworks-it/slicer-catalogue/tree/5gzorro-core-1.0-alfa

[38] NXW nfvo-drivers, 5GZORRO-core-1.0-alfa release, https://github.com/nextworks-it/nfvo-drivers/tree/5gzorro-core-1.0-alfa

[39] NXW slicer-catalogue, 5GZORRO-core-1.0-alfa release, https://github.com/nextworks-it/slicer-catalogue/tree/5gzorro-core-1.0-alfa

[40] MDA OpenAPI specification, https://github.com/5GZORRO/mda/blob/main/doc/openapi.json

[41] E-Licensing core OpenAPI specification, https://github.com/5GZORRO/elicensing-manager-core/blob/master/elicensemanagercore/elmc_front/swagger.yaml

[42] 5GZORRO Consortium, Deliverable D5.1 – Use case validation plan and testbed design

[43] 5GZORRO GitHub space - https://github.com/5GZORRO

7. Abbreviations and Definitions

7.1. Abbreviations

API	Application Programming Interface
CLI	Command Line Interface
DID	Distributed Identifier
GST	Generic Slice Template
MANO	Management and Orchestration
MNO	Mobile Network Operator
NBI	North Bound Interface
NEST	Network Slice Type
NFVO	Networks Function Virtualization Orchestrator
NSI	Network Service Instance
OSM	Open Source MANO
PLMN	Public Land Mobile Network
RAN	Radio Access Network
REST	Representational State Transfer
SBI	South Bound Interface
SSID	Service Set Identifier
TAP	Terminal Access Point
VIM	Virtual Infrastructure Manager
VNF	Virtual Network Function
VPN	Virtual Private Network
WP	Work Package
WSGI	Web Server Gateway Interface

8. Appendix I – Trust Management Framework

8.1. Trust Management Framework Information Model

The redesigned information model of the Trust Management Framework is presented in Figure 8-1. In particular, it presents the set of characteristics that are interpreted by the PeerTrust reputation model to generate a final score for each trust computation request.

