5G ZORRO

Grant Agreement 871533

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

D3.2: Prototypes of evolved 5G Service layer solutions

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: **871533**	Project Acronym: **5GZORRO**	Project title: **Zero-touch security and trust for ubiquitous computing and connectivity in 5G networks.**

Lead Beneficiary: **i2CAT**	Document **version: V1.1**

Work package: **WP3 - Evolved 5G Service layer with 5G DLT and distributed AI**

Deliverable title: **D3.2: Prototypes of evolved 5G Service layer solutions**

Start date of the project: **01/11/2019 (duration 30 months) (extended to 36 months)**	Contractual delivery date: **30.06.2022**	Actual delivery date: **30.06.2022**

Editor(s)
Adriana Fernández-Fernández (i2CAT)

List of Contributors

Participant	Short Name	Contributor
Fundaciò i2CAT	i2CAT	Adriana Fernández-Fernández, Carlos Herranz, Daniel Bautista, S. Siddiqui
Nextworks	NXW	Michael De Angelis, Pietro G. Giardina, Giacomo Bernini
IBM Israel Science and Technology	IBM	Kalman Meth, Katherine Barabash
Altice Lab	ALB	Bruno Santos, André Gomes
Ubiwhere	UW	Filipa Martins
Intracom Telecom	ICOM	Dimitris Laskaratos, Georgios Samaras
ATOS Spain	ATOS	Guillermo Gomez
Universidad de Murcia	UMU	José María Jorquera Valero
Fondazione Bruno Kessler	FBK	Raman Kazhamiakin
Malta Communications Authority	MCA	Jean-Marie Mifsud, Antoine Sciberras, Andrew Caruana, Charlo Baldacchino

List of Reviewers

Participant	Short Name	Contributor
IBM Israel Science and Technology	IBM	Katherine Barabash
Ubiwhere	UW	Rita Santiago
Nextworks	NXW	Giacomo Bernini
Fundaciò i2CAT	i2CAT	S. Siddiqui

Change History

Version	Date	Partners	Description/Comments
0.0	18/05/2022	i2CAT	Table of contents and assignments
0.1	02/06/2022	ALB, NXW, IBM	Contents of Governance DLT (2.1), Identity and Permissions Manager (2.2), Legal Prose Repository (2.3) and Operational Data Lake (4.1)
0.2	14/06/2022	ALB, ICOM, UMU, i2CAT, UW, FBK, NXW	Contents of Identity and Permissions Manager (2.2), Governance Portal (2.4), Governance Validation Tests (2.5), Resource and Service Offer Catalogue (3.2), Smart Contracts Lifecycle Manager (3.3.), (Marketplace Portal (3.4), Marketplace Validation Tests (3.5), Monitoring Data Aggregator (4.2), SLA Monitoring (4.3), Intelligent SLA Breach Predictor (4.4) and Smart Resource and Service Discovery (4.5), AIOps Validation Tests (4.6) and Installation Procedures (5)
0.3	17/06/2022	ATOS, IBM, UW NXW, ICOM	Contents of Governance Validation Tests (2.5), Marketplace DLT (3.1), Marketplace Validation Tests (3.5) and AIOps Validation Tests (4.6) Version ready for tech review

0.4	23/06/2022	UW, IBM, ALB, i2CAT	Provide tech review comments, Address review comments
0.5	27/06/2022	ALB, NXW, IBM, FBK, i2CAT	Process tech review outcomes and update references/formatting across the document Version ready for QA review
1.0	30/06/2022	MCA, ICOM, IBM, NXW, i2CAT	QA review comments, Process QA review outcomes and update references/formatting across the document Version ready for submission
1.1	22/02/2023	i2CAT	Typo correction. Final version for public release.

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 presents the resulting prototypes of the Evolved 5G Service layer solutions that have been developed as part of the 5GZORRO platform. These have been derived from the activities done within the Work Package (WP) 3, and specifically the document presents the essential features of all the Evolved 5G Service layer software components. The implemented prototypes are aligned to the design guidelines reported in previous deliverables (i.e., D3.1 [1] and D3.3 [2]) and are the result of continuous development and integration efforts performed by multiple software teams, taking also into account validation feedback from WP5.

In summary, this document is organized to describe:

- The actual software components that have been implemented inside each one of the three WP3 functional pillars: Governance, Marketplace, and Cross-domain Analytics & Intelligence.
- The description of underlying technology enablers (i.e., Distributed Ledger Technologies (DLTs) and the Data Lake) in terms of configurations and/or customizations implemented in order to realise the 5GZORRO Service layer solutions.
- The main features and implementation details of each developed component, including the intended usage patterns to show what can be achieved by using this prototype.
- Results of validation tests, including functional tests for the various modules and integration tests, to demonstrate the proper functionality of the involved components and their interrelation.
- Hardware and software requirements and installation instructions for the presented prototype, including pointers to the additional documentation about installation and usage that is publicly available online in the project software repository.

1. Introduction

This Deliverable D3.2 documents the final implementation of the 5GZORRO Evolved 5G Service layer solutions, enabled through Distributed Ledger Technologies (DLTs) and Artificial Intelligence for IT operations (AIOps). The work described in this Deliverable represents the main implementation-related outcomes of tasks T3.1, T3.2 and T3.3. This document complements the documentation of the 5GZORRO Software Platform presented in D2.4 [3], mainly focusing on software modules related to the Evolved 5G Service layer solutions, which are realized by the Governance, Marketplace, Cross-domain Analytics & Intelligence applications as designed in D3.3 [2].

The development work reported in this document reflects the implementation of the 5GZORRO Software Architecture described in Deliverable D2.4 [3]. Starting from this guideline, the architectural components of the 5GZORRO Platform have been implemented in terms of software modules resulting in a mapping between functional entities and actual software components as depicted in Figure 1-1. In order to help the readers to navigate through the rest of the document, the specific software modules developed as part of WP3 are listed and highlighted in the yellow notes of Figure 1-1.

Moreover, we describe the capabilities of the released prototypes by going through some usage scenarios of the user-facing components of the platform, namely the 5GZORRO (Governance & Marketplace) Portal. Results of functional tests for the various modules are also included, together with the validation of the prototypes integration to demonstrate the system capabilities for multi-party offers trading, SLA lifecycle management, as well as the distributed intelligent and automated data-driven resource management. Additionally, this document provides the information necessary to deploy and use the developed prototypes.

This document presents the final release of software prototypes corresponding to the consolidation of WP3 outcomes. These are available in the 5GZORRO GitHub organization [4] and tagged as *5gzorro-full-1.0-pre-final*. Nevertheless, some adjustments can still be realized in the different prototypes until the end of the project based on feedback received from the validation trials performed under WP5, which will be directly reported in the software documentation material included in the available repositories (tagged as *final*).

1.1. Document outline

The document is structured as follows:

- Section 2 describes the software prototypes of the Governance Platform, including the main aspects of the Governance DLT and validation tests related to the management of distributed identifiers and credentials.
- Section 3 describes the software prototypes of the Marketplace Platform, including the main aspects of the Marketplace DLT and validation tests related to the management of offers, contracts, and spectokens.
- Section 4 describes the software prototypes of the AIOps Platform, including the main aspects of the Operational Data Lake and validation tests related to the discovery of offers, the processing of monitoring data, and SLA breach predictions.
- Section 5 provides the instructions necessary to install the developed prototypes, including the mapping of modules required according to the stakeholder role.
- Section 6 concludes and summarises the document.
- Section 7 lists the references.
- Section 8 lists the abbreviations used that appear in the document.

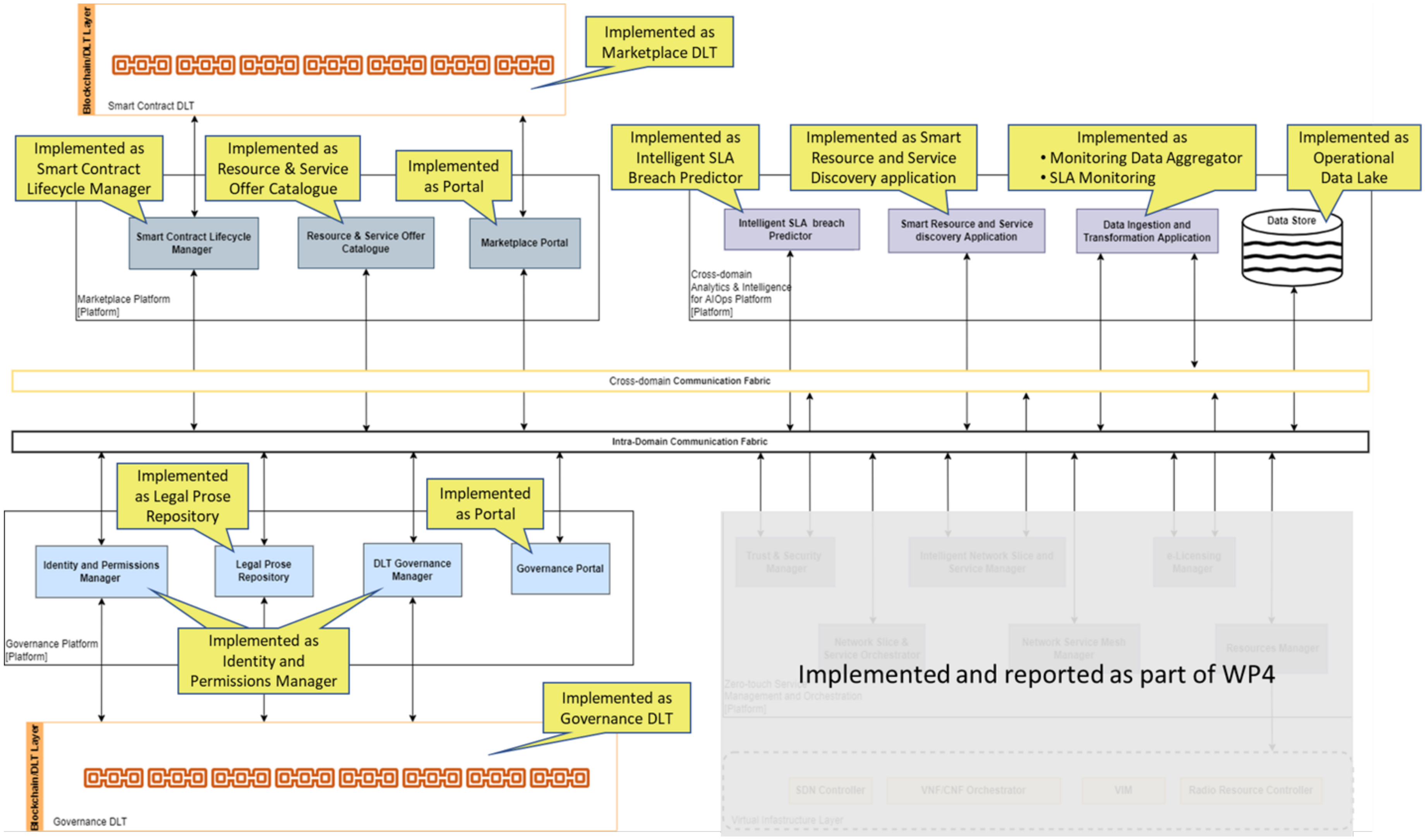

Blockchain/DLT Layer
Smart Contract DLT
Implemented as Marketplace DLT
Implemented as Smart Contract Lifecycle Manager
Implemented as Resource & Service Offer Catalogue
Implemented as Portal
Smart Contract Lifecycle Manager
Resource & Service Offer Catalogue
Marketplace Portal
Marketplace Platform [Platform]
Implemented as Intelligent SLA Breach Predictor
Implemented as Smart Resource and Service Discovery application
Implemented as
• Monitoring Data Aggregator
• SLA Monitoring
Implemented as Operational Data Lake
Intelligent SLA breach Predictor
Smart Resource and Service discovery Application
Data Ingestion and Transformation Application
Data Store
Cross-domain Analytics & Intelligence for AIOps Platform [Platform]
Cross-domain Communication Fabric
Intra-Domain Communication Fabric
Implemented as Legal Prose Repository
Implemented as Portal
Identity and Permissions Manager
Legal Prose Repository
DLT Governance Manager
Governance Portal
Governance Platform [Platform]
Implemented as Identity and Permissions Manager
Implemented as Governance DLT
Blockchain/DLT Layer
Governance DLT
Trust & Security Manager
Intelligent Network Slice and Service Manager
e-Licensing Manager
Network Slice & Service Orchestrator
Network Service Mesh Manager
Resources Manager
Implemented and reported as part of WP4
Zero-touch Service Management and Orchestration [Platform]
SDN Controller
VNF/CNF Orchestrator
VIM
Radio Resource Controller
Virtual Infrastructure Layer

Figure 1-1: Software modules implementing the 5GZORRO Marketplace, Governance and Cross-domain Analytics & Intelligence for AIOps Platforms

2. Governance Platform Prototypes

The 5GZORRO Governance Platform comprises a set of software modules that are deployed by Administrative type stakeholders, and provides core governance capabilities essential to marketplace functionalities. Services include marketplace governance management through stakeholder identity, decentralized credentials & licenses verification and authorization, Distributed Identifier (DID) tagging of assets (Product Offers, SLAs, etc.), and a managed repository for submitted legal prose templates. Whilst deployed and operated exclusively by admin stakeholders, the services provided by these applications are made available to all 5GZORRO stakeholders, that are Regulators or Traders, and their platforms, through service interfaces. Some of these services can be utilised by non-admin stakeholders, using enabled service endpoints according to the type of stakeholder interacting with them.

A governance portal is provided to all stakeholder types, to enable governance admin stakeholders to perform associated workflow tasks, such as accepting/rejecting platform membership requests, reviewing proposed legal prose template updates, etc., and to enable non-admin stakeholder actions in the scope of requesting membership to the marketplace, requesting licenses to trade certain products and/or services, and more. Figure 2-1 quickly summarizes the interactions between the available Governance applications and the Governance DLT that supports the aforementioned capabilities.

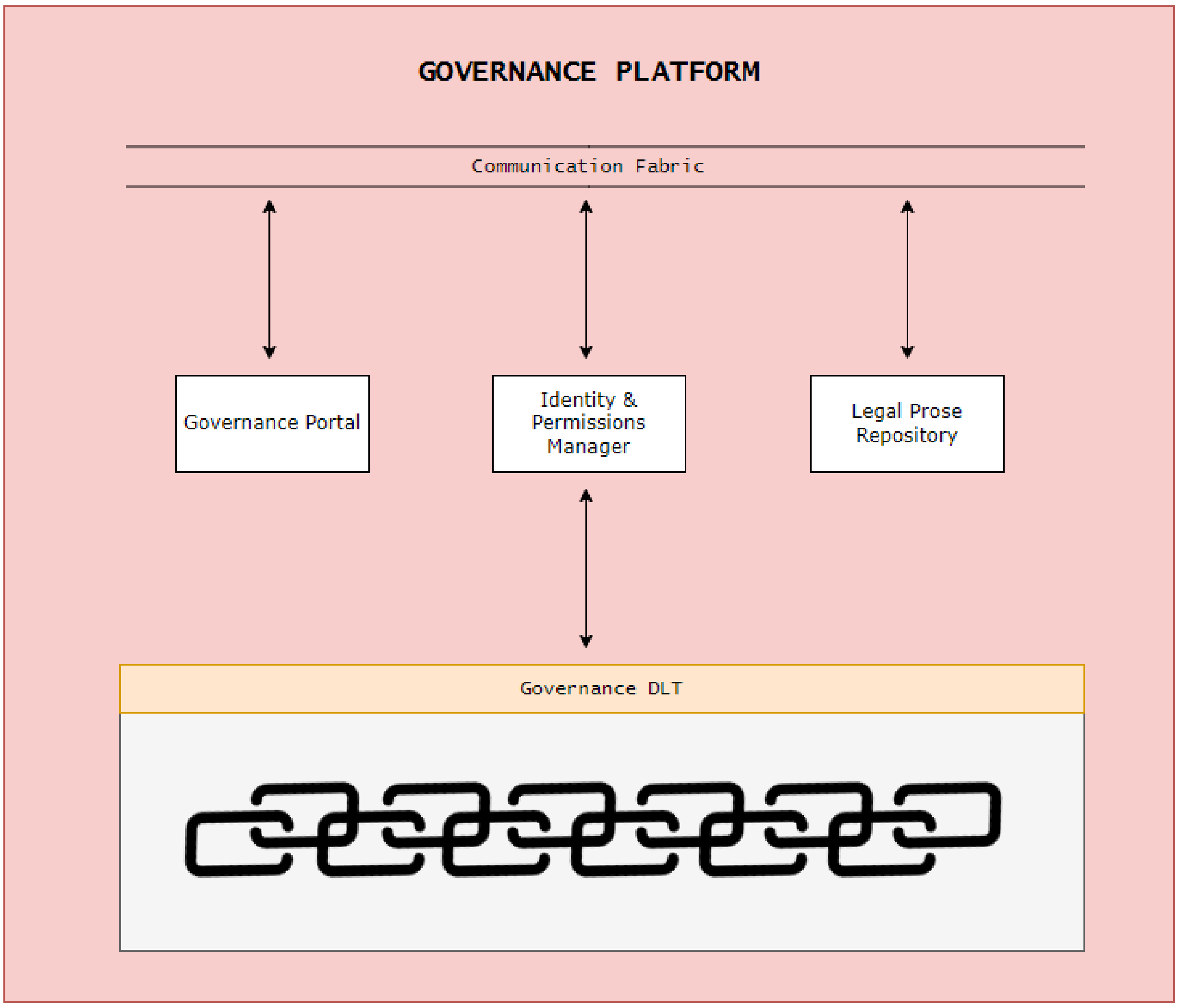


Figure 2-1: Governance Platform Architecture

The Governance Manager framework functionalities, detailed in previous design documents (i.e. D3.1 [1] and D3.3 [2]), are provided by the Identity & Permissions Manager (ID&P) component, assuring, at a technical level, the logic to perform governance decisions based on proposals. This is done by issuing Verifiable Credential (VC) claims, to check the membership status of stakeholders' credentials, and to revoke issued memberships whenever a breach is confirmed.

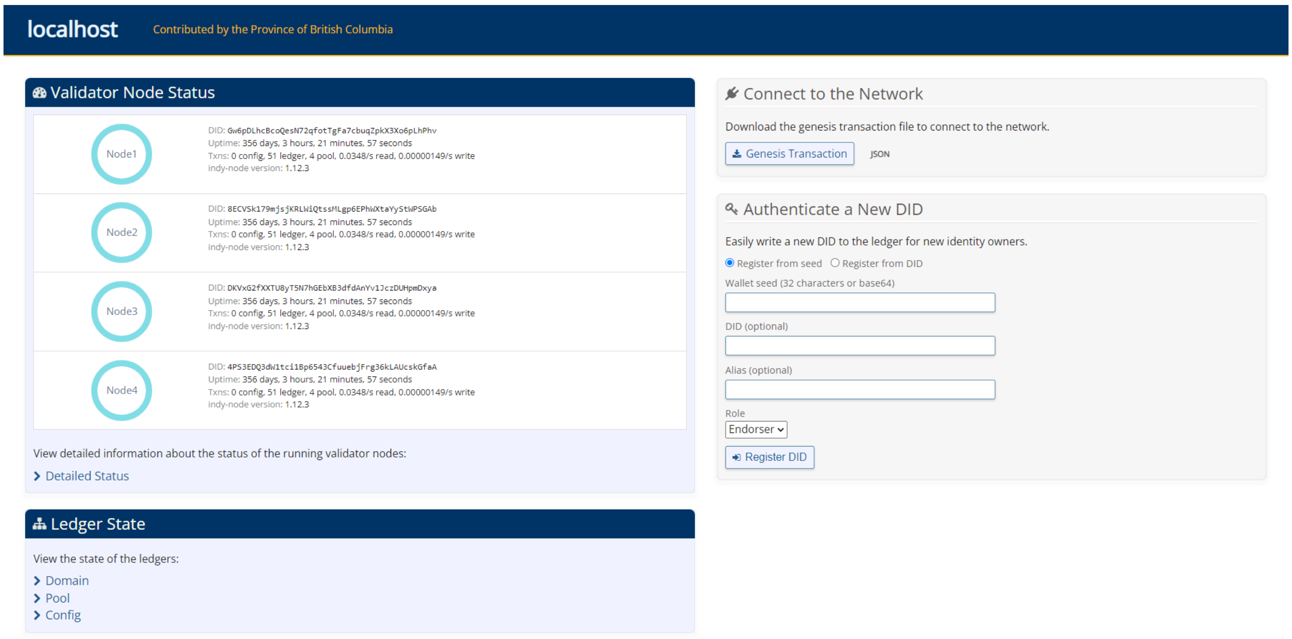
2.1. Governance DLT

As a distributed ledger, the Governance DLT has the job of supporting the ID&P in providing decentralized verifiable DID Registry features, including creation of Distributed Identifiers and VC schemas, enabling the revocation of issued credentials, and issuing public keys.

2.1.1. Main Functionalities

As detailed in Deliverable D3.1 [1], the DLT is meant to support the governance aspect of the marketplace, providing a high level of transparency by enabling a decentralized management of the platform. A DLT is crucial to give the ability for each stakeholder to participate in the marketplace, stating their intended role and capabilities. In turn, by having their information validated and approved, the marketplace will inherently have a high level of confidence and trust in performing negotiation actions with the stakeholder and its assets.

Having the Sovrin Foundation [5] type of DIDs as basis, the Governance DLT includes a Ledger Browser, contributed by the Province of British Columbia [6], that enables the registration of new DIDs to the Ledger for new identity owners. The Ledger Browser also allows an admin to see the status of the nodes of a network (under Validator Node Status), and browse, search, and filter the transactions performed on the Ledger (by accessing Ledger State). As depicted in Figure 2-2, it is possible to overview the layout of the Governance DLT's GUI.


localhost Contributed by the Province of British Columbia

Validator Node Status

Node1
DID: Gw6pDLhcBcoQesN72qfotTgFa7cbuqZpkX3Xo6pLhPhv
Uptime: 356 days, 3 hours, 21 minutes, 57 seconds
Txns: 0 config, 51 ledger, 4 pool, 0.0348/s read, 0.00000149/s write
indy-node version: 1.12.3

Node2
DID: 8ECVSk179mjsjKRLWiQtssMLgp6EPhWXtaYyStWPSGAb
Uptime: 356 days, 3 hours, 21 minutes, 57 seconds
Txns: 0 config, 51 ledger, 4 pool, 0.0348/s read, 0.00000149/s write
indy-node version: 1.12.3

Node3
DID: DKVxG2fXXTU8yT5N7hGEbXB3dfdAnYv1JczDUHpmDxya
Uptime: 356 days, 3 hours, 21 minutes, 57 seconds
Txns: 0 config, 51 ledger, 4 pool, 0.0348/s read, 0.00000149/s write
indy-node version: 1.12.3

Node4
DID: 4PS3EDQ3dW1tci1Bp6543CfuuebjFrg36kLAUcskGfaA
Uptime: 356 days, 3 hours, 21 minutes, 57 seconds
Txns: 0 config, 51 ledger, 4 pool, 0.0348/s read, 0.00000149/s write
indy-node version: 1.12.3

View detailed information about the status of the running validator nodes:
› Detailed Status

Ledger State

View the state of the ledgers:
› Domain
› Pool
› Config

Connect to the Network

Download the genesis transaction file to connect to the network.
Genesis Transaction JSON

Authenticate a New DID

Easily write a new DID to the ledger for new identity owners.
[✓ Register from seed] Register from DID
Wallet seed (32 characters or base64)
DID (optional)
Alias (optional)
Role: Endorser
Register DID

Figure 2-2: Governance DLT Ledger Browser

By having distributed nodes, the ecosystem does not require reliance on a centralized system to represent, control, or verify an entity's digital identity data, providing load balancing features whenever one of the nodes is unavailable at a given time.

Taking into account that various legal requirements apply across Europe in relation to authenticated persons/entities that own certain types of assets, there is a huge level of obligation for an entity to be represented, exchanged, secured, protected, and verified in an interoperable manner, using open, public standards. As such, the Governance DLT is crucial component on the operations of the ID&P, because it allows for the latter to issue a requested VC, as part of the stakeholder on-boarding workflow, that can be used in an open Marketplace.

2.1.2. Prototype Implementation

The Governance DLT in 5GZORRO is based on a VON Network [7], an open-source project developed by the Province of British Columbia, which corresponds to the Governance DLT detailed before. This project itself is based on the Hyperledger Indy project [8], which is currently the main reference implementation of a distributed ledger, based on Redundant Byzantine Fault Tolerant (RBFT) state machine replication consensus protocols, to provide a W3C compliant self-sovereign identity ecosystem.

As illustrated below, in Figure 2-3, by relying on Indy's nodes, it is possible to establish DID Communications between the Hyperledger Aries-based Identity & Permissions Agent (see Section 2.2) and the Governance DLT, by connecting to one of the available nodes, which provide an asynchronous encrypted protocol for secure, private, and authenticated message-based communication, where trust is rooted in DIDs and used over a wide variety of transports.

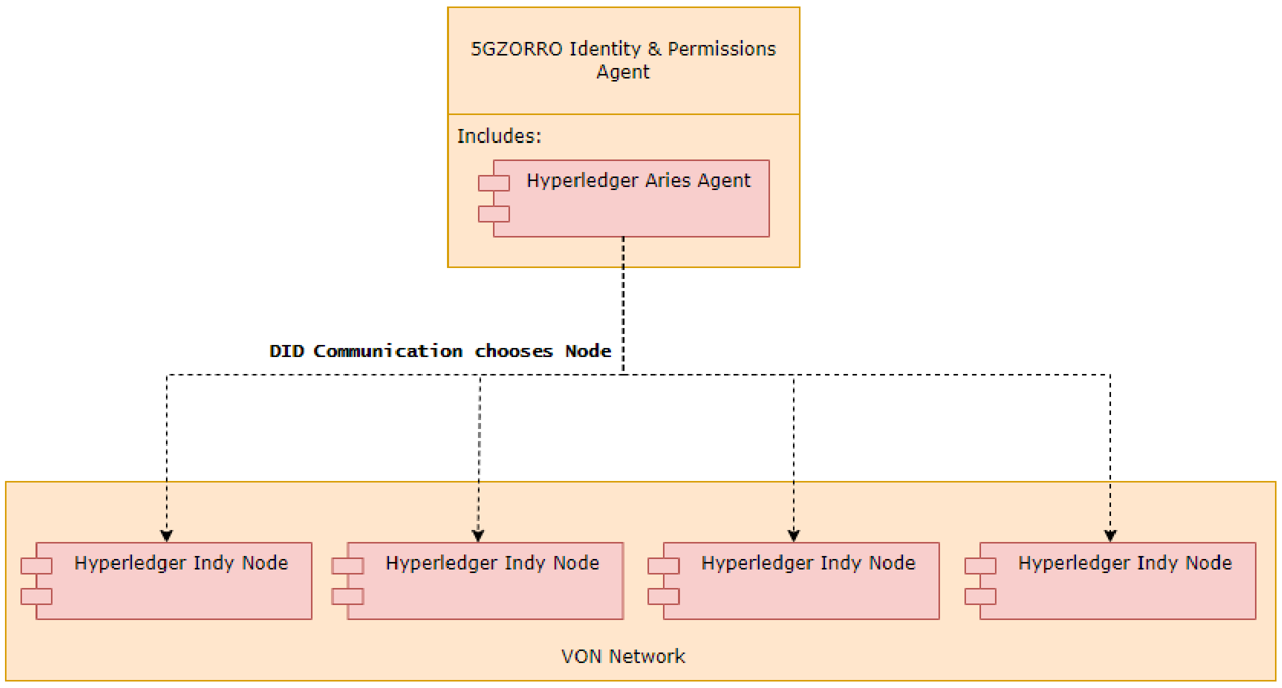


Figure 2-3: DID Communication between Governance DLT – Identity & Permissions

An Indy Node contains everything required to run the node of a distributed ledger for Self-Sovereign Identity. It is separated from, but commonly associated with, the distributed ledger from the Sovrin Foundation. Due to the usage of the Hyperledger Identity stack (see D3.1 [1]), there is a more seamless integration with Indy and other layers of the application, which effectively removes the worrying aspect of setting up each of the nodes.

Currently, the Governance DLT requires a Virtual Machine running with an Ubuntu Operating System compatible with Docker, to leverage the project's images into several containers, both for the virtualized nodes and the controller, considering that the DLT is a multi-container application. At the hardware level, the Governance DLT shares the same Virtual Machine with the ID&P component, which had allocated to it 2 virtual CPUs, with a 2GHz maximum potency each, 4 GB of RAM, and 80 GB of internal storage.

This module, by being the main artefact in the registration of DIDs and VCs into the blockchain, alongside the Hyperledger Aries Agents, is a stated pre-requisite of the Identity & Permissions component.

2.2. Identity and Permissions Manager

The Identity and Permissions Manager (ID&P) module is responsible for facilitating and coordinating marketplace governance in a decentralized fashion. It supplies the mechanisms required to:

- Generate unique identifiers in the 5GZORRO ecosystem.
- Recognise communicating endpoints between the Agents.
- Identify and authorize entities, services, and organizations to access provisioned services and resources in the 5GZORRO marketplace.

This decentralization aspect is comprised by having several ID&P Agents, that are attributed to certain entities during business negotiations, to which they occupy certain roles in the marketplace: Administrators, Regulators or Traders.

ID&P enables trading-centric entities to identify providers, consumers, services, offers, etc., by upholding VCs associated with DIDs. The module prevents unauthorised access to functionalities, services, resources, and data, making access control enforcement as granular as possible. Credentials that represent rights to engage on certain assets are reviewed and managed by Administrators & Regulators.

Any Administrator of the platform is responsible for reviewing and issuing a decision on a stakeholder credential proposal, subject to governance. Likewise, a Regulator is tasked to oversee and resolve a license credential proposal. This decentralized process is supported by the Governance DLT (described previously) in order to ensure full transparency and auditability.

2.2.1. Main Functionalities

To support the functionalities envisioned for the component and the governance aspects of the platform, Hyperledger ARIES [10] serves as the main baseline for some of the Identity and Permission Manager features.

More precisely, ID&P relies on Hyperledger ARIES Cloud Agent Python (ACA-Py) [9], a framework that complements Hyperledger Indy, by providing a toolkit to create DID Agents that manage the creation, transmission, storage and/or verification of DID verifiable digital credentials that are compliant with W3C VCs. This project is jointly working with the Decentralized Identity Foundation (DIF) [11] to develop a secure and standard communication based on DIDs — DID Comm – to enable DID Agents interoperability, regardless of any used DLT technology, as interoperability is crucial for the transmission aspect of the DIDs.

As described in Figure 2-4, the high-level architecture of an ID&P Agent is implemented as follows:

Step 1: An Identity & Permissions Agent controller is responsible for interacting with an associated Hyperledger ACA-Py instance. This instance registers a seed in the Governance DLT, in order to generate a unique wallet, bound to the agent, that is provisioned by the DLT.

Step 2: The ACA-Py instance exposes a REST API to be used by the ID&P controller for all the functionalities it is configured to handle. Each of the DID Comm protocols supported by the agent adds a set of messages for the controller to use in responding to events.

Step 3: Whenever an ID&P controller interacts with its assigned ACA-Py for credential requests and/or issuance orders, the DID Comm protocols existent between the Governance DLT and the Hyperledger ARIES Agent are triggered, thus resulting in writing events to the Blockchain, logging the performed transactions.

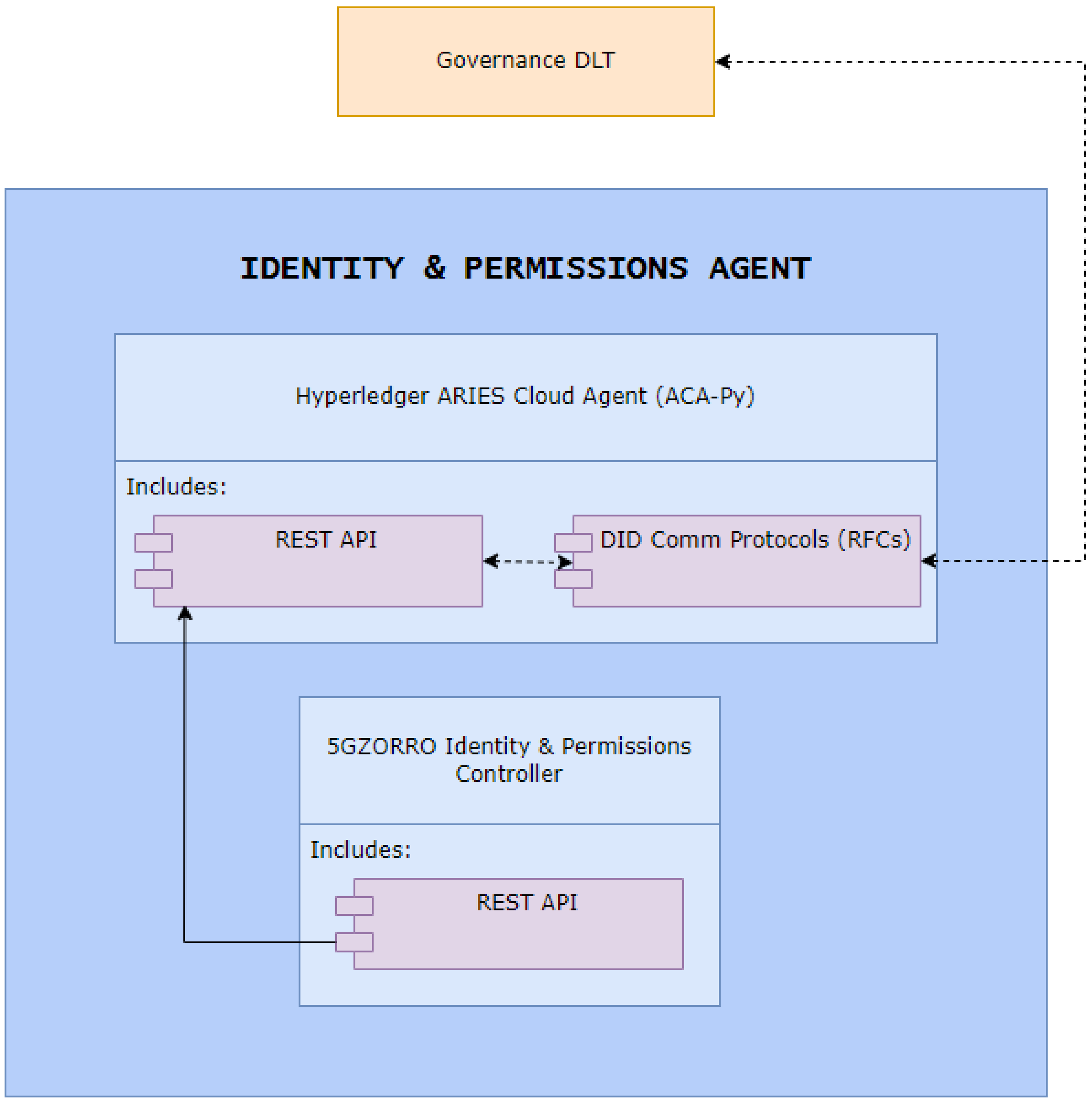

Governance DLT
IDENTITY & PERMISSIONS AGENT
Hyperledger ARIES Cloud Agent (ACA-Py)
Includes:
REST API
DID Comm Protocols (RFCs)
5GZORRO Identity & Permissions Controller
Includes:
REST API

Figure 2-4: Architecture of Identity & Permissions integration with Hyperledger ARIES

The process of approving or declining a Trader-type stakeholder application for a Stakeholder Membership is partaken by an Administrator. In this case, a Trader has to state their intended role and capabilities. Regarding Stakeholder Licensing, the process is largely the same, with the difference being the Regulator Agent handling it. The steps are illustrated in Figure 2-5.

Step 1: A Credential proposal is made by an entity to the Governance Portal deployed in its assigned stakeholder domain. The user can keep track of the status of the Credential request on the appropriate interfaces.

Step 2: Governance Portal dispatches the request to the corresponding Identity & Permissions Agent, where a unique DID is created for global identification of the proposal, and the proposal is stored in distributed storage with a status of REQUESTED.

Step 3-4: After being notified by the ID&P Trading Agent, the Administrator/Regulator stakeholder reviews the information associated with the proposal through the Governance Portal and, by cross-checking with relevant documents, issues a decision (approval/rejection) for the VC proposal.

Step 5-6 (only on approval): When deciding to approve the Credential, the required procedures are performed to issue the VC on the blockchain, through the Governance DLT, and to register a transaction of this action.

Step 7: The outcome is then relayed across the involved components, to then reach the Governance Portal where the entity can overview the decision & the VC's info.

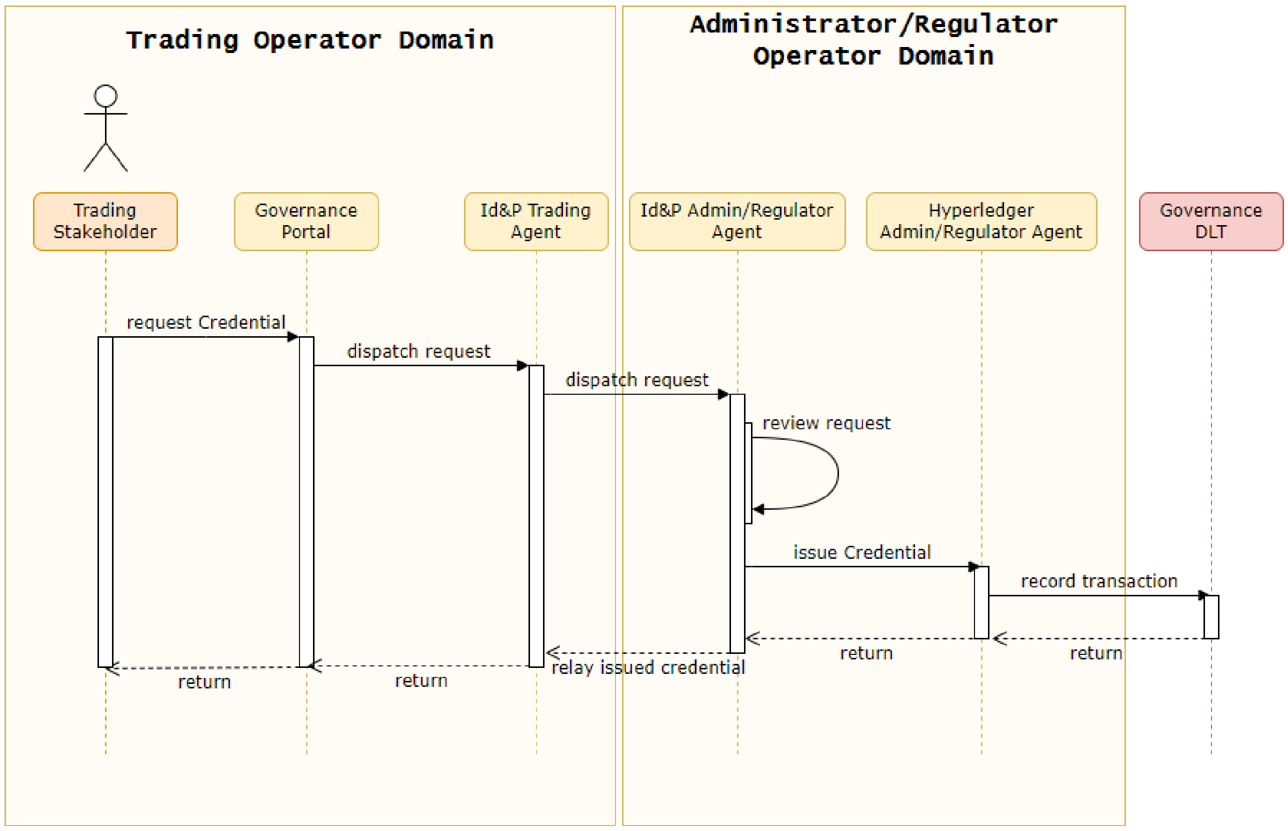


Figure 2-5: VC request/issuance Workflow

2.2.2. Prototype Implementation

The DID Agent component, as explained previously, is the core deployable component of 5GZORRO ID&P, and it follows the main principles of DIF Cloud Agent design [11]. Each DID Agent holds a DID Wallet, according to the DLT technology used, that includes DIDs issued by the Agent, VCs issued by an Admin/Regulator Agent, and secured storage to handle private keys.

To check the module's code and instructions on the needed requirements and how to deploy it, the following reference is applicable [12]. The repository, accessible through the public 5GZORRO GitHub space, is open-source and available under APACHE 2.0 License.

By exposing the REST API of the DID Agents through OpenAPI, components such as the Governance Portal, the Legal Prose Repository (LPR), the Smart Contract Lifecycle Manager (SCLCM), etc., can interact with the interfaces of the appropriate ID&P Operator instances.

For the prototype released with this deliverable, there are envisaged three main types of DID Agent components (see Figure 2-6).