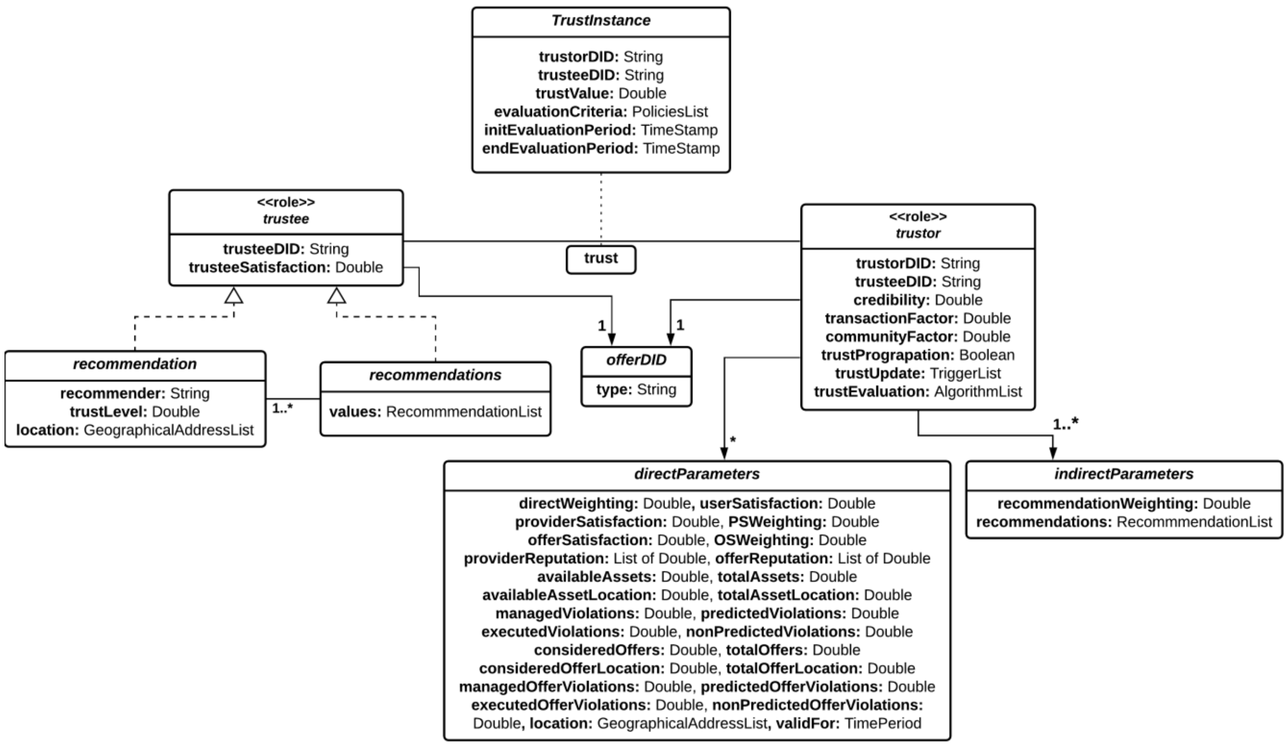

TrustInstance
trustorDID: String
trusteeDID: String
trustValue: Double
evaluationCriteria: PoliciesList
initEvaluationPeriod: TimeStamp
endEvaluationPeriod: TimeStamp
<<role>>
trustee
trusteeDID: String
trusteeSatisfaction: Double
trust
<<role>>
trustor
trustorDID: String
trusteeDID: String
credibility: Double
transactionFactor: Double
communityFactor: Double
trustPrograpation: Boolean
trustUpdate: TriggerList
trustEvaluation: AlgorithmList
1
1
recommendation
recommender: String
trustLevel: Double
location: GeographicalAddressList
1..*
recommendations
values: RecommmendationList
offerDID
type: String
*
1..*
directParameters
directWeighting: Double, userSatisfaction: Double
providerSatisfaction: Double, PSWeighting: Double
offerSatisfaction: Double, OSWeighting: Double
providerReputation: List of Double, offerReputation: List of Double
availableAssets: Double, totalAssets: Double
availableAssetLocation: Double, totalAssetLocation: Double
managedViolations: Double, predictedViolations: Double
executedViolations: Double, nonPredictedViolations: Double
consideredOffers: Double, totalOffers: Double
consideredOfferLocation: Double, totalOfferLocation: Double
managedOfferViolations: Double, predictedOfferViolations: Double
executedOfferViolations: Double, nonPredictedOfferViolations: Double, location: GeographicalAddressList, validFor: TimePeriod
indirectParameters
recommendationWeighting: Double
recommendations: RecommmendationList

Figure 8-1 : UML diagram of Trust Management Framework

Table 8-1: Trust Management Framework Instance Information Model

Parameter	Type	Description
trustorDID	String	Unique identifier for a resource or service consumer.
trusteeDID	String	Unique identifier for a resource or service provider.
trustValue	Double	Current trust value assigned.
evaluationCriteria	List of intra or inter-domain policies	Criterion selected by trust model to assign the values to the trustee.
initEvaluationPeriod	TimeStamp	The time when trust value was generated.
endEvaluationPeriod	TimeStamp	The time when trust value will be over and has to be reassigned, if required.

Table 8-2: Trustee Entity Information Model

Parameter	Type	Description
trusteeDID	String	Unique identifier for a resource or service provider.
offerDID	String	Unique identifier for a particular product offer of the provider.
type	String	Kind of offer (RAN, spectrum, VNF/CNF, slice, or edge)
trusteeSatisfaction	Double	Truestee's satisfaction after x interactions with other providers.
recommendation	List of objects	A recommendation about a third-party.
recommender	String	Unique identifier for a recommender.
trustLevel	Double	Final trust score ranged from 0.0 to 1.0.
location	List of GeographicalAddress objects	It constitutes a group of GeographicalAddress.
recommendations	List of recommendations	Set of recommendations about a third entity from one or more external entities (recommenders).