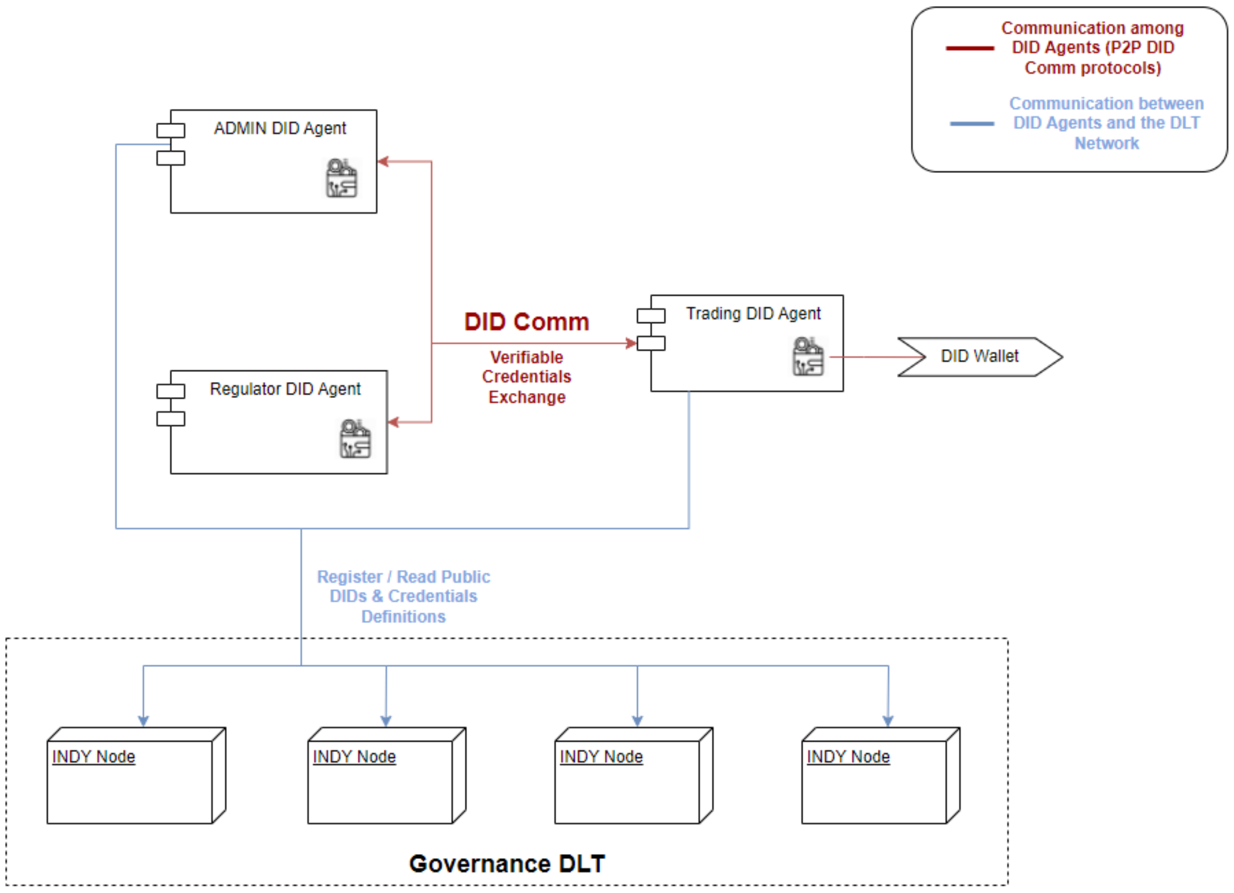
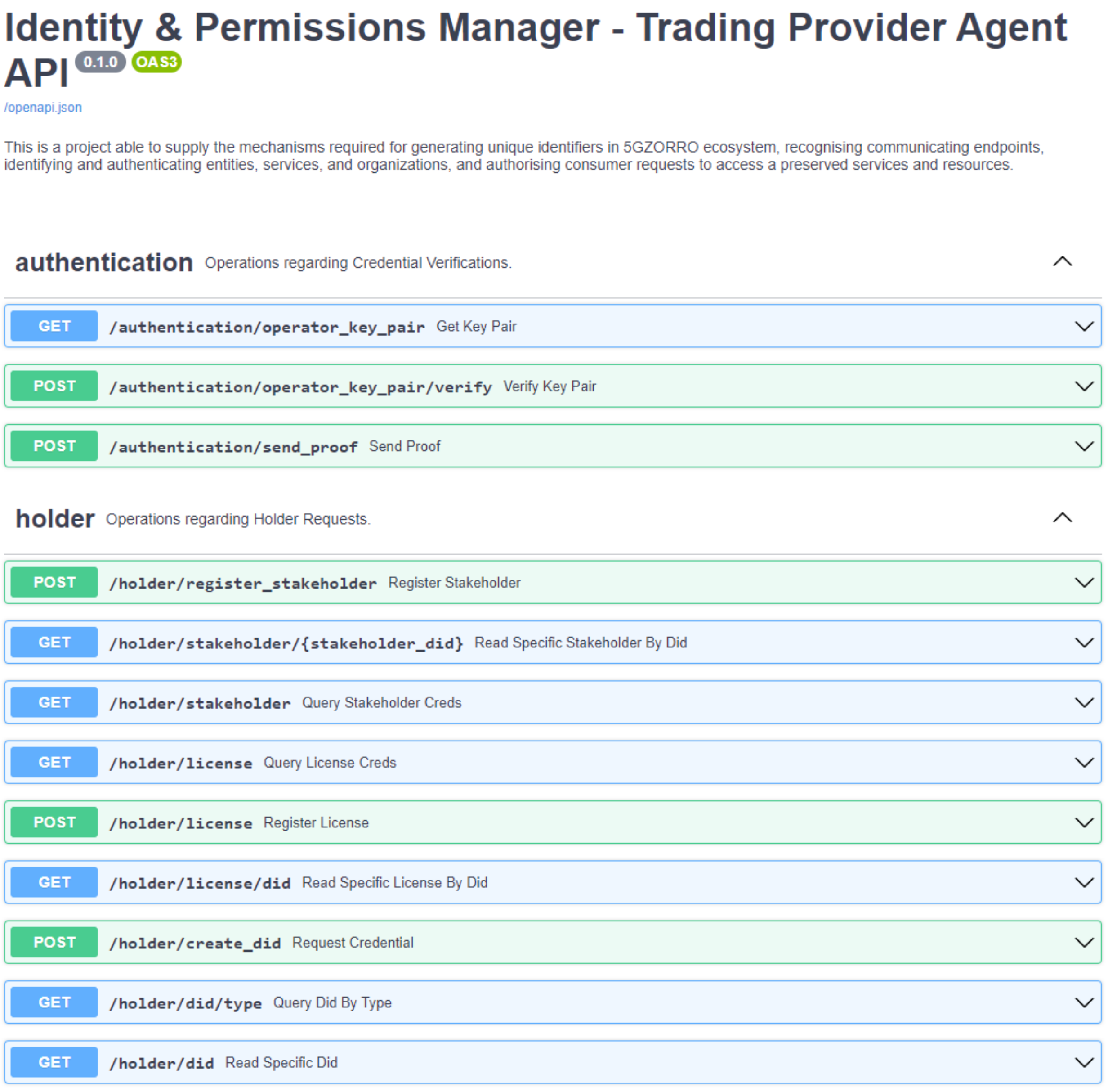

Communication among DID Agents (P2P DID Comm protocols)
Communication between DID Agents and the DLT Network
ADMIN DID Agent
Regulator DID Agent
DID Comm
Verifiable Credentials Exchange
Trading DID Agent
DID Wallet
Register / Read Public DIDs & Credentials Definitions
INDY Node
INDY Node
INDY Node
INDY Node
Governance DLT

Figure 2-6: ID&P Deployable Agents and their interactions

- **Trading DID Agent**: it provides specific business logic to request & manage VCs required to support trustworthy trading of 5G Offers in 5GZORRO Marketplace. The Trading DID Agent will require to have a Credential approved by an Admin/Regulator Agent. Trading Agent DIDs will be registered in the Governance DLT via endorsers, which correspond to the existing Admin DID Agents. By checking Figure 2-7, the endpoints from the OpenAPI available to the Trading Agent type can be consulted. As detailed in D3.3 [2], this DID Agent may play two roles:
    - Provider Trading Agent: it plays the role of the Verifiable Credential Offer Provider
    - Consumer Trading Agent: it plays the role of the Verifiable Credential Offer Consumer.



Identity & Permissions Manager - Trading Provider Agent API 0.1.0 OAS3

/openapi.json

This is a project able to supply the mechanisms required for generating unique identifiers in 5GZORRO ecosystem, recognising communicating endpoints, identifying and authenticating entities, services, and organizations, and authorising consumer requests to access a preserved services and resources.

authentication Operations regarding Credential Verifications.

```
GET  /authentication/operator_key_pair          Get Key Pair
POST /authentication/operator_key_pair/verify   Verify Key Pair
POST /authentication/send_proof                 Send Proof
```

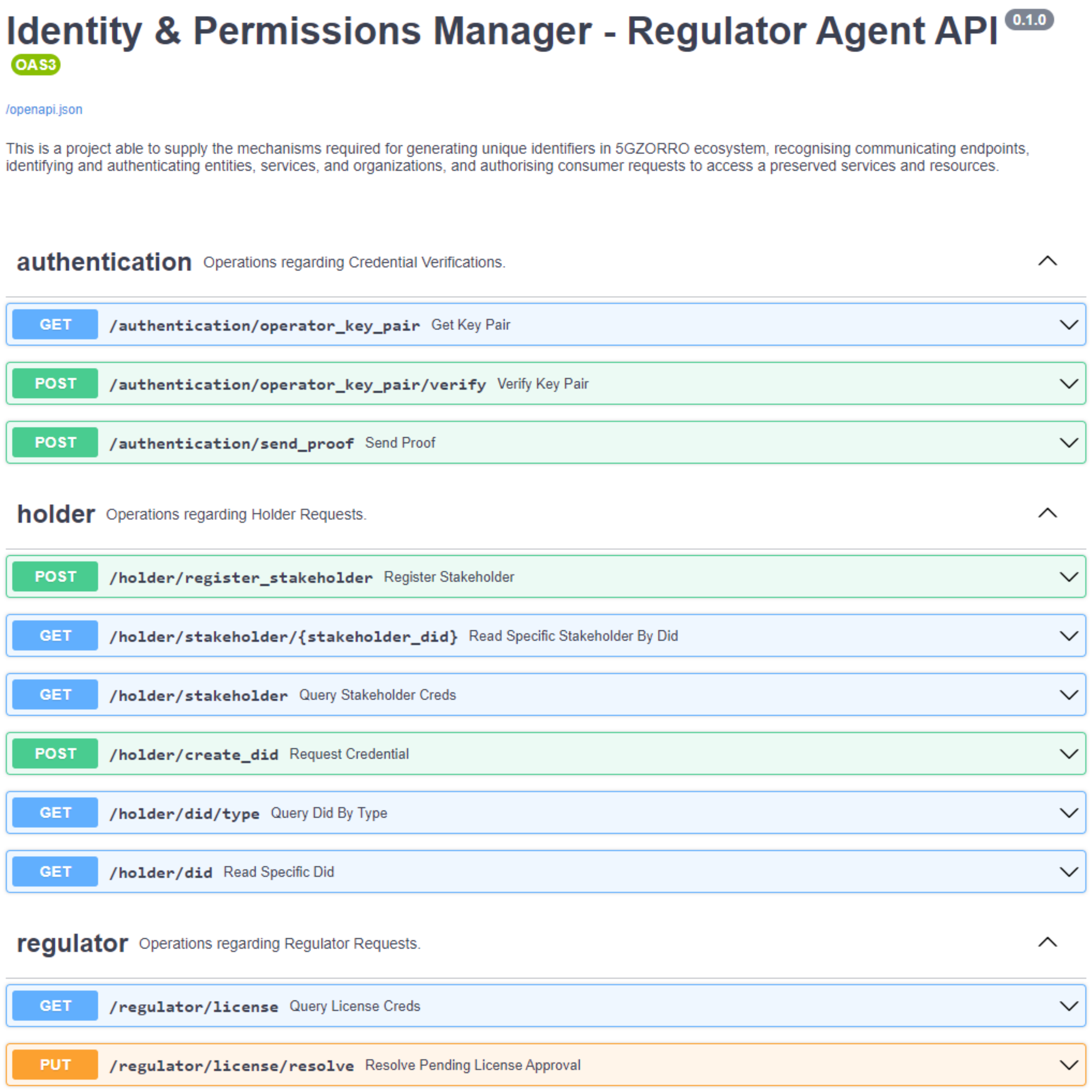
holder Operations regarding Holder Requests.

```
POST /holder/register_stakeholder               Register Stakeholder
GET  /holder/stakeholder/{stakeholder_did}      Read Specific Stakeholder By Did
GET  /holder/stakeholder                        Query Stakeholder Creds
GET  /holder/license                            Query License Creds
POST /holder/license                            Register License
GET  /holder/license/did                        Read Specific License By Did
POST /holder/create_did                         Request Credential
GET  /holder/did/type                           Query Did By Type
GET  /holder/did                                Read Specific Did
```

Figure 2-7: ID&P Trading Agent OpenAPI

The main operations partaken by the Trading Agent when using its corresponding API can be found under the **Holder** tag, as it includes the operations & capabilities of a Trading Operator, such as applying for a Stakeholder Membership and/or License Credential. The endpoints under the aforementioned tag serve the Governance Portal interface for the Trader.

- **Regulator DID Agent**: This Agent provides additional business logic required to issue VCs representative of Licenses, for 5G regulated resources notably 5G Spectrum resources. Since acting on issuance of credentials just as Admin DID Agents, Regulator DID Agents also have permissions to write in the Governance DLT. Figure 2-8 displays the Regulator's OpenAPI interface.


Identity & Permissions Manager - Regulator Agent API 0.1.0 OAS3

/openapi.json

This is a project able to supply the mechanisms required for generating unique identifiers in 5GZORRO ecosystem, recognising communicating endpoints, identifying and authenticating entities, services, and organizations, and authorising consumer requests to access a preserved services and resources.

authentication Operations regarding Credential Verifications.

GET /authentication/operator_key_pair Get Key Pair

POST /authentication/operator_key_pair/verify Verify Key Pair

POST /authentication/send_proof Send Proof

holder Operations regarding Holder Requests.

POST /holder/register_stakeholder Register Stakeholder

GET /holder/stakeholder/{stakeholder_did} Read Specific Stakeholder By Did

GET /holder/stakeholder Query Stakeholder Creds

POST /holder/create_did Request Credential

GET /holder/did/type Query Did By Type

GET /holder/did Read Specific Did

regulator Operations regarding Regulator Requests.

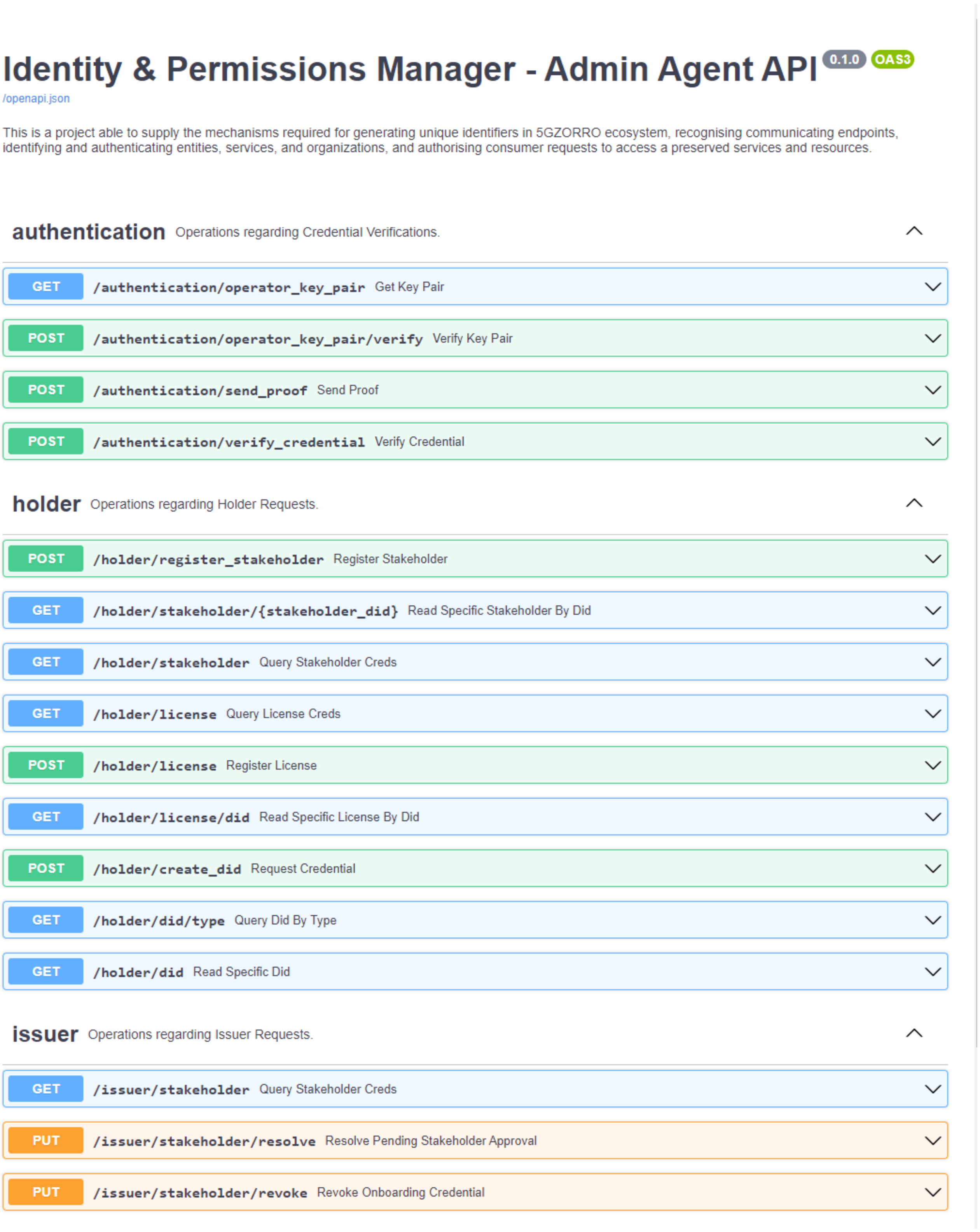
GET /regulator/license Query License Creds

PUT /regulator/license/resolve Resolve Pending License Approval

Figure 2-8: ID&P Regulator Agent OpenAPI

For the Regulator Agent, specifically in the **Regulator** tag, are bestowed the required operations for its role, which correspond to the assessment & decision-making on Licenses requested by Traders, that, in case of approval, result in the issuance of a Stakeholder License Credential. Despite having access to some of the same type of endpoints found in a Trading Agent, the operations & capabilities under the **Holder** tag are limited to the scope of applying & checking for a Stakeholder Membership Credential, since it's not possible for a Regulator to self-issue Licenses to himself. This would also cause conflict of interests. The endpoints that fall under the reported tags serve the Governance Portal interface for the Regulator.

- **Admin DID Agent**: It provides specific business logic to be used to issue VCs on Providers & Regulators request, and to manage said governance VCs, particularly credentials about 5GZORRO Stakeholders Marketplace membership. Admin DID Agents have permissions to write in the Governance DLT and can also interact with the same type of endpoints as the Trading DID Agent, for non-Administrative purposes. In Figure 2-9, the OpenAPI showcasing all available Admin functionalities can be checked.


Identity & Permissions Manager - Admin Agent API 0.1.0 OAS3

/openapi.json

This is a project able to supply the mechanisms required for generating unique identifiers in 5GZORRO ecosystem, recognising communicating endpoints, identifying and authenticating entities, services, and organizations, and authorising consumer requests to access a preserved services and resources.

authentication Operations regarding Credential Verifications.

GET	`/authentication/operator_key_pair`	Get Key Pair
POST	`/authentication/operator_key_pair/verify`	Verify Key Pair
POST	`/authentication/send_proof`	Send Proof
POST	`/authentication/verify_credential`	Verify Credential

holder Operations regarding Holder Requests.

POST	`/holder/register_stakeholder`	Register Stakeholder
GET	`/holder/stakeholder/{stakeholder_did}`	Read Specific Stakeholder By Did
GET	`/holder/stakeholder`	Query Stakeholder Creds
GET	`/holder/license`	Query License Creds
POST	`/holder/license`	Register License
GET	`/holder/license/did`	Read Specific License By Did
POST	`/holder/create_did`	Request Credential
GET	`/holder/did/type`	Query Did By Type
GET	`/holder/did`	Read Specific Did

issuer Operations regarding Issuer Requests.

GET	`/issuer/stakeholder`	Query Stakeholder Creds
PUT	`/issuer/stakeholder/resolve`	Resolve Pending Stakeholder Approval
PUT	`/issuer/stakeholder/revoke`	Revoke Onboarding Credential

Figure 2-9: ID&P Admin Agent OpenAPI

In regards to the Admin Agent, the required operations for its role can be found under the **Issuer** tag. Here, the Admin Agent can use his set of endpoints for the assessment & decision-making on Membership requests made by Traders & Regulators alike. If the Admin chooses to approve the request, this will result in the issuance of a Stakeholder Membership Credential. Contrary to a Regulator Agent, the operations & capabilities under the **Holder** tag attributed to an Admin have no limitations, since the Admin can act as a

Trader in a different Marketplace. The endpoints reported under the specified tags serve the Governance Portal interface for the Administrator.

2.3. Legal Prose Repository

The main scope of the Legal Prose Repository (LPR) is to provide a storage service for the legal templates used in 5GZORRO Platform as baseline for the creation of legal documents such as Service Level Agreements (SLAs) and Licenses. The design of the modules, originally reported in deliverable D3.1 [1], has been evolved in line with the platform requirements as explained in D3.3 [2] where its final version is reported.