Table 8-3: Trustor Entity Information Model

Parameter	Type	Description
trustorDID	String	Unique identifier for a resource or service consumer.
trusteeDID	String	Unique identifier for a resource or service provider.
offerDID	String	Unique identifier for a particular product offer.
type	String	Kind of offer (RAN, spectrum, VNF/CNF, slice, or Edge)
directParameters	List of key-value features	Dictionary with direct trust data to calculate trust level.
directWeighting	Double	Direct weighting parameter.
userSatisfaction	Double	Internal assessment of the service or resource provided by a stakeholder (trustor).
providerSatisfaction (PS)	Double	Trustor satisfaction in a third-party provider (trustee).
PSWeighting	Double	Weighting factor $\in$ [0,1], PS + OS = 1
offerSatisfaction (OS)	Double	Trustor satisfaction in a particular kind of offer of a third-party provider.
OSWeighting	Double	Weighting factor $\in$ [0,1], PS + OS = 1
providerReputation	List (double)	Set of previous trust evaluations about a provider.
offerReputation	List (double)	Set of previous trust evaluations about a specific kind of offer of a provider.
availableAssets	Integer	The available assets (services and resources) of the trusteeDID when the trustor is determining the reputation.
totalAssets	Integer	The total assets of the trusteeDID when the trustor is determining the reputation (active and inactive).
availableAssetLocation	Integer	The available assets of the trusteeDID, at a specific location, when the trustor is determining the reputation.
totalAssetLocation	Integer	The total assets of the trusteeDID, at a specific location, when the trustor is determining the reputation (active and inactive).
managedViolations	Integer	The total number of predicted SLA violations that were finally managed successful, associated with the trusteeDID.
predictedViolations	Integer	The total number of predicted SLA violations associated with the trusteeDID.

Parameter	Type	Description
executedViolations	Integer	The total number of predicted SLA violations that were finally managed unsuccessful (violation), associated with the trusteeDID.
nonPredictedViolations	Integer	The number of SLA violations that were not predicted and turned out to be SLA violations, associated with the trusteeDID.
consideredOffers	Integer	The number of offers considered by the Smart Resource and Service Discovery (SRSD), for a particular type of offer, from the trusteeDID when trustor is determining the reputation.
totalOffers	Integer	The available number of a particular offer type from the trusteeDID when trustor is determining the reputation.
consideredOfferLocation	Integer	The number of offers considered by the Smart Resource and Service Discovery (SRSD) for a particular type of offer from the trusteeDID, at a specific location, when trustor is determining the reputation.
totalOfferLocation	Integer	The available number of a particular offer type from the trusteeDID, at a specific location, when trustor is determining the reputation.
managedOfferViolations	Integer	The total offer number (for a particular kind of offer) of predicted SLA violations that were finally managed successful, associated with the trusteeDID.
predictedOfferViolations	Integer	The total offer number (for a particular kind of offer) of predicted SLA violations associated with the trusteeDID.
executedOfferViolations	Integer	The total offer number (for a particular kind of offer) of predicted SLA violations that were finally managed unsuccessful (violation), associated with the trusteeDID.
nonPredictedOfferViolations	Integer	The total offer number (for a particular kind of offer) of SLA violations that were not predicted and turned out to be SLA violations, associated with the trusteeDID.
interactionNumber	Integer	Number of interactions carried out by the trusteeDID with the other domains.
feedbackNumber	Integer	Number of feedbacks made by other providers about the trusteeDID.
feedbackOfferNumber	Integer	Number of feedbacks made by other providers about a particular kind of trusteeDID's offer.
location	List of GeographicalAddress objects	It constitutes a group of GeographicalAddress.
validFor	TimePeriod	The period for which this resource or service is valid.
indirectParameters	List of key-value features	Dictionary with indirect trust data to calculate trust level.
recommendationWeighting	Double	Recommender's weighting parameter(s).
recommendations	List of recommendations	Set of recommendation about a third entity from one or more external entities.
credibility	Double	Factor that determines how accurate the recommendations are.
transactionFactor	Double	The context factor adapted on the current transaction of the trustee (required by PeerTrust model).
communityFactor	Double	The context factor adapted on the community of stakeholders to which the trustee belongs (required by PeerTrust model).