2.3.1. Main Functionalities

The LPR offers to other platform modules a specific interface to interact with the legal template storage, enabling the possibility to perform template manipulation operations, in line with APIs reported in D3.3 [2]:

- Create new Legal Templates
- Retrieve existing template list and/or specific Legal Template
- Remove existing Legal Templates from the storage

The interface is usually consumed by modules requiring legal templates to perform their operations, i.e., the SCLCM, described in Section 3.3, and the Governance Portal, described in Section 2.4. In particular, the Governance Portal provides a GUI that allows human users to store new templates in the LPR and to modify the existing ones to create legal documents. Figure 2-10 shows the LPR interaction with the rest of 5GZORRO Platform.

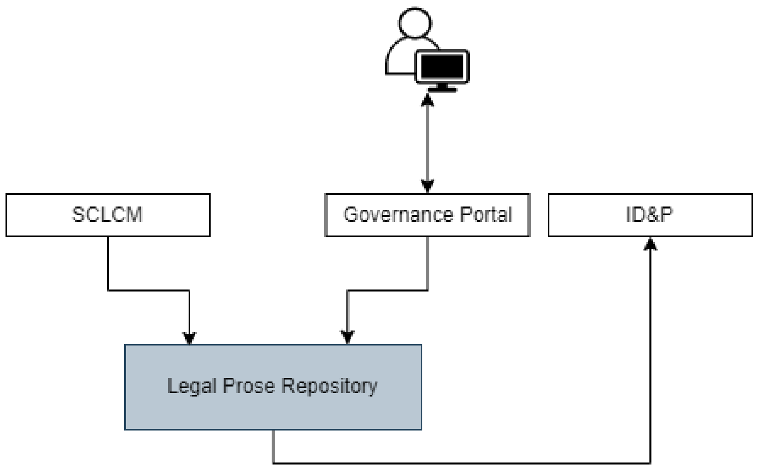


Figure 2-10: LPR interaction schema

Following the figure above, it can be noticed that the LPR interacts also with the ID&P (see Section 2.2). This is due to the fact that each Legal Template in the LPR is univocally identified in the federation of 5GZORRO stakeholders through a DID, released by the ID&P when a new template is created and stored in the internal LPR storage.

2.3.2. Prototype Implementation

The LPR is implemented as a Java application, based on Spring Boot [13] application framework that facilitates the integration of the internal submodules. Figure 2-11 depicts the LPR software architecture consisting of 3 main submodules and an internal storage.

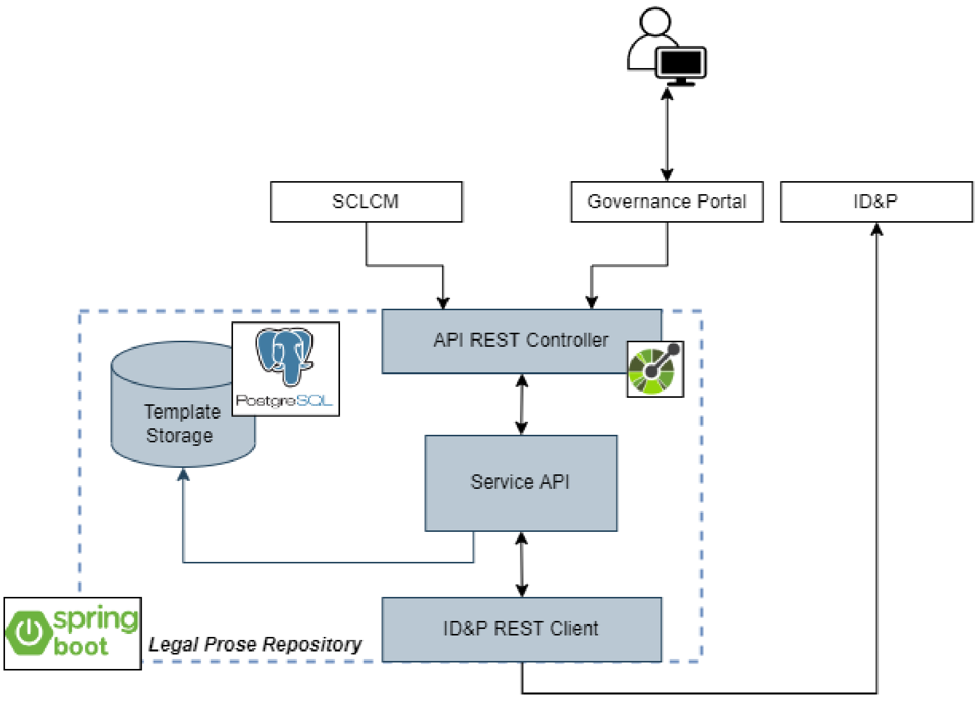
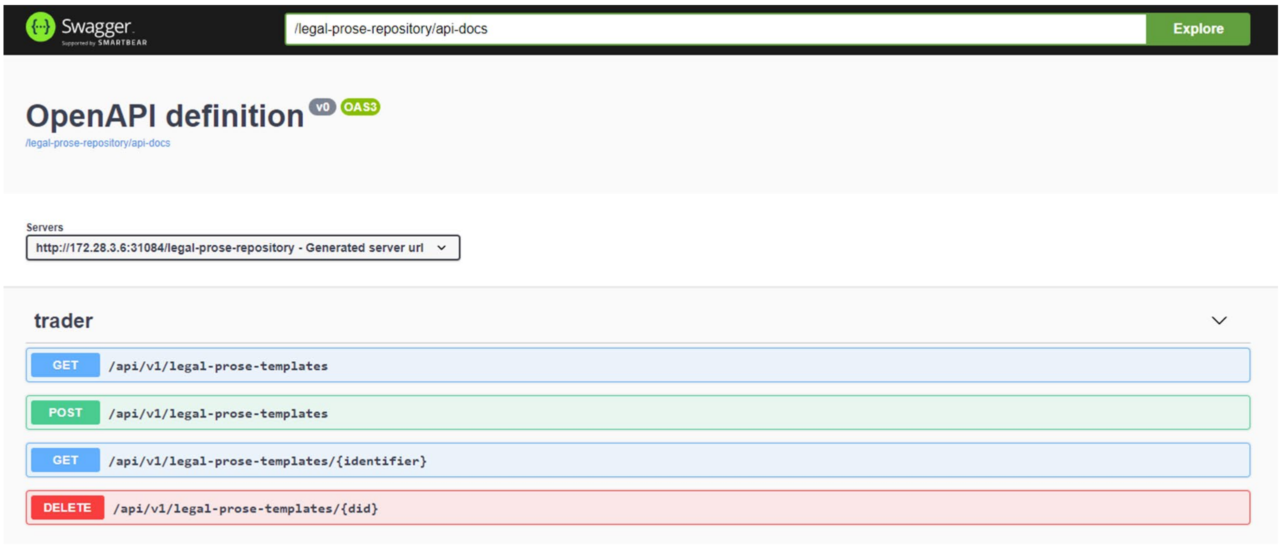


Figure 2-11: LPR software architecture

The *API REST Controller* implements the LPR Northbound Interface (NBI) that exposes the services provided towards the SCLCM and Governance Portal, which consume them for the purposes of the 5GZORRO Marketplace. The set of APIs exposed follows the OpenAPI specification [14] as shown in Figure 2-12.



/legal-prose-repository/api-docs

OpenAPI definition

/legal-prose-repository/api-docs

Servers

http://172.28.3.6:31084/legal-prose-repository - Generated server url

trader

GET /api/v1/legal-prose-templates

POST /api/v1/legal-prose-templates

GET /api/v1/legal-prose-templates/{identifier}

DELETE /api/v1/legal-prose-templates/{did}

Figure 2-12: LPR OpenAPI NBI

The NBI is internally mapped to a specific module that implements the *Service API,* which represent the Service Logic behind the API exposed towards the other modules. The Service API module interfaces directly with both *ID&P REST Client* and the *Template Storage*. The former provides the mean to interact with the ID&P for the management of Template DIDs, while the logic behind the requests still resides in the Service API module. The latter is an instance of PostgreSQL [15], used for the storage of the template themselves. As discussed in D3.3 [2], Legal Templates are data in JSON format, parametrized by using a technology called JSON Template Language [16], that allows the dynamic substitution of given parameters injected into a natural language text. Below an example taken from D3.3 [2], here reported for the sake of clarification.

Example Data Model with values

```
{
  "agreement-data": {
    "number": 3,
    "time-unit": "months"
  }
}
```

Example Parametric Clause

Clause 1: **"**Deployed service can be accessed for maximum `${agreement-data.number}` `${agreement-data.time-unit}`**"**

Resulting Text Expansion (Data Model values + Parametric Clause)

Clause 1: **"**Deployed service can be accessed for maximum ***3 months*** **"**

In terms of deployment, the LPR can run directly into a bare metal server, a virtual machine or even as a software container. This third one, i.e. the software container, is the target virtualization form selected for the module in 5GZORRO, where LPR is deployed as Docker [17] container orchestrated by Kubernetes (k8s) [18]. The explicit interaction with the ID&P through a request response paradigm, builds a weak dependency between the two modules: the LPR can run alone, but requires the ID&P for a minimal set of operations related to the creation of a DID. In absence of ID&P, no new templates can be stored, and the operations allowed are limited to Retrieval and Delete.

The code and the OpenAPI definition are open-source and available under an APACHE 2.0 License in a public repo on GitHub, in the project space maintained by 5GZORRO, using the following reference [19].

2.4. Governance Portal

The Governance Portal is a component of 5GOZRRO's web GUI which supports Governance functionalities by interacting directly with ID&P (Section 2.2) and the LPR (Section 2.3).

As previously described in D3.1 [1], 5GZORRO follows a model in which a set of stakeholders have the power to decide on actions previously requested by other stakeholders. This model itself differs for each type of request: e.g., an Administrator Agent is in charge of handling the onboarding of stakeholder into the platform, and a Regulator Agent is in charge of managing the registration of stakeholders' licenses for resource negotiation.

Upon approval or rejection of a stakeholder request, the action is properly signed and registered on-chain, ensuring the authenticity of the process.

2.4.1. Main Functionalities

Being a Web GUI, the Governance Portal provides the views and interfaces needed for the stakeholders to directly interact with the 5GZORRO implemented governance functionalities. As mentioned before, these functionalities are made available by the Portal through the interaction with ID&P and the LPR modules.

Regarding the interaction with the ID&P, the Portal allows a stakeholder to apply for a Stakeholder and/or a Licensing Credential. Depending on the stakeholder type, a different set of functionalities are made available by the Portal that include the proposal, review, approval or rejection of these credentials. The available functionalities are determined by the stakeholder type that is accessing the Portal.

The workflow for requesting, approving, or refusing a Stakeholder Membership Credential and can be described as follows:

1. A Trader/Regulator stakeholder accesses the 5GZORRO Portal and registers.
2. A Stakeholder Membership Credential request is sent to ID&P module.
3. The stakeholder navigates to the login page and receives the approval status of his credential.
4. A Governance Administrator stakeholder accesses the 5GZORRO Portal.
5. The Administrator navigates to his personal interface showing all Stakeholder Membership Credentials pending for approval.
6. The Administrator approves or refuses the Stakeholder Membership Credential.
7. If the credential was approved the Trader/Regulator stakeholder is given access to the 5GZORRO Portal.

The workflow for requesting, approving, or refusing a license credential can be described as follows:

1. A Trader stakeholder accesses the 5GZORRO Portal and requests a License Credential.
2. A License Credential request is sent to ID&P module.
3. The Stakeholder navigates to his personal interface to watch the approval status of his credential.
4. A Regulator Stakeholder accesses the 5GZORRO Portal.
5. The Regulator navigates to his personal interface showing all License Credentials pending for approval.
6. The Regulator approves or refuses the License Credential.

By interacting with the LPR, the Portal supports the creation and retrieval of legal prose templates. The workflow for these functionalities can be described as follows:

1. A stakeholder accesses the 5GZORRO Portal and submits a new legal prose template.
2. The request is sent to LPR module.
3. The stakeholder navigates to his personal interface and retrieves all the templates created.
4. The stakeholder can see the details of each template directly from the Portal.

2.4.2. Prototype Implementation

As a component of 5GOZRRO's web GUI, the Governance Portal is a Frontend application written in JavaScript using ReactJS framework. Additionally, it uses the CoreUI react dashboard template library. As a Frontend web application, it provides the structure, appearance, behaviour, and content of everything that appears on browser displays when 5GZORRO portal/website is accessed. All the data and functionality are provided indirectly from the backstage 5GZORRO components' APIs with which the Portal interacts.

The deployment of the Portal consists of three separate nginx images, each one with the files for one specific deploy instance (operator-a, operator-b, operator-c, or regulator-a).

The source code and instructions on how to deploy can be found on 5GZORRO Github page, using the reference [20]. Deployment files related to k8s deployments are also available.

2.5. Validation Tests

This subsection reports the tests performed to validate the different operations supported by the Governance applications. For each software prototype, we first document the set of functional tests executed per component, which are in line with the supported operations declared in D3.3 [2], and then we provide some usage scenarios illustrating the joint operation of the components to handle the considered requests from the user-facing Governance Portal.

2.5.1. Functional Tests

Table 2-1 lists the set of tests performed to validate the functionalities implemented by the ID&P's Authentication API. These tests are the result of the integration work done primarily with the VPN-as-a-Service component (see D4.3 [21]). The tests reported were performed by the VPN-as-a-Service, where it would query an Id&P Agent to receive and validate the obtained key-pairing against other Agents, in order to generate a secure tunnel.

Table 2-1: ID&P Authentication functional test set

Name	Description	Passed (Yes/No/Partially)
Operator Key Pair Generation	Generation of encoded operator key pairing for secure channel establishment purposes	Yes
Operator Key Pair Verification	Verification of operator key pairing issued by another DID Agent to enable secure channel establishment	Yes

Table 2-2 lists the set of tests performed to validate the functionalities implemented by the ID&P's Holder API. Some of these tests correspond to the result of the integration work done with the Governance Portal and the SCLCM components. The tests reported have been performed in the Governance Portal Trader interfaces, validating the interaction with ID&P for operations related to credential requests, and in the SCLCM, to validate the DID generation.

Table 2-2: ID&P Holder functional test set

Name	Description	Passed (Yes/No/Partially)
Stakeholder Membership Credential request	Request creation of a 5GZORRO Stakeholder Membership Credential to be resolved by a platform Administrator	Yes
Check Stakeholder Membership appliance status	Overview the current status of Membership appliances done by a Trader	Yes
Check Stakeholder Membership by DID	Overview the current status of a specific Membership appliance according to the provided DID	Yes
Stakeholder License Credential request	Request to create a 5GZORRO Stakeholder License Credential to be resolved by a platform Regulator	Yes
Check all Stakeholder Licenses	Overview all the issued, declined and requested Stakeholder Licenses done by a Trader	Yes
Generate a DID	Create a 5GZORRO DID to be used to identify SLAs, Legal Prose Templates, etc.	Yes

Check specific DID status	Retrieve a specific DID with an issued status	Yes
List DIDs	Lists all or specific agent's unique DID on the wallet	Yes

Table 2-3 lists the set of tests performed to validate the functionalities implemented by the ID&P's Administrator API. The provided tests correspond to the result of the integration work done with the Governance Portal component. The tests reported have been performed in the Governance Portal Admin interfaces, validating the interaction with ID&P for operations related to Membership credentials management.

Table 2-3: ID&P Admin functional test set

Name	Description	Passed (Yes/No/Partially)
Approve Stakeholder Membership Credential request	Approves and issues a Stakeholder Membership Credential requested by a Holder/Regulator Agent	Yes
Decline Stakeholder Membership Credential request	Rejects a Stakeholder Membership Credential requested by a Holder/Regulator Agent	Yes
List approved Stakeholder Membership Credentials	Retrieves all the Stakeholder Membership requests issued by the Admin Agent	Yes
List declined Stakeholder Membership Credentials	Retrieves all the Stakeholder Membership requests declined by the Admin Agent	Yes
List pending Stakeholder Membership Credentials	Retrieves all the Stakeholder Membership pending requests to be attended by the Admin Agent	Yes
Revoke Stakeholder Membership	Revoke an active Stakeholder Membership Credential of a Trader/Regulator in the platform	Yes

Table 2-4 lists the set of tests performed to validate the functionalities implemented by the ID&P's Regulator API. The tests reported in the table correspond to the result of the integration work done with the Governance Portal component. The tests reported have been performed in the Governance Portal Regulator interfaces, validating the interaction with ID&P for operations related to License credentials management.

Table 2-4: ID&P Regulator functional test set

Name	Description	Passed (Yes/No/Partially)

Approve Stakeholder License Credential request	Approves and issues a Stakeholder License Credential requested by a Holder Agent	Yes
Decline Stakeholder License Credential request	Rejects a Stakeholder License Credential requested by a Holder Agent	Yes
List approved Stakeholder License Credentials	Retrieves all the Stakeholder License requests issued by the Regulator Agent	Yes
List declined Stakeholder License Credentials	Retrieves all the Stakeholder License requests declined by the Regulator Agent	Yes
List pending Stakeholder License Credentials	Retrieves all the Stakeholder License pending requests to be attended by the Regulator Agent	Yes

Table 2-5 lists the set of tests performed to validate the functionalities implemented by the LPR. The tests reported have been performed against the LPR NBI and include also the integration with the ID&P, directly queried by the LPR during the creation of a new template to request a new DID.

Table 2-5: LPR functional test set

Name	**Description**	**Passed (Yes/No/Partially)**
New Template Creation	Creation of SLA and eLicensing templates through the REST NBI. This test implies the interaction the ID&P to request and obtain a new DID per each template.	Yes
Template List Retrieval	Retrieval of existing Template List through the NBI	Yes
Specific Template Retrieval	Retrieval of specific template given a DID	Yes
Template Delete	Removal of an existing template from the internal storage by its own DID	Yes

Table 2-6 lists the set of tests performed to validate the functionalities implemented by the Portal related to Governance. The tests reported have been performed through the Portal interface and validate also the integration when interacting with ID&P (for operations related to stakeholder and credentials management) and with LPR (for operations related to templates management).

Table 2-6: Portal functional test set related to Governance

Name	**Description**	**Passed (Yes/No/Partially)**

Register a stakeholder	Request a Stakeholder Membership Credential to be resolved by a platform Administrator and to have access to 5GZORRO portal	Yes
Check Stakeholder Membership appliance status	Receive the current status of Membership Credential request when trying to access the Portal with provided DID	Yes
Check all Stakeholder Membership Credentials	Overview all issued, declined, revoked and requested/pending Stakeholder Membership Credentials from the Administrator private portal interface	Yes
Approve Stakeholder Membership Credential request	Approve a Stakeholder Membership Credential requested by a Holder/Regulator Agent from the Administrator private Portal interface	Yes
Decline Stakeholder Membership Credential request	Reject a Stakeholder Membership Credential requested by a Holder/Regulator Agent from the Administrator private Portal interface	Yes
Revoke Stakeholder Membership	Revoke an active Stakeholder Membership Credential of a Trader/Regulator from the Administrator private Portal interface	Yes
Request Stakeholder License Credential	Request the creation of a Stakeholder License Credential from the Trader Portal interface to be resolved by a platform Regulator	Yes
Check Stakeholder License Credential status	Overview the status of a Stakeholder License Credential request from the Trader Portal interface	Yes
Check all Stakeholder License Credential	Overview all issued, declined, and requested Stakeholder License Credentials from the Regulator private portal interface	Yes
Approve Stakeholder License Credential request	Approve a Stakeholder License Credential requested by a Holder Agent from the Regulator private portal interface	Yes
Decline Stakeholder License Credential request	Reject a Stakeholder License Credential requested by a Holder Agent from the Regulator private portal interface	Yes
Create a new Legal Template	Submit SLA and eLicensing templates through the Portal	Yes
Check Legal templates	Overview the details of the submitted templates from the Portal	Yes

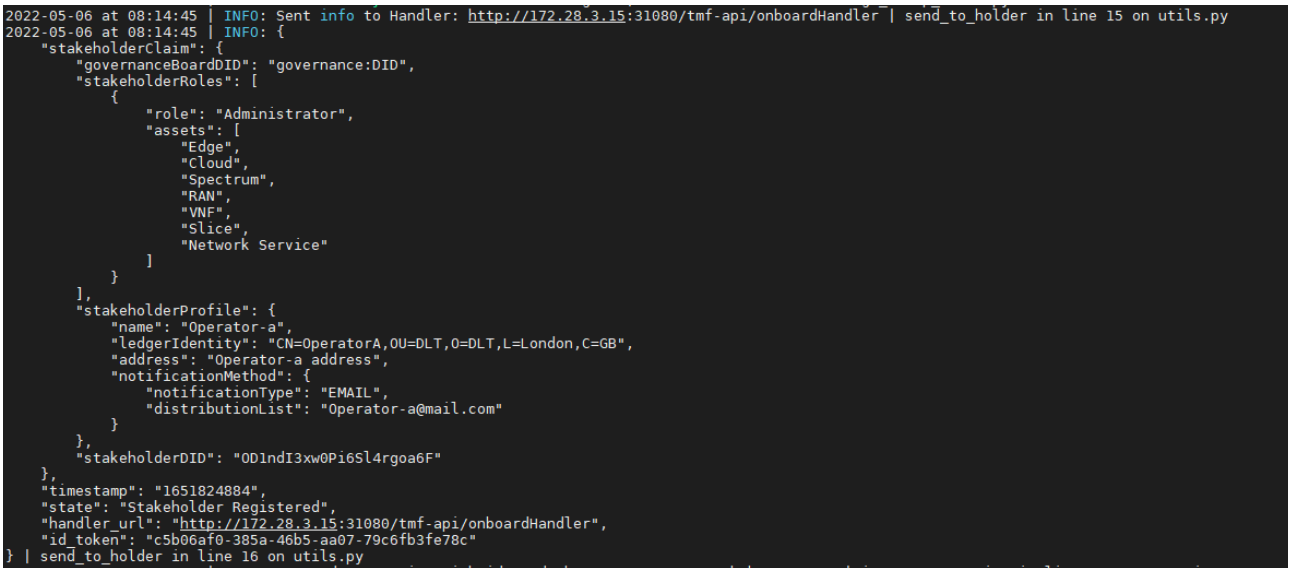
2.5.2. Platform Operations

Next, some usage scenarios of the Governance prototypes, in conjunction with the underlying DLT, are described to validate the system capabilities related to the management of stakeholders, DIDs, membership and license credentials and templates, and primitive spectoken generation. This report serves not only to demonstrate the attainment of expected capabilities, but also to illustrate the usage of the platform through the 5GZORRO Portal.

2.5.2.1. Admin bootstrapping

As part of the 5GZORRO stakeholder management process, an Administrative-type stakeholder is required to overview the incoming requests of new partners that intent to join the decentralized marketplace. As such, for the implemented prototype, the envisioned Operator-A will be tasked as being the Administrator Agent of the platform.

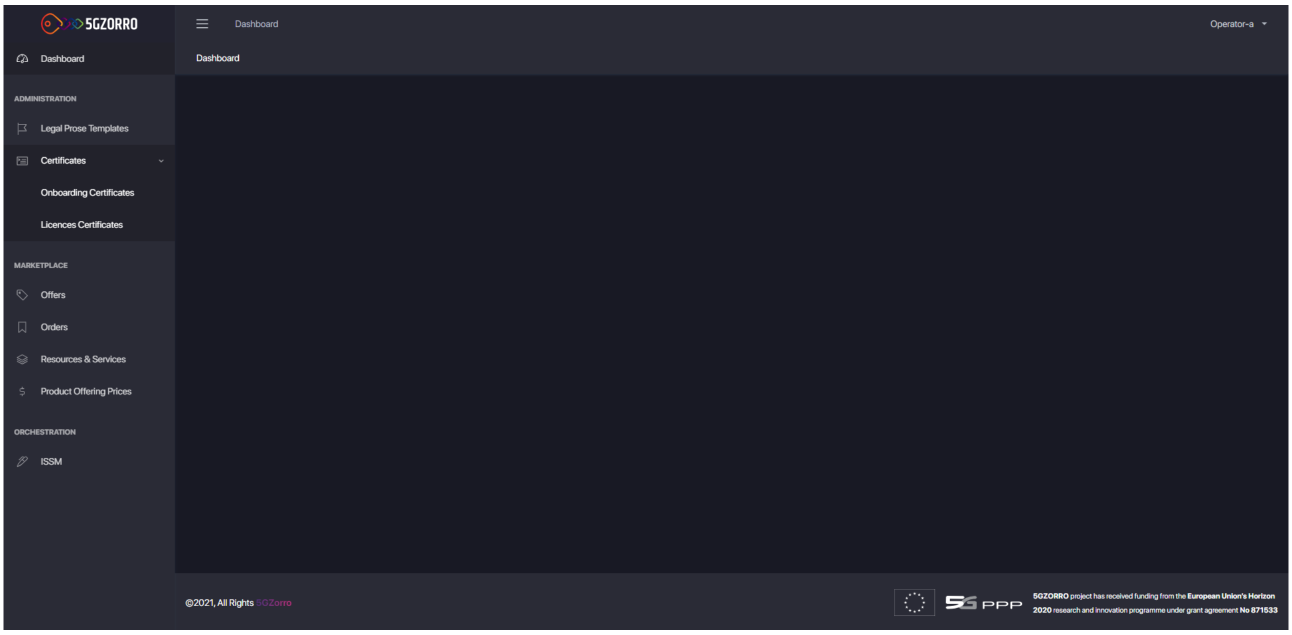
During the deployment of Operator-A's ID&P Agent instance, the aforementioned will generate a unique and exclusive Stakeholder Membership Credential for Administrators and synchronize this information with its respective TMF Catalogue. Figure 2-13 showcases the information of the Credential that is passed to the RSOC API.



```
2022-05-06 at 08:14:45 | INFO: Sent info to Handler: http://172.28.3.15:31080/tmf-api/onboardHandler | send_to_holder in line 15 on utils.py
2022-05-06 at 08:14:45 | INFO: {
    "stakeholderClaim": {
        "governanceBoardDID": "governance:DID",
        "stakeholderRoles": [
            {
                "role": "Administrator",
                "assets": [
                    "Edge",
                    "Cloud",
                    "Spectrum",
                    "RAN",
                    "VNF",
                    "Slice",
                    "Network Service"
                ]
            }
        ],
        "stakeholderProfile": {
            "name": "Operator-a",
            "ledgerIdentity": "CN=OperatorA,OU=DLT,O=DLT,L=London,C=GB",
            "address": "Operator-a address",
            "notificationMethod": {
                "notificationType": "EMAIL",
                "distributionList": "Operator-a@mail.com"
            }
        },
        "stakeholderDID": "OD1ndI3xw0Pi6Sl4rgoa6F"
    },
    "timestamp": "1651824884",
    "state": "Stakeholder Registered",
    "handler_url": "http://172.28.3.15:31080/tmf-api/onboardHandler",
    "id_token": "c5b06af0-385a-46b5-aa07-79c6fb3fe78c"
} | send_to_holder in line 16 on utils.py
```

Figure 2-13: ID&P Admin instance communicating with RSOC API during bootstrap

Upon this process completion, the Administrator of the Marketplace can then access its unique Governance Portal interface, by using the attributed *stakeholderDID* of the credential. Figure 2-14 represents the initial page that is presented to, in this case, Operator-A, upon login. As a Platform Administrator, Operator-A has access to unique views to manage the different types of Certificates issued to other stakeholders in the Marketplace.



©2021, All Rights 5GZorro

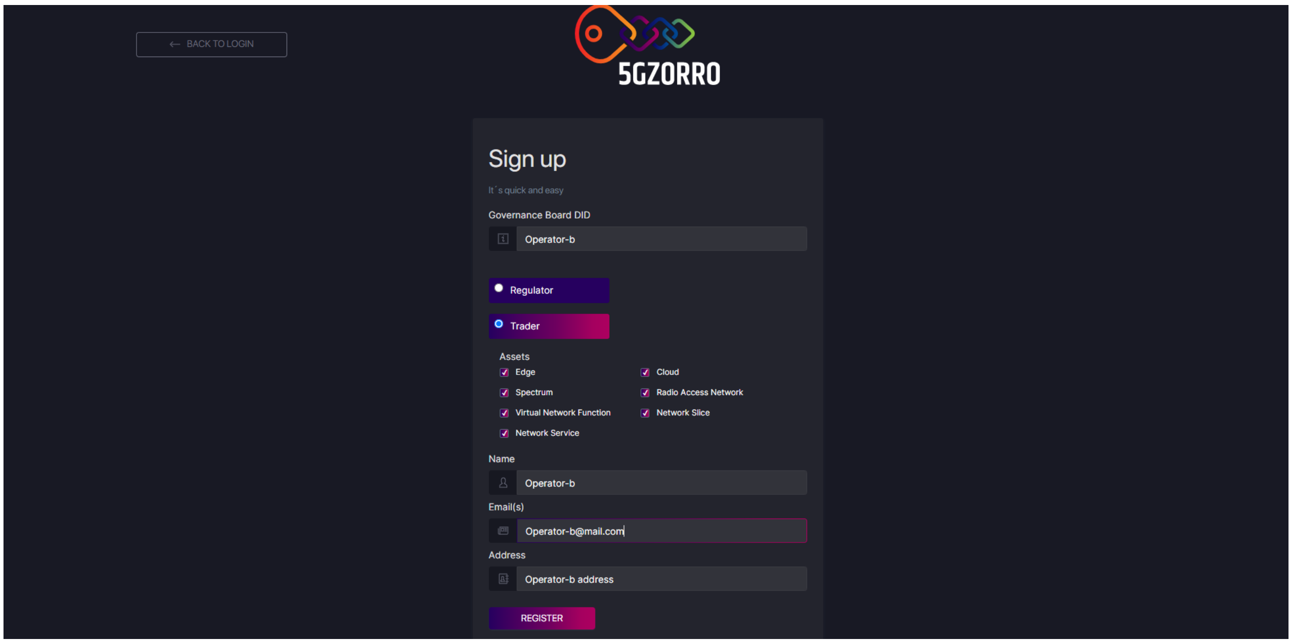
5GZORRO project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871533

Figure 2-14: Operator-A's Governance Portal interface after successful login

2.5.2.2. *Onboarding of new trading or regulator stakeholder*

When intending to access the 5GZORRO Marketplace, a potential Trader-type stakeholder must request a Stakeholder Membership Credential, to be granted the chance of performing business with other registered partners. Since, for the case of, i.e., Spectrum assets, there is a necessity for a regulatory entity to overview the kind of Licenses Traders claim to possess, Regulator-A is deployed for those specific scenarios. Both Trader-type (Operators B & C) and Regulator-type (Regulator-A) Agents are subject to the approval process partaken by and Administrative Agent.

Any Trader or Regulator can request an Onboarding Certificate to be attributed to them by accessing the Governance Portal interface for registration. Figure 2-15 details said interface with the required information that the stakeholder must fulfil in order to proceed.



Sign up

It´s quick and easy

Governance Board DID: Operator-b

Regulator [✓ Trader]

Assets
- ✓ Edge
- ✓ Cloud
- ✓ Spectrum
- ✓ Radio Access Network
- ✓ Virtual Network Function
- ✓ Network Slice
- ✓ Network Service

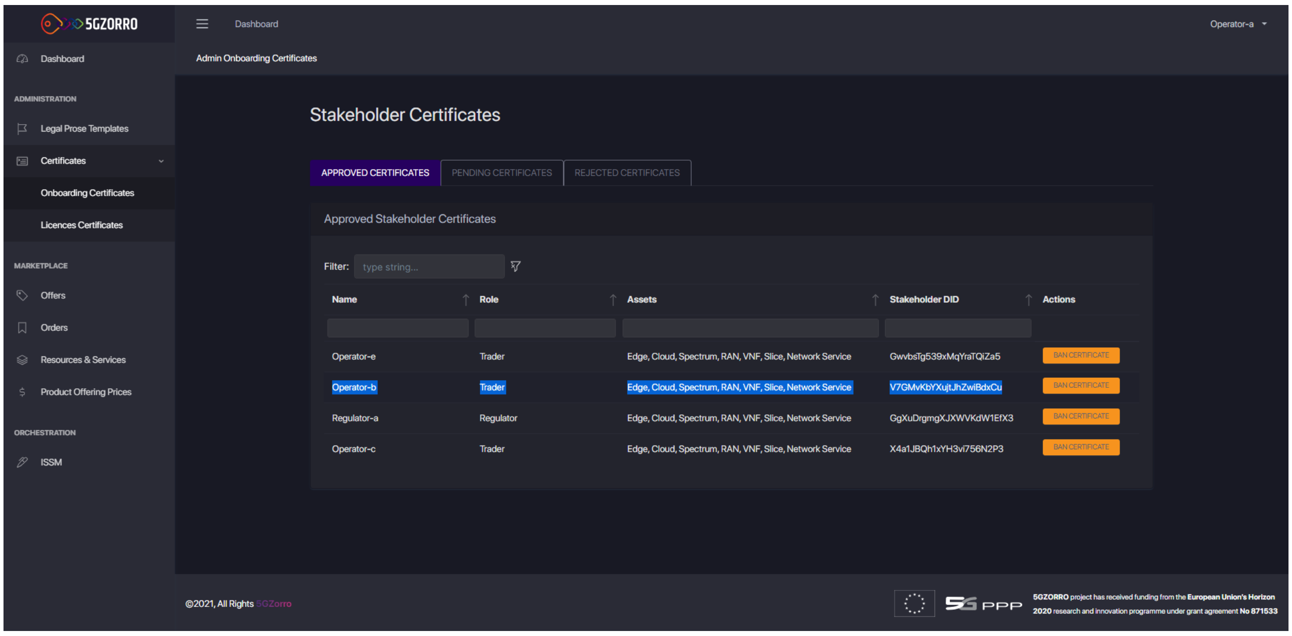
Name: Operator-b

Email(s): Operator-b@mail.com

Address: Operator-b address

Figure 2-15: Operator-B's Governance Portal interface for Membership registration w/ filled info

Upon filling the information & submitting it, the request will be appended with a unique *stakeholderDID*, which the stakeholder must save in order to use it on the login page. If it tries to login straight away, since it is still unresolved, the Portal will validate and display a message appropriately. On the other hand, the Administrator can decide on how to proceed, approving or rejecting the appliance, through its own designated Governance Portal, by checking the 'Pending Certificates' interface. Figure 2-16 displays the Admin's Governance Portal interface of approved credentials.



Stakeholder Certificates

Approved Stakeholder Certificates

Name	Role	Assets	Stakeholder DID	Actions
Operator-e	Trader	Edge, Cloud, Spectrum, RAN, VNF, Slice, Network Service	GwvbsTg539xMqYraTQiZa5	BAN CERTIFICATE
Operator-b	Trader	Edge, Cloud, Spectrum, RAN, VNF, Slice, Network Service	V7GMvKbYXujtJhZwiBdxCu	BAN CERTIFICATE
Regulator-a	Regulator	Edge, Cloud, Spectrum, RAN, VNF, Slice, Network Service	GgXuDrgmgXJXWVKdW1EfX3	BAN CERTIFICATE
Operator-c	Trader	Edge, Cloud, Spectrum, RAN, VNF, Slice, Network Service	X4a1JBQh1xYH3vi756N2P3	BAN CERTIFICATE

Figure 2-16: Operator-A's Governance Portal interface for 'Approved Certificates' w/ highlighted Operator-B info

If the Administrator decided to approve, as it's possible to see above, Operator-B can now use its *stakeholderDID* to login, and by doing so, it has now access to the 5GZORRO Marketplace. Figure 2-17 represents the interface Operator-B will enter, upon login with the highlighted *stakeholderDID* on the previously shown Figure 2-16.

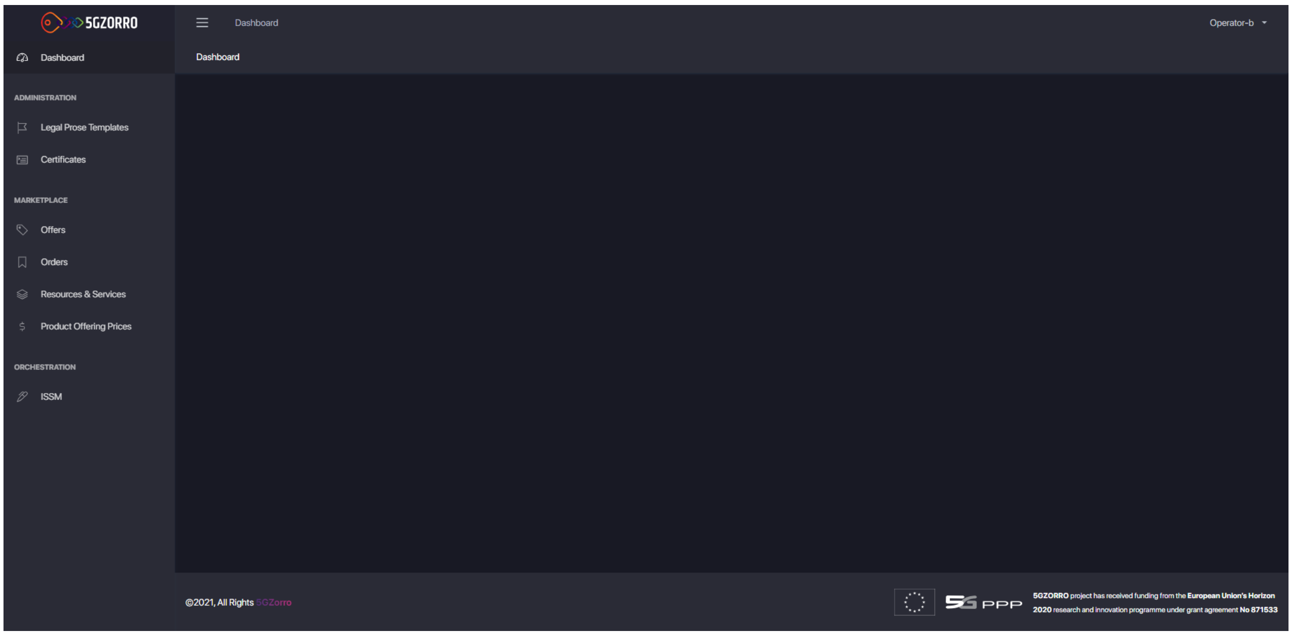
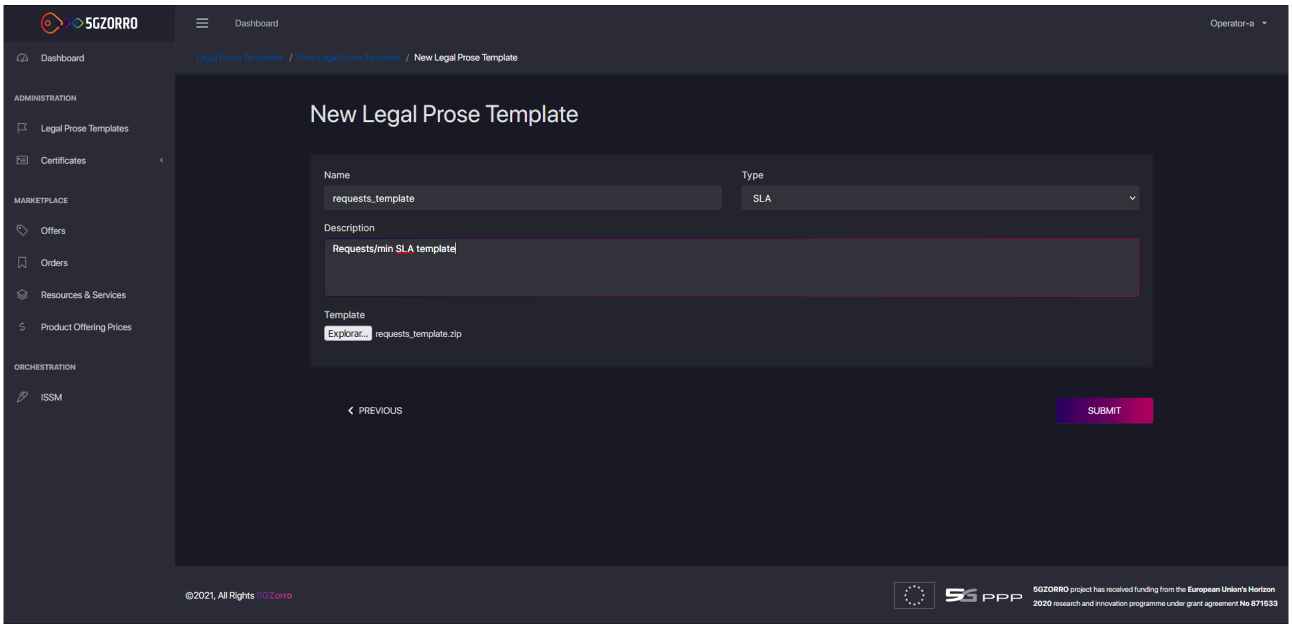

5GZORRO
Dashboard
Operator-b
Dashboard
ADMINISTRATION
Legal Prose Templates
Certificates
MARKETPLACE
Offers
Orders
Resources & Services
Product Offering Prices
ORCHESTRATION
ISSM
©2021, All Rights 5GZorro
5GZORRO project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871533

Figure 2-17: Operator-B's Governance Portal interface after successful login

2.5.2.3. Onboarding of SLA and License Templates

If a Trader wants to submit a new Legal Prose Template, she/he can access the Legal Prose Templates view in the Portal. Upon filling the required information and submitting the template, a request is sent to LPR module (Figure 2-18). If the submission is successful, the template is made available in the Portal.