Parameter	Type	Description
trustPropagation	Boolean	Intra or inter-domain trust score and parameterTuple propagation (0 means intra, 1 means inter).
trustUpdate	List of objects	It indicates the triggers to recompute trust score.
trustEvaluation	List of objects	It identifies different evaluation algorithms such as PeerTrust reputation model.

8.2. Trust Management Framework Equations

8.2.1. General PeerTrust equation

The PeerTrust model is composed of a main equation that allows representing how a *domain v* can evaluate the trust score of a *domain u*. In particular, the principal equation is as follow:

$$T(u) = \alpha * \left(\frac{\sum_{i=1}^{I(u)} S(u,i) \; * \; Cr\big(p(u,i)\big) * \; TF(u,i)}{I(u)}\right) + \beta \; * \; CF(u)$$

where

- α , β depict weighting factors to be considered by the *domain v*. $\alpha, \beta \in [0,1]$ and $\alpha + \beta = 1$. It is advisable that α has a higher value than β.
- $I(u)$ is the total interaction number of *domain u* with the rest of domains.
- $p(u,i)$ denotes the rest of domains participating in the i-th interaction with the *domain u*.
- $S(u,i)$ represents the normalized value of *satisfaction* that the *domain u* obtains from $p(u,i)$ in its i-th interaction.
- $Cr(v)$ references the *domain v*'s credibility has in the *domain u*'s opinion.
- $TF(u,i)$ is the context *factor* adapted on the i-th *transaction* of *domain u*.
- $CF(u)$ indicates the context *factor* adapted on the *community* of entities to which the *domain u* belongs.

8.2.1.1. Satisfaction equation

Once the principal equation has been introduced (see section above), each of the parts that make up the main equation will be split into sub-equations to explain in more detail how each of the parameters is evaluated. In particular, next equation defines the *domain u*'s satisfaction on a product offer published by a particular domain in the i-th interaction.

$$S(u,i) \; = \gamma \; * \; PS(u,i) \; + \varphi \; * \; OS(u,i)$$

where

- γ, φ depict weighting factors to be considered by the *domain u*. γ, φ ∈ [0,1] and $\gamma + \varphi$ = 1.
- $PS(u,i)$ is the *satisfaction* that the *domain u* has on the i-th *domain* (*provider*).
- $OS(u,i)$ is the *satisfaction* that the *domain u* has on the i-th *domain*'s *offer*.
- It should be pointed out that $PS(u,i) + OS(u,i) \; = \; 1$.

The provider's satisfaction of the *domain u* on the *i-th* interaction will be computed about the *domain j* stakeholder.

$$PS(u,j) \; = \; Rep(u,j) \; * \; \bigoplus_{x=1}^{n} Rec(x,j) \;\; * \; T^{(t-1)}(u,x)$$

where

- $\oplus$ is a aggregation operation such as Minimum value, Maximum value, Arithmetic mean, or Harmonic mean. n denotes the rest of domains participating in the x-th interaction with the *domain j*.
- $Rec(x, j)$ is the recommendation of the x-th domain with which the *domain j* has a trust relationship. In other words, $T(x, j)$.
- $T^{(t-1)}(u, x)$ is the last trust score that the *domain u* has on the *domain x*.