Legal Prose Templates / New Legal Prose Template / New Legal Prose Template

New Legal Prose Template

Name: requests_template

Type: SLA

Description: Requests/min SLA template

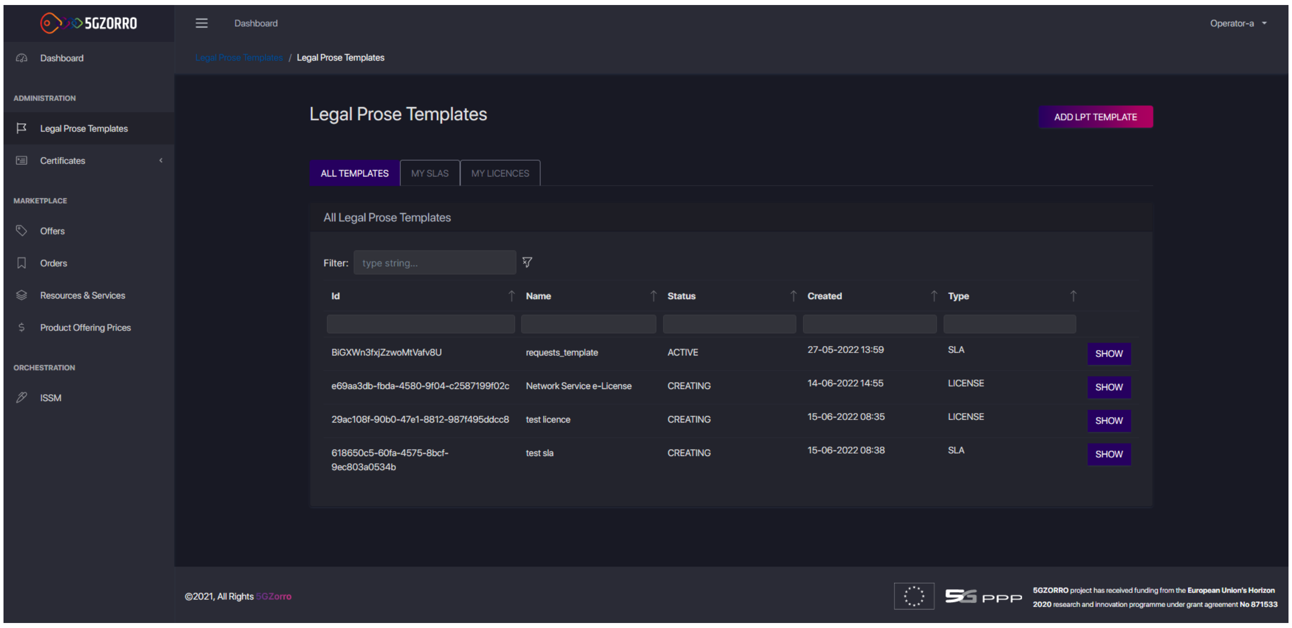
Template: requests_template.zip

©2021, All Rights 5GZorro

5GZORRO project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871533

Figure 2-18: Submission of a new Legal Prose Template

The stakeholder can see the details of each template directly from the Portal. A Trader stakeholder can access the Legal Prose Templates portal interface where all the submitted templates are displayed (Figure 2-19). From this view it is possible to check the details of a specific template and read the templated document. Figure 2-20 shows a sample legal prose template for SLA, Figure 2-21 shows a sample legal prose template for e-License.



Legal Prose Templates / Legal Prose Templates

Legal Prose Templates

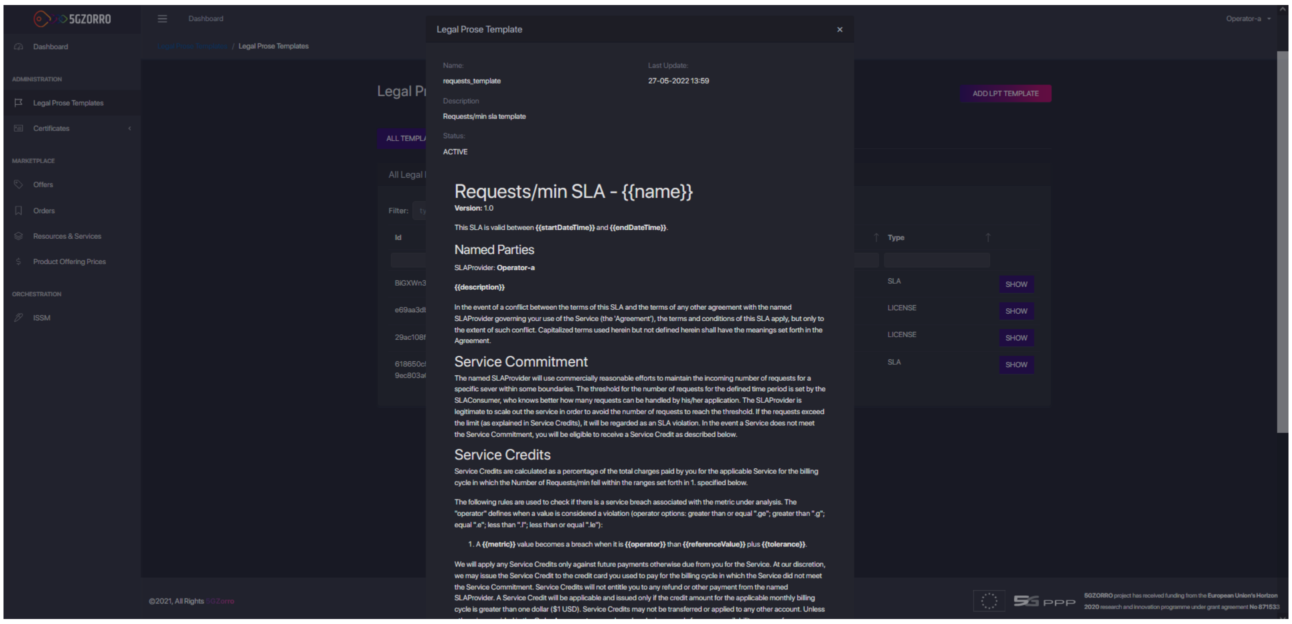
All Legal Prose Templates

Id	Name	Status	Created	Type	
BiGXWn3fxjZzwoMtVafv8U	requests_template	ACTIVE	27-05-2022 13:59	SLA	SHOW
e69aa3db-fbda-4580-9f04-c2587199f02c	Network Service e-License	CREATING	14-06-2022 14:55	LICENSE	SHOW
29ac108f-90b0-47e1-8812-987f495ddcc8	test licence	CREATING	15-06-2022 08:35	LICENSE	SHOW
618650c5-60fa-4575-8bcf-9ec803a0534b	test sla	CREATING	15-06-2022 08:38	SLA	SHOW

©2021, All Rights 5GZorro

5GZORRO project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871533

Figure 2-19: Overview of all the submitted Legal Prose Templates



Legal Prose Template

Name: requests_template

Last Update: 27-05-2022 13:59

Description: Requests/min sla template

Status: ACTIVE

Requests/min SLA - {{name}}

Version: 1.0

This SLA is valid between **{{startDateTime}}** and **{{endDateTime}}**.

Named Parties

SLAProvider: **Operator-a**

{{description}}

In the event of a conflict between the terms of this SLA and the terms of any other agreement with the named SLAProvider governing your use of the Service (the 'Agreement'), the terms and conditions of this SLA apply, but only to the extent of such conflict. Capitalized terms used herein but not defined herein shall have the meanings set forth in the Agreement.

Service Commitment

The named SLAProvider will use commercially reasonable efforts to maintain the incoming number of requests for a specific sever within some boundaries. The threshold for the number of requests for the defined time period is set by the SLAConsumer, who knows better how many requests can be handled by his/her application. The SLAProvider is legitimate to scale out the service in order to avoid the number of requests to reach the threshold. If the requests exceed the limit (as explained in Service Credits), it will be regarded as an SLA violation. In the event a Service does not meet the Service Commitment, you will be eligible to receive a Service Credit as described below.

Service Credits

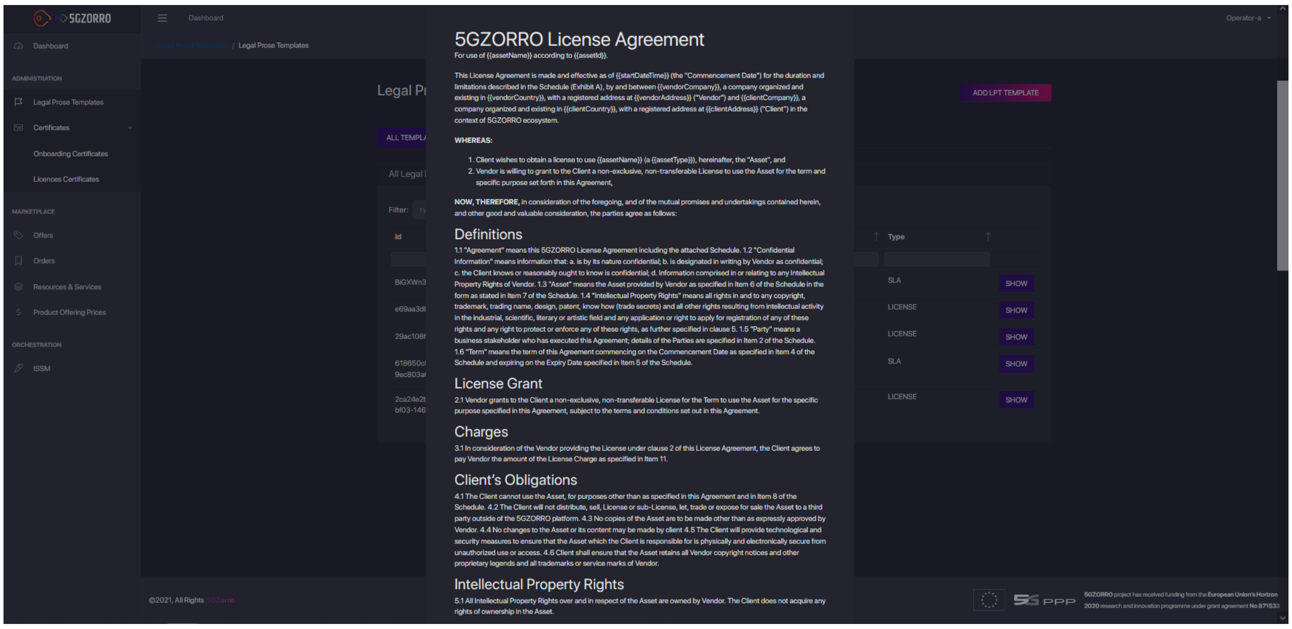
Service Credits are calculated as a percentage of the total charges paid by you for the applicable Service for the billing cycle in which the Number of Requests/min fell within the ranges set forth in 1. specified below.

The following rules are used to check if there is a service breach associated with the metric under analysis. The "operator" defines when a value is considered a violation (operator options: greater than or equal ".ge"; greater than ".g"; equal ".e"; less than ".l"; less than or equal ".le"):

1. A **{{metric}}** value becomes a breach when it is **{{operator}}** than **{{referenceValue}}** plus **{{tolerance}}**.

We will apply any Service Credits only against future payments otherwise due from you for the Service. At our discretion, we may issue the Service Credit to the credit card you used to pay for the billing cycle in which the Service did not meet the Service Commitment. Service Credits will not entitle you to any refund or other payment from the named SLAProvider. A Service Credit will be applicable and issued only if the credit amount for the applicable monthly billing cycle is greater than one dollar ($1 USD). Service Credits may not be transferred or applied to any other account. Unless

Figure 2-20: Details of an SLA Legal Prose Template



5GZORRO License Agreement

For use of {{assetName}} according to {{assetId}}.

This License Agreement is made and effective as of {{startDateTime}} (the "Commencement Date") for the duration and limitations described in the Schedule (Exhibit A), by and between {{vendorCompany}}, a company organized and existing in {{vendorCountry}}, with a registered address at {{vendorAddress}} ("Vendor") and {{clientCompany}}, a company organized and existing in {{clientCountry}}, with a registered address at {{clientAddress}} ("Client") in the context of 5GZORRO ecosystem.

WHEREAS:

1. Client wishes to obtain a license to use {{assetName}} (a {{assetType}}), hereinafter, the "Asset", and
2. Vendor is willing to grant to the Client a non-exclusive, non-transferable License to use the Asset for the term and specific purpose set forth in this Agreement,

NOW, THEREFORE, in consideration of the foregoing, and of the mutual promises and undertakings contained herein, and other good and valuable consideration, the parties agree as follows:

Definitions

1.1 "Agreement" means this 5GZORRO License Agreement including the attached Schedule. 1.2 "Confidential Information" means information that: a. is by its nature confidential; b. is designated in writing by Vendor as confidential; c. the Client knows or reasonably ought to know is confidential; d. Information comprised in or relating to any Intellectual Property Rights of Vendor. 1.3 "Asset" means the Asset provided by Vendor as specified in Item 6 of the Schedule in the form as stated in Item 7 of the Schedule. 1.4 "Intellectual Property Rights" means all rights in and to any copyright, trademark, trading name, design, patent, know how (trade secrets) and all other rights resulting from intellectual activity in the industrial, scientific, literary or artistic field and any application or right to apply for registration of any of these rights and any right to protect or enforce any of these rights, as further specified in clause 5. 1.5 "Party" means a business stakeholder who has executed this Agreement; details of the Parties are specified in Item 2 of the Schedule. 1.6 "Term" means the term of this Agreement commencing on the Commencement Date as specified in Item 4 of the Schedule and expiring on the Expiry Date specified in Item 5 of the Schedule.

License Grant

2.1 Vendor grants to the Client a non-exclusive, non-transferable License for the Term to use the Asset for the specific purpose specified in this Agreement, subject to the terms and conditions set out in this Agreement.

Charges

3.1 In consideration of the Vendor providing the License under clause 2 of this License Agreement, the Client agrees to pay Vendor the amount of the License Charge as specified in Item 11.

Client's Obligations

4.1 The Client cannot use the Asset, for purposes other than as specified in this Agreement and in Item 8 of the Schedule. 4.2 The Client will not distribute, sell, License or sub-License, let, trade or expose for sale the Asset to a third party outside of the 5GZORRO platform. 4.3 No copies of the Asset are to be made other than as expressly approved by Vendor. 4.4 No changes to the Asset or its content may be made by client 4.5 The Client will provide technological and security measures to ensure that the Asset which the Client is responsible for is physically and electronically secure from unauthorized use or access. 4.6 Client shall ensure that the Asset retains all Vendor copyright notices and other proprietary legends and all trademarks or service marks of Vendor.

Intellectual Property Rights

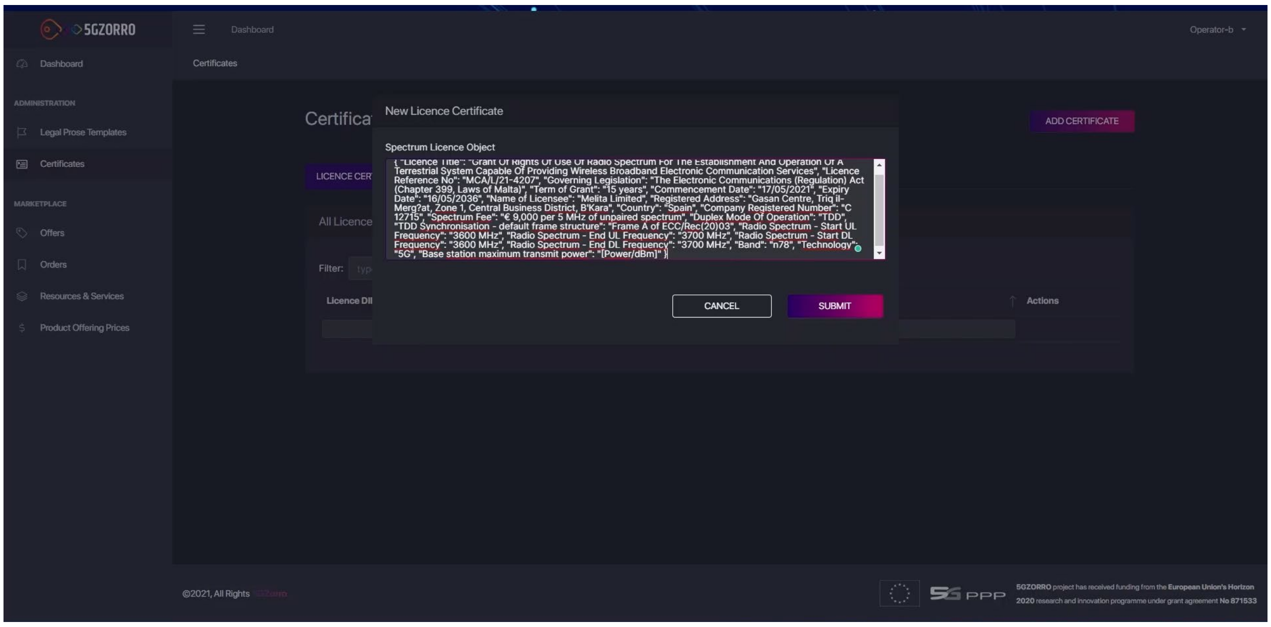
5.1 All Intellectual Property Rights over and in respect of the Asset are owned by Vendor. The Client does not acquire any rights of ownership in the Asset.

Figure 2-21: Details of an e-License Legal Prose Template

2.5.2.4. Spectrum license approval

To allow a given spectrum resource provider to create offers, it needs the explicit authorisation from the Regulator for one (or many) of the spectrum resource provider's spectrum licenses.

This authorisation process starts with the spectrum resource provider sending a request to create a spectrum certificate via the Portal, as it can be seen in Figure 2-22. The spectrum resource provider (Operator-b in Figure 2-22) needs to provide the technical information of a valid spectrum license.

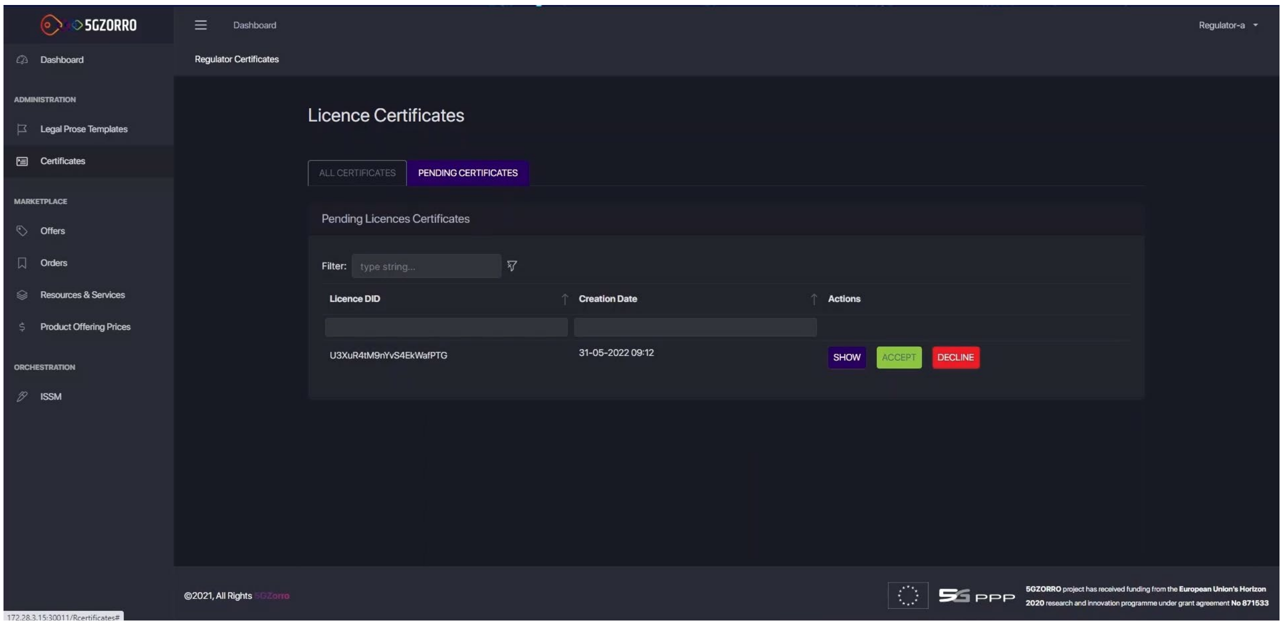


Spectrum Licence Object

```
{ "Licence Title": "Grant Of Rights Of Use Of Radio Spectrum For The Establishment And Operation Of A Terrestrial System Capable Of Providing Wireless Broadband Electronic Communication Services", "Licence Reference No": "MCA/L/21-4207", "Governing Legislation": "The Electronic Communications (Regulation) Act (Chapter 399, Laws of Malta)", "Term of Grant": "15 years", "Commencement Date": "17/05/2021", "Expiry Date": "16/05/2036", "Name of Licensee": "Melita Limited", "Registered Address": "Gasan Centre, Triq il-Merg?at, Zone 1, Central Business District, B'Kara", "Country": "Spain", "Company Registered Number": "C 12715", "Spectrum Fee": "€ 9,000 per 5 MHz of unpaired spectrum", "Duplex Mode Of Operation": "TDD", "TDD Synchronisation - default frame structure": "Frame A of ECC/Rec(20)03", "Radio Spectrum - Start UL Frequency": "3600 MHz", "Radio Spectrum - End UL Frequency": "3700 MHz", "Radio Spectrum - Start DL Frequency": "3600 MHz", "Radio Spectrum - End DL Frequency": "3700 MHz", "Band": "n78", "Technology": "5G", "Base station maximum transmit power": "[Power/dBm]" }
```

Figure 2-22: Spectrum resource provider sends a spectrum certificate request containing the technical details of the spectrum license

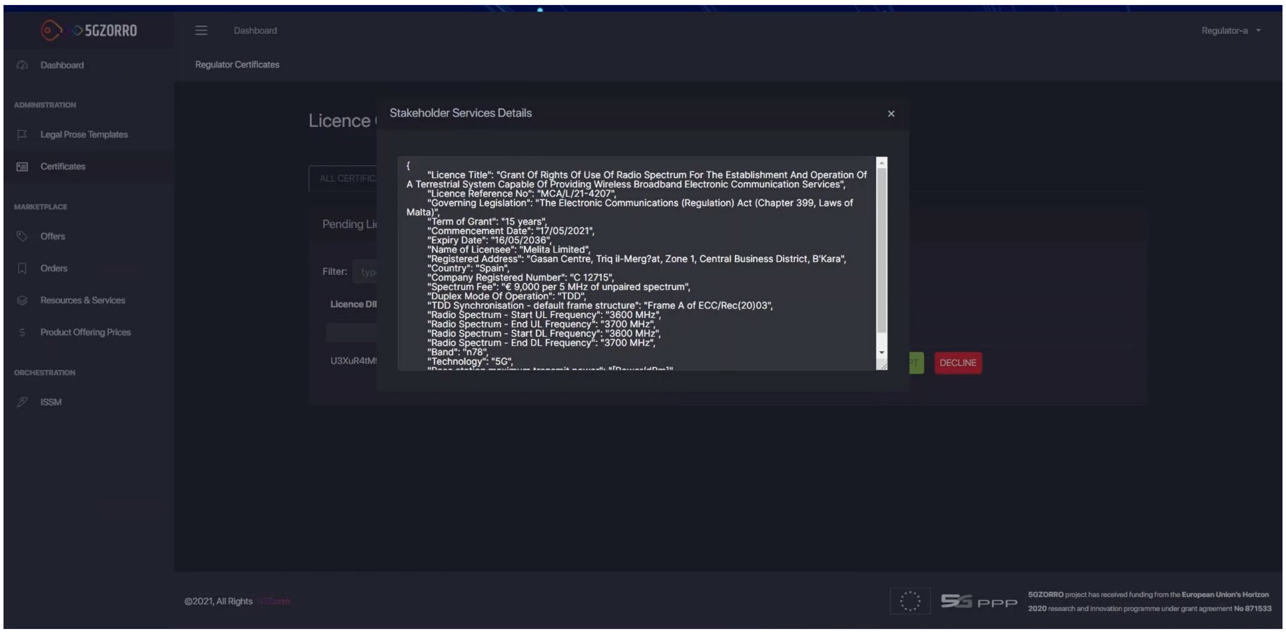
Afterwards, the Regulator can access to the list of pending requests as shown in Figure 2-23 and proceed with their approval or rejection.