The $Rep(u, j)$ is the average reputation that the *domain u* has on the *provider j* based on all assets (service and resources). This reputation contemplates features such as available assets, assets in a particular location, and multiple time windows to compute these features along the *provider j* lifecycle.

$$Rep(u,j) = \sum_{k=1}^{n} \varepsilon(k) * \frac{\left(\frac{AA(j)}{IA(j)} + \frac{AAL(j)}{IAL(j)} + 2 * \frac{MV(j)}{PV(j)} - 2 * \frac{EV(j)+NPV(j)}{PV(j)}\right) + 2}{6}$$

where

- $\sum_{k=1}^{n}$ represents the n time windows established by the *domain u*.
- $\varepsilon(k)$ depicts weighting factor to be considered by the *domain u* for each the time window k. $\varepsilon(k) \in [0,1]$ and $\varepsilon_1 + \cdots + \varepsilon_n = 1$.
- $AA(j)$ means the *available assets* of *provider j* when the *domain u* determined the reputation on *provider j*.
- $IA(j)$ depicts the *total assets* of the *provider j* when the *domain u* determined the reputation on *provider j*.
- $AAL(j)$ means the *available assets* of the *provider j,* at a particular *location,* when the *domain u* determined the reputation on *provider j*.
- $IAL(j)$ depicts the *total assets* of the *provider j,* at a particular *location,* when the *domain u* determined the reputation on *provider j*.
- $MV(j)$ represents the total number of predicted SLA *violations* that were finally *managed* successful.
- $PV(j)$ represents the total number of *predicted* SLA *violations*.
- $EV(j)$ represents the total number of predicted SLA *violations* that were finally managed unsuccessful (*executed*).
- $NPV(j)$ represents the number of SLA *violations* that were *not predicted* and turned out to be SLA violations.

The provider's satisfaction of the *domain u* on the *i-th* interaction will be computed about a particular offer of the *domain j* stakeholder.

$$OS(u, o_j) = Rep(u, o_j) * \bigoplus_{x=1}^{n} Rec(x, o_j) * T^{(t-1)}(u, x)$$

where

- $\oplus$ is a aggregation operation such as Minimum value, Maximum value, Arithmetic mean, or Harmonic mean. n denotes the rest of domains participating in the x-th interaction with the *domain j*.
- $Rec(x, o_j)$ is the recommendation of the x-th domain with which the *domain j* has a trust relationship about a specific kind of offer (RAN, Spectrum, Edge, Slice, or VNF/CNF). In other words, $T(x, o_j)$.
- $T^{(t-1)}(u, x)$ is the last trust score that the *domain u* has on a kind of specific offer of the the *domain x*.

The $Rep(u, o_j)$ is the average reputation that the *domain u* has on a kind of specific offer of the *provider j*. This reputation contemplates features such as available offers, offer in a particular location, and multiple time windows to compute these features along the *provider j's* offer lifecycle.

$$Rep(u, o_j) = \sum_{k=1}^{n} \varepsilon(k) * \frac{\left(\frac{CO(j)}{IO(j)} + \frac{COL(j)}{IOL(j)} + 2 * \frac{MOV(j)}{POV(j)} - 2 * \frac{EOV(j)+NPOV(j)}{POV(j)}\right) + 2}{6}$$

where

- $\sum_{k=1}^{n}$ represents the n time windows established by the *domain u*.
- $\varepsilon(k)$ depicts weighting factor to be considered by the *domain u* for each the time window k. $\varepsilon(k) \in [0,1]$ and $\varepsilon_1 + \cdots + \varepsilon_n = 1$.
- $CO(j)$ means the number of *offers considered* by the Smart Resource and Service Discovery (SRSD), for a particular type of offer, from the *provider j* when the *domain u* determined the reputation on *provider j*.
- $IO(j)$ depicts the available number of a particular *offer* type from the *provider j* when the *domain u* determined the reputation on *provider j*.
- $COL(j)$ means the number of *offers considered* by the Smart Resource and Service Discovery (SRSD), for a particular type of offer from *provider j,* at a particular *location,* when the *domain u* determined the reputation on *provider j*.
- $IOL(j)$ depicts the *available number* of a particular *offer* type from the *provider j,* at a particular *location,* when the *domain u* determined the reputation on *provider j*.
- $MOV(j)$ represents the total *offer* number (for a particular kind of offer) of predicted SLA *violations* that were finally *managed* successful.
- $POV(j)$ represents the total offer number (for a particular kind of offer) of *predicted* SLA *violations*.
- $EOV(j)$ represents the total offer number (for a particular kind of offer) of predicted SLA *violations* that were finally managed unsuccessful (*executed*).
- $NPOV(j)$ represents the offer number of SLA *violations* that were *not predicted* and turned out to be SLA violations.

It is worth mentioning that $IA, IAL, PV, IO, IOL, and\ POV$ are parameters whose minimum value is 1. In other case, if these parameters were initialized to 0, such equation parts should be omitted due to the fact that the division of 0 by 0 is not allowed.

8.2.2. Feedback Credibility equation

In this first iteration, we have contemplated using a general credibility metric that can be applied to multiple contexts. Specifically, the personalized similarity metric (PSM) is the one selected. The objective of this formula is to determine how similar *v and w domains* are when evaluating the same *domain u*.

$$Cr(p(u,i)) = \frac{Sim(\ p(u,i), w)}{\sum_{j=1}^{I(u)} Sim(p(u,j), w)} \equiv \frac{Sim(v,w)}{\sum_{j=1}^{I(u)} Sim(v,w)}$$

$$Sim(v,w) = 1 - \sqrt{\frac{\sum_{x \in IJS(v,w)} \left(\frac{\sum_{i=1}^{I(x,v)} S(x,i)}{I(x,v)} - \frac{\sum_{i=1}^{I(x,w)} S(x,i)}{I(x,w)}\right)^2}{|\ IJS(v,w)\ |}}$$

where

- $I(x, v)$ depicts the total number of interactions that have been carried out by the *domain x* with the *domain v*.

- $I(x, w)$ depicts the total number of interactions that have been carried out by the *domain x* with the *domain w*.
- | $IJS(v, w)$ | is the set of domains that are interacted both with *domain v* and *domain w*.

8.2.3. Transaction Context Factor equation

The purpose of this equation is to calculate a final value associated with the current transaction type (product offer and provider) from the number of feedbacks provided in different time windows established. A higher number of feedbacks in the different time windows will indicate that both the type of offer and the provider are currently being used by other domains, and therefore, there will be a higher number of recommenders to be contemplated for finally determining a stable reputation.

$$TF(u,i) = \frac{\sum_{j=1}^{n} \varepsilon(j) * \left(\frac{\frac{FO(u,i)}{TOI(u,i)} + \frac{R(u,i)}{TI(u,i)}}{2} \right)}{n}$$

where

- $\sum_{j=1}^{n}$ represents the number of time windows established by the *domain u*.
- $\varepsilon(j)$ depicts weighting factor to be considered by the *domain u* for each the time window j. $\varepsilon(j) \in [0,1]$ and $\varepsilon_1 + \cdots + \varepsilon_n = 1$.
- $FO(u, i)$ is the total number of *feedbacks* of a particular type of *offer* that have been published about the *domain u* in the DLT.
- $R(u, i)$ means the total number of recommendations made by other domains about the *domain u* and published in the DLT.
- $TOI(u, i)$ is the total number of offer interactions recorded in the i-th interaction of *the domain u*.
- $TI(u, i)$ is the total number of provider interactions recorded in the i-th interaction of *the domain u*.

8.2.4. Community Context Factor equation

The purpose of the $CF(u)$ is to obtain the feedacks about a *domain u*. For this purpose, the interaction number that the *domain u* had in the community through the contribution of services or resources with other domains are evaluated. In addition, a dynamic list of trustworthy recommenders is contemplated to ask for *domain u*'s feedbacks. Finally, we also consider an aggregation function in order to set up.

$$CF(u) = \frac{\frac{R(u)}{TI(u)} + \frac{\oplus_{j=1}^{n} \left(Rec(j,u) * Cr(j)\right)}{n}}{2}$$

where

- $Rec(j, u)$ the recommendation of the j-th domain with which the *domain u* has a trust relationship and it is in our list of trustworthy recommenders. In other words, $T(j, u)$.

<END OF DOCUMENT>