Licence DID	Creation Date	Actions
U3XuR4tM9nYvS4EkWafPTG	31-05-2022 09:12	SHOW ACCEPT DECLINE

Figure 2-23: Regulator can list all pending spectrum certificate requests, show their technical information and based on this, decide to accept or decline the request

Acceptance or rejection decision is based on a manual inspection of the information with the objective of comparing the technical information sent in the request (see Figure 2-24) with the real spectrum license that the Regulator approved beforehand.



```
{
        "Licence Title": "Grant Of Rights Of Use Of Radio Spectrum For The Establishment And Operation Of
A Terrestrial System Capable Of Providing Wireless Broadband Electronic Communication Services",
        "Licence Reference No": "MCA/L/21-4207",
        "Governing Legislation": "The Electronic Communications (Regulation) Act (Chapter 399, Laws of
Malta)",
        "Term of Grant": "15 years",
        "Commencement Date": "17/05/2021",
        "Expiry Date": "16/05/2036",
        "Name of Licensee": "Melita Limited",
        "Registered Address": "Gasan Centre, Triq il-Merg?at, Zone 1, Central Business District, B'Kara",
        "Country": "Spain",
        "Company Registered Number": "C 12715",
        "Spectrum Fee": "€ 9,000 per 5 MHz of unpaired spectrum",
        "Duplex Mode Of Operation": "TDD",
        "TDD Synchronisation - default frame structure": "Frame A of ECC/Rec(20)03",
        "Radio Spectrum - Start UL Frequency": "3600 MHz",
        "Radio Spectrum - End UL Frequency": "3700 MHz",
        "Radio Spectrum - Start DL Frequency": "3600 MHz",
        "Radio Spectrum - End DL Frequency": "3700 MHz",
        "Band": "n78",
        "Technology": "5G",
        "Base station maximum transmit power": "[Power/dBm]"
```

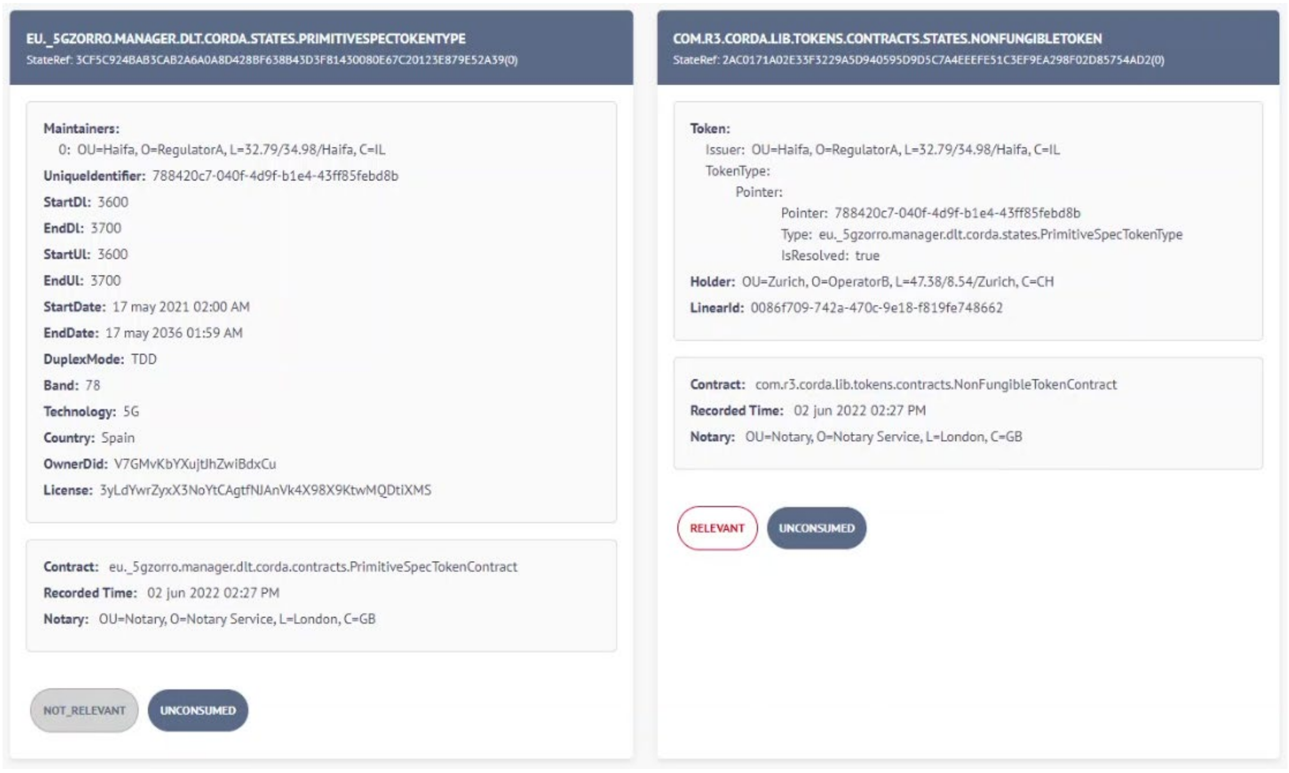
Figure 2-24: View of the technical information included in the spectrum certificate request to be reviewed by the Regulator

The Regulator, after inspecting the information in the request and comparing it with spectrum resource provider's spectrum licenses, decides to approve or reject the request. Independently of the decision, the certificate is recorded in the Governance DLT by the ID&P. The only difference is that the certificate is set as valid or not valid, depending on if the request was accepted or rejected, respectively.

2.5.2.5. Generation of primitive spectoken

After the acceptance of a spectrum certificate request, the Regulator issues a Primitive Spectoken for the spectrum resource provider. This operation is handled internally by the joint interaction of the Portal with the SCLCM, a module that is further detailed in Section 3.3.

Figure 2-25 illustrates the Primitive Spectoken information that can be retrieved from the holder's node (i.e., the spectrum resource provider) belonging to the Marketplace DLT. On the left side, the technical information of the Primitive SpectokenType is showed, including starting and ending frequencies, duplex mode, start and expiration dates of the license, radio access technology bounded to the band, and country. Moreover, specific information to this SpectokenType is also shown, such as its unique ID, the maintainer Marketplace DLT node related to it (i.e., RegulatorA), and the contract ID in the Marketplace DLT related to the Primitive SpectokenType. On the right side, the resulting Non-Fungible Token (NFT) for the Primitive Spectoken is showcased, containing in addition to the reference to the SpectokenType it is derived from, the ledger identity of the Marketplace DLT node acting as holder (i.e., OperatorB).



EU._5GZORRO.MANAGER.DLT.CORDA.STATES.PRIMITIVESPECTOKENTYPE
StateRef: 3CF5C924BAB3CAB2A6A0A8D428BF638B43D3F81430080E67C20123E879E52A39(0)

Maintainers:
0: OU=Haifa, O=RegulatorA, L=32.79/34.98/Haifa, C=IL
UniqueIdentifier: 788420c7-040f-4d9f-b1e4-43ff85febd8b
StartDl: 3600
EndDl: 3700
StartUl: 3600
EndUl: 3700
StartDate: 17 may 2021 02:00 AM
EndDate: 17 may 2036 01:59 AM
DuplexMode: TDD
Band: 78
Technology: 5G
Country: Spain
OwnerDid: V7GMvKbYXujtJhZwiBdxCu
License: 3yLdYwrZyxX3NoYtCAgtfNJAnVk4X98X9KtwMQDtiXMS

Contract: eu._5gzorro.manager.dlt.corda.contracts.PrimitiveSpecTokenContract
Recorded Time: 02 jun 2022 02:27 PM
Notary: OU=Notary, O=Notary Service, L=London, C=GB

COM.R3.CORDA.LIB.TOKENS.CONTRACTS.STATES.NONFUNGIBLETOKEN
StateRef: 2AC0171A02E33F3229A5D940595D9D5C7A4EEEFE51C3EF9EA298F02D85754AD2(0)

Token:
Issuer: OU=Haifa, O=RegulatorA, L=32.79/34.98/Haifa, C=IL
TokenType:
Pointer:
Pointer: 788420c7-040f-4d9f-b1e4-43ff85febd8b
Type: eu._5gzorro.manager.dlt.corda.states.PrimitiveSpecTokenType
IsResolved: true
Holder: OU=Zurich, O=OperatorB, L=47.38/8.54/Zurich, C=CH
LinearId: 0086f709-742a-470c-9e18-f819fe748662

Contract: com.r3.corda.lib.tokens.contracts.NonFungibleTokenContract
Recorded Time: 02 jun 2022 02:27 PM
Notary: OU=Notary, O=Notary Service, L=London, C=GB

Figure 2-25: Primitive Spectoken information from the Marketplace DLT vault

3. Marketplace Platform Prototypes

The 5GZORRO Marketplace Platform is composed of a set of software modules that provide core trustworthy marketplace services, either to 5GZORRO stakeholders or to other 5GZORRO platforms. Such modules are deployed in every stakeholder domain, representing a party node of the distributed Marketplace platform. Overall, exposed services allow the on-boarding of assets and offers composition; the sharing of offer updates among participants; the on-demand order capture and agreement settlement for offer purchase and consumption; and the continuous SLA management to ensure the fulfilment of agreed conditions.

With the ultimate goal of facilitating the interaction between different stakeholders involved in multi-party service delivery, the 5GZORRO Marketplace Platform enables the trading of heterogeneous types of product offers across multiple operators and resource providers. As anticipated in previous Deliverables, such as D3.1 [1] and D3.3 [2], the product offers that are managed by the 5GZORRO Marketplace Platform represent a variety of exposed telco digital assets. These products can correspond to resource offerings, ranging from infrastructure components, like cloud, edge, and Radio Access Network (RAN), to software assets such as Virtual Network Functions (VNFs) or Cloud Native Functions (CNFs) and spectrum chunks; as well as service offerings, represented by network services and slices.

As illustrated in Figure 3-1, the Marketplace platform logic is ruled by two main software modules, namely the Resource & Service Offer Catalogue (RSOC) and the Smart Contract Lifecycle Manager (SCLCM). To simplify the user interaction with the platform, a dedicated Portal is coupled to the aforementioned backend components exposing all the Marketplace capabilities.

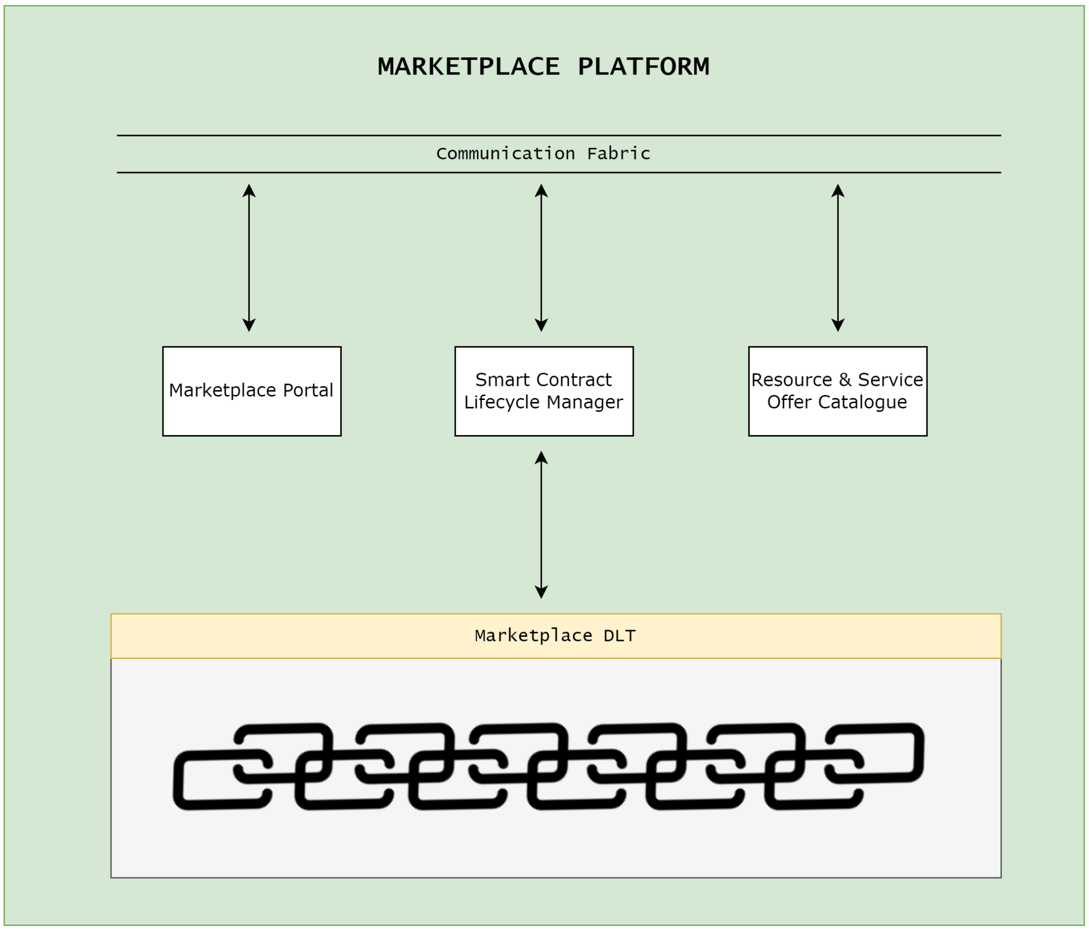


Figure 3-1: Marketplace Platform Architecture

The Marketplace platform is underpinned by an underlying DLT layer, as represented in Figure 3-1, utilised to support the realisation of a decentralized Marketplace of product offers, the subsequent trading of these product offers and the management of associated SLA. Each marketplace stakeholder operates a DLT node and interfaces with it through the hosted marketplace instance.

This section describes the prototypes composing the 5GZORRO Marketplace Platform, including the main aspects for integration with the underlying Marketplace DLT and validation tests performed to corroborate the platform operation and illustrate the platform usage though its user interface (i.e., the Marketplace Portal).

3.1.Marketplace DLT

The R3 Corda [22] DLT is the Blockchain implementation chosen for the 5GZORRO Marketplace Architecture due to its peculiar characteristics:

- Permissioned Blockchain: only authorized stakeholders can access the 5GZORRO Marketplace Network. Each party on the network has a known identity that is used when communicating with counterparties, and network access is controlled by a doorman. This makes the Blockchain ideally suited to offer the privacy required to support agreements between stakeholders in the 5GZORRO Marketplace.
- On-ledger facts are shared only between the parties directly involved in a transaction following a need-to-know basis.

3.1.1. Main Functionalities

The Corda DLT has the purpose of supporting the 5GZORRO Marketplace functionalities, such as:

- Supporting the synchronization of the resource and service offers through the persistence of on-ledger facts among the stakeholders that joined the 5GZORRO Network.
- Persistence of on-ledger facts regarding the buying and selling of 5G-related assets among parties of the 5GZORRO Network (Virtual Network Functions, Network Services, Edge resources, Cloud Resources, Radio resources, Spectrum resources and Network Slices).
- SLAs and Licensing Terms lifecycle management as on-ledger facts, created when an order has been placed and disseminated among the relevant parties of the 5GZORRO DLT Network.
- Spectoken NFT to enable Spectrum trading among the stakeholders of the 5GZORRO Network.

3.1.2. Prototype Implementation

The 5GZORRO Marketplace Corda Network has been envisioned to be deployed in a virtualized environment, in particular, k8s was chosen as target. Each Stakeholder Node of the 5GZORRO Marketplace Network resides in a namespace of a k8s cluster containing all the services needed to join the Corda Network, communicate with the services provided by Corda (i.e., Network Map, Doorman, Notary), and run the CordApp that define the flows defined to accomplish the functionalities of the 5GZORRO Marketplace. Each Stakeholder's Node Namespace contain:

- The Stakeholder's Corda Node: a Java application representing an entity in a Corda network that usually represents one party in a business network. One party operates the node, which contains the CordApps that the party uses to interact with other peers on the network.
- The Stakeholder's Corda Vault: it contains data extracted from the ledger that is considered relevant to the node's owner, stored in a relational model that can be easily queried and worked with.

The 5GZORRO Marketplace Corda Network Services run in a dedicated namespace. To automate the deployment of the whole 5GZORRO Corda Marketplace Network a dedicated tool called Blockchain Automation Framework was used. The Blockchain Automation Framework allows to specify which k8s cluster must be used to deploy a specific Corda Node or Corda Network Service. The Blockchain Automation Framework makes use of an external service called Vault (to not be confused with the Vault of a Corda Node) to store the credentials and certificate of each Corda Node and Corda Service that was deployed. The SCLCM module of the 5GZORRO architecture is the gateway to access the functionalities of the 5GZORRO Marketplace Corda DLT acting as an entry point to start the CordApp workflows that realize the Marketplace DLT functionalities. The SCLCM is a per-domain component, so each stakeholder in addition to its own Corda Node will dispose of its SCLCM component. Figure 3-2 shows the 5GZORRO Marketplace Corda Network as described in the current section.

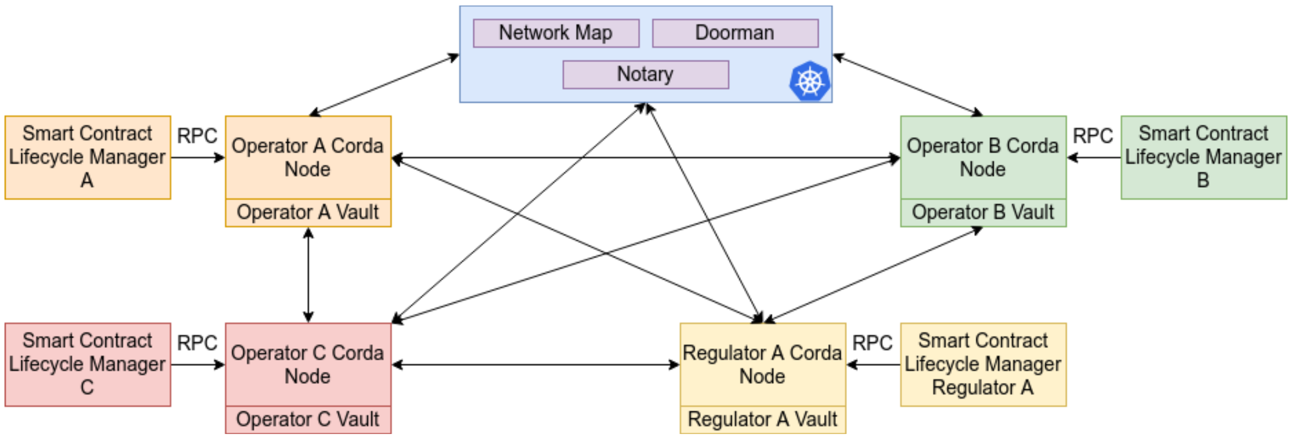


Figure 3-2: 5GZORRO Marketplace Corda Network

3.2. Resource and Service Offer Catalogue

The Resource and Service Offer Catalogue (RSOC) is the module responsible for collecting the 5G assets that are available to be traded among providers and customers. This decentralized portfolio enables the process of registering resources and services, derived from technical specifications, creating, browsing and ordering product offers based on on-boarded resource and service assets. This is done across multiple parties acting as infrastructure providers, spectrum traders, VNF vendors and service providers. The product offers and their relative orders are exposed to the market by means of smart contracts. Essentially, 5GZORRO Stakeholders, acting as offer providers, consolidate resources and services by abstracting features and characteristics from their technical specification. The RSOC defines how assets (resources and services) and product offers and orders are modeled through standard TM Forum OpenAPIs, reported below:

- *Resource Catalog Management API (TMF 634) [47]*: provides the models, dependencies, lifecycle management operations (CRUD) and event notification capabilities to handle entities of the Resource Catalogue.
- *Service Catalog Management API (TMF 633) [48]:* provides the models, dependencies, lifecycle management operations (CRUD) and event notification capabilities to handle entities of the Service Catalogue.
- *Product Catalog Management API (TMF 620) [49]*: provides the models, dependencies, lifecycle management operations (CRUD) and event notification capabilities to handle entities of the Product Catalogue.

- *Product Order API (TMF 622) [50]*: provides the models, dependencies, lifecycle management operations (CRUD) and event notification capabilities to issue and cancel a Product Order regarding some advertised Product Offer.
- *Geographic Address API (TMF 673) [51]*: provides the models, dependencies, lifecycle management operations (CRUD) and event notification capabilities to handle geographic address entities.
- *Party Management API (TMF 632 - Organization) [52]*: provides the models, dependencies, lifecycle management operations (CRUD) and event notification to handle Party entities and, more specifically, the Organization Object that describes the on-boarded stakeholder.

In 5GZORRO, the marketplace RSOC is mainly composed of the following object types:

- Resource Specifications backing VNF/CNF, Radio, Spectrum, Edge, and Cloud; these kinds of assets are maintained in the Resource Catalogue of the RSOC.
- Service Specifications backing Network Services and Network Slices; these kinds of assets are maintained in the Service Catalogue of the RSOC.
- Product Offerings that envelop business terms like pricing, SLA (described as standard TM Forum model stored in the SCLCM and only referenced in the Product Offering model), Offer Category, Geographical Address and, in the end, a Product Specification object that references the Resource and Service Specifications, stored in the RSOC, that the offer provider intends to propose to the Marketplace through the offer (i.e., the technical assets the provider wants to offer);
- Product Order representing the purchasing of an offer of the Marketplace.
- Organization object representing the information of the stakeholder owner of the RSOC of his domain.
- Various support objects for the RSOC internal functionalities.

3.2.1. Main Functionalities

The RSOC offers to other 5GZORRO Platform modules a specific interface to interact with the Marketplace catalogue. The main functionalities of the RSOC are reported below:

- Stakeholder on-boarding, triggered by the Identity and Permission Manager, resulting in the creation of the Organization object and the Product Offer Categories in the local RSOC. The Organization object contains information about the stakeholder (name, address, etc.), its associated DID and the token that can be used to request DIDs to the Identity and Permission Manager.
- Resource and Service Specifications on-boarding, retrieval, removal, and patch (CRUD).
- Product Specifications on-boarding, retrieval, removal, and patch (CRUD).
- Product Offering Prices on-boarding, retrieval, removal, and patch (CRUD).
- Product Offerings on-boarding, retrieval, removal, and patch (CRUD).
- Request of DIDs to Identity and Permission Manager for Product Offerings and Product Orders.
- Request of Product Offerings classification to SRSD (See Section 4.5).
- Request of Product Offerings DLT publication to SCLCM.
- Product Offerings synchronization, receiving Product Offerings and all the referred resources from other domains through Kafka Message Broker [30]; the RSOC, in this way, enables the automated discovery of available Product Offerings from different domains and service providers.

- Product Orders on-boarding, retrieval, removal and patch (CRUD); the Product Order represents a commercial agreement between the provider and the customer.
- Request of Product Orders DLT publication to SCLCM.

The NBI of the RSOC is consumed by a number of different 5GZORRO components: Governance Portal, Any Resource Manager (xRM) and E-Licensing Manager (eLM). Figure 3-3 summarizes the interactions of the RSOC with the other modules of the 5GZORRO Platform.

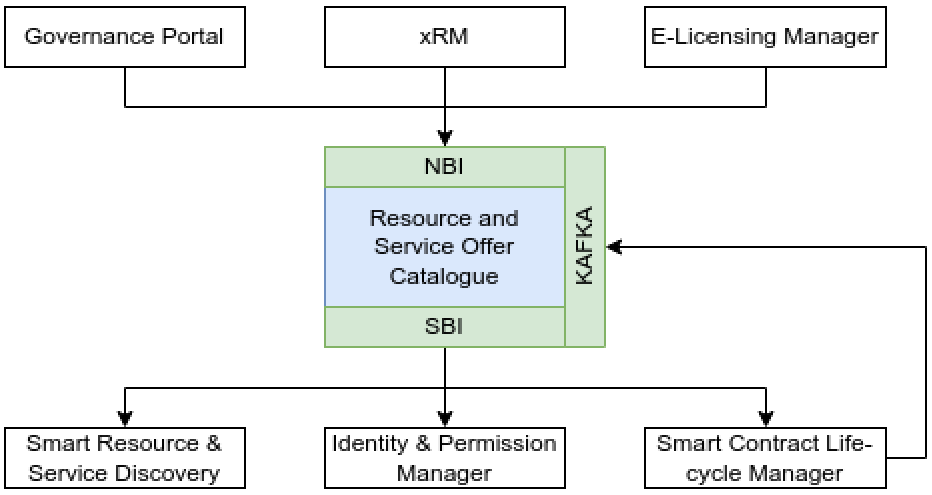

Governance Portal
xRM
E-Licensing Manager
NBI
Resource and Service Offer Catalogue
KAFKA
SBI
Smart Resource & Service Discovery
Identity & Permission Manager
Smart Contract Life-cycle Manager

Figure 3-3: RSOC interactions

3.2.2. Prototype Implementation

The RSOC is implemented as a *Spring Boot Java* based application. In the context of the application, *Java Persistence API (JPA)* is used to annotate the java classes representing the resources stored in the catalogue generated from the TM Forum Open API listed in the previous section. JPA APIs facilitate the integration and the interactions with the local Database Management System (DBMS), in this case an instance of *PostgreSQL*, where the elements of the catalogues are stored.

A number of REST Controllers, one for each kind of resource the catalogue can handle, implement the NBI; the logic to perform the internal life-cycle management of the entities of the catalogue is demanded to the Java Spring Services that implement the catalogue functionalities. The Southbound interface (SBI) of the RSOC consists of three REST clients used by the catalogue services to perform the internal life-cycle management of the catalogue entities: request of the DIDs to the ID&P, Classification and Publishing of the Product Offerings. A publish-subscribe communication mechanism is also available in the RSOC to implement asynchronous communications through a Kafka Message Broker, used for the synchronization (automated discovery) of available Product Offerings from different domains and service providers. Figure 3-4 shows the internal architecture of the RSOC.

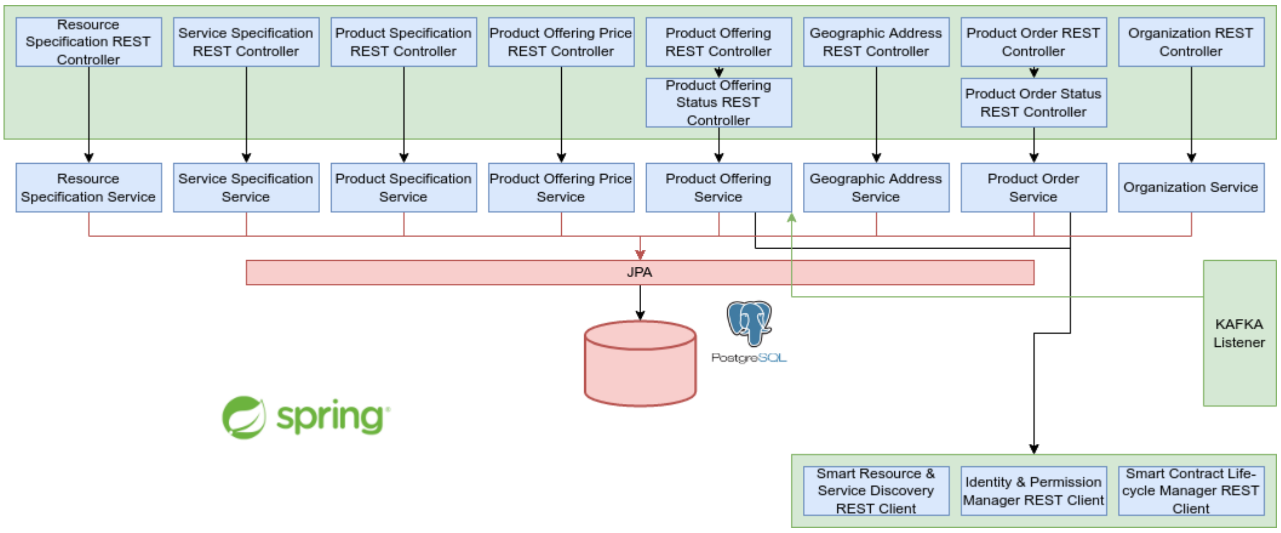
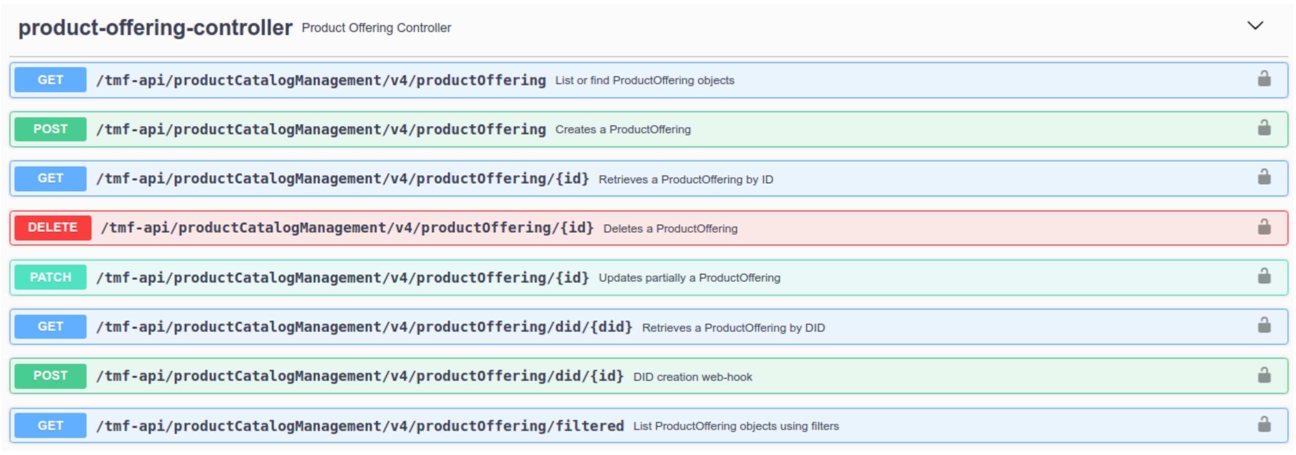

Resource Specification REST Controller
Service Specification REST Controller
Product Specification REST Controller
Product Offering Price REST Controller
Product Offering REST Controller
Product Offering Status REST Controller
Geographic Address REST Controller
Product Order REST Controller
Product Order Status REST Controller
Organization REST Controller
Resource Specification Service
Service Specification Service
Product Specification Service
Product Offering Price Service
Product Offering Service
Geographic Address Service
Product Order Service
Organization Service
JPA
PostgreSQL
spring
KAFKA Listener
Smart Resource & Service Discovery REST Client
Identity & Permission Manager REST Client
Smart Contract Life-cycle Manager REST Client

Figure 3-4: RSOC architecture

For the sake of brevity, only the swagger interface of the Product Offering REST Controller is reported and shown in Figure 3-5.



product-offering-controller Product Offering Controller

GET /tmf-api/productCatalogManagement/v4/productOffering List or find ProductOffering objects

POST /tmf-api/productCatalogManagement/v4/productOffering Creates a ProductOffering

GET /tmf-api/productCatalogManagement/v4/productOffering/{id} Retrieves a ProductOffering by ID

DELETE /tmf-api/productCatalogManagement/v4/productOffering/{id} Deletes a ProductOffering

PATCH /tmf-api/productCatalogManagement/v4/productOffering/{id} Updates partially a ProductOffering

GET /tmf-api/productCatalogManagement/v4/productOffering/did/{did} Retrieves a ProductOffering by DID

POST /tmf-api/productCatalogManagement/v4/productOffering/did/{id} DID creation web-hook

GET /tmf-api/productCatalogManagement/v4/productOffering/filtered List ProductOffering objects using filters

Figure 3-5: Product Offering Swagger Interface

In terms of deployment, the RSOC can run directly on a machine, Virtual or Physical, or as a software container. The containerized solution is the selected one to be able to deploy the module as a docker container orchestrated by k8s.

The code and the OpenAPI definitions are open source and available under Apache 2.0 License in a public repository into the 5GZORRO GitHub Organization (see [23]).

3.3.Smart Contracts Lifecycle Manager

The Smart Contract Lifecycle Manager (SCLCM) is the component of the 5GZORRO Marketplace that manages the interactions and events between the service layer of the 5GZORRO platform and the Marketplace DLT, acting as a gateway, using ledger-centric drivers to map abstract interfaces to DLT specific functionalities. This enables the module to meet the needs of broader business-centric workflows, while being agnostic to the DLT implementation (as key objective of the 5GZORRO architecture).

The goal of this module is to manage the lifecycle of 5GZORRO entities deployed to the underlying ledger, exposing the necessary abstract interfaces for managing these on the DLT and publish associated lifecycle events for subscribing applications to consume. For this purpose, the SCLCM, encompasses a set of DLT-specific drivers that implement abstract interfaces that define the key operations towards the DLT, for managing at DLT level: Product Offerings, Product Orders, SLAs, Licensing Terms and Spectokens (i.e. NFTs persisted on the DLT). Specific information models that encapsulate the corresponding TM Forum models for Product Offerings, Product Orders and SLAs were defined to persist on the underlying ledger this information (see deliverable D3.3 [2]). Whereas Licensing Terms and Spectoken make use of custom information models. Agreements, such as SLAs and Licensing Terms, are also backed, at SCLCM level, by traditional local storage to allow stakeholders to manage these domain-level Marketplace assets.

Product Offerings updates and their availability are disseminated to all trading stakeholder Marketplace nodes as a result of being posted to the SCLCM and then available to be purchased. Product Orders are also disseminated among the parties involved in a transaction regarding a specific Product Offering and, subsequently, also the related entities, such as SLAs and Licensing Terms, are published to the DLT and stored by the relevant parties of the aforementioned transaction: in this way the parties involved enter into an immutable agreement over the terms and conditions of the order. SLA violations and Licensing Term updates are handled, validated by means of smart contract and persisted to the underlying DLT.

3.3.1. Main Functionalities

The SCLCM offers to other 5GZORRO Platform modules a specific interface to interact with the underlying DLT to manage the lifecycle of the entities mentioned in the previous section; the main functionalities of the SCLCM are reported below:

- Exposing CRUD capabilities to publish and manage the lifecycle of Product Offerings in the DLT.
- Exposing publish and retrieve capabilities to manage Product Orders in the DLT and consequent management of SLAs and Licensing Terms.
- Publish the disseminated Product Offerings on a Message Broker Bus, Kafka [30] in the 5GZORRO implementation, in order to be fetched by RSOC instances in other domains (different from the one where the publication of the Offer occurred) in order to realize the synchronization of the 5GZORRO Marketplace catalogs (automatic offer discovery).
- Exposing CRUD capabilities for SLAs to define SLAs (off-chain persistence) that can be attached to Product Offerings and published on the DLT when a Product Order regarding a Product Offering that reference that SLA is performed.
- Exposing CRUD capabilities for Licensing Terms to define License (off-chain persistence) that can be attached to Product Offerings and published on the DLT when a Product Order regarding a Product Offering that reference that License is performed.
- Exposing CRUD capabilities to publish and manage Spectokens in order to enable spectrum trading.
- Define DLT-specific drivers and workflows to accomplish the aforementioned functionalities.

The NBI of the SCLCM is consumed by several different 5GZORRO components: RSOC, Governance Portal and Intelligent Network Slice and Service Manager (ISSM). In turn, SCLCM consumes services from other modules through its SBI. Figure 3-6 summarizes the interactions of the SCLCM with the other modules of the 5GZORRO Platform.

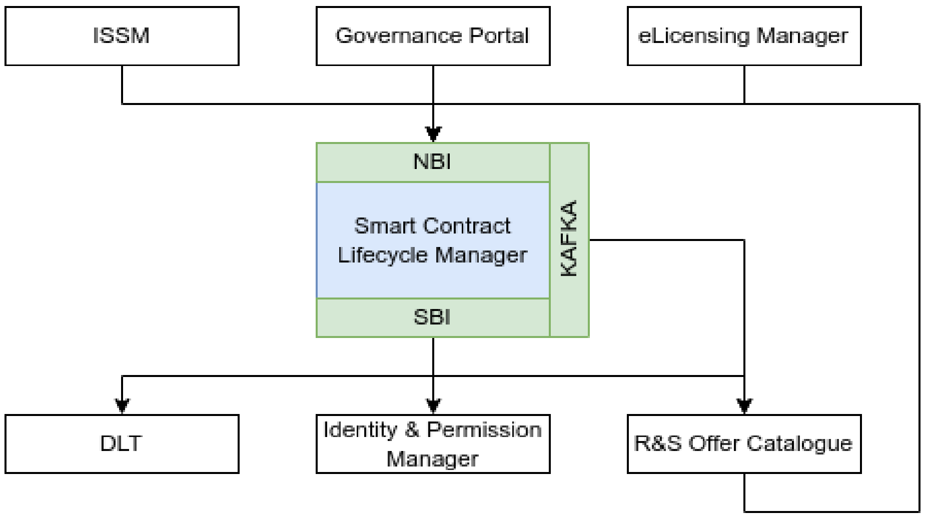


Figure 3-6: SCLCM interactions

3.3.2. Prototype Implementation

The SCLCM project is organized into three macro-sections:

- The SCLCM Application, implemented as a Spring Boot Java based application, exposes the REST NBI that enables access to the functionalities mentioned in the section above.
- The SCLCM DLT Driver API that define the interfaces that will be implemented by the DLT-specific drivers in order to define the management operations of the assets handled by the Marketplace shared between the stakeholders and reported in the previous chapter.
- The Corda DLT Driver that implements the DLT Driver API to cast the abstract management operations to the Corda DLT: this leads to the definition of the main elements of a Corda Application: *States, Contracts* and *Workflows*. The Java code that defines the implementation of the Driver will be packaged as JARs that will be consumed by the Corda Node in order to build the needed CordaApp.

The SCLCM Application makes use of Java Persistence API (JPA) to annotate the Java classes representing the TM Forum SLAs and the Licensing Terms to store these assets off-ledger (as previously mentioned, agreements are published on the DLT once they are referred by a Product Offering involved in an Order) into a local instance of PostgreSQL. Several REST Controllers, one for each kind of resource the Application can handle, implement the NBI, whilst the logic to use the DLT-specific driver, and to perform the lifecycle management of the DLT assets, is implemented by Java Spring services. The SBI of the SCLCM consists of different clients used by the SCLCM services to perform the internal life-cycle management of the DLT entities:

- Corda RPC client is used by the application to connect to the Corda DLT Node of its own domain to start the execution, triggered on-demand, of different workflows to accomplish the functionalities of the module and persist the required information.
- REST client to request DIDs for SLAs and Licensing Terms to the ID&P module deployed in the same domain of the SCLCM Application. The ID&P is also queried to retrieve the ledger identity of other stakeholders that joined the 5GZORRO Marketplace, e.g., since a Product Order DLT state needs to be disseminated both to the buyer and the seller, the SCLCM will query the ID&P to know the ledger identity of the seller (the buyer triggers the transaction and is known) in order to execute the Corda Workflows between the involved parties.
- REST client to retrieve assets from the RSOC, e.g., in the publication of a Product Order, before starting the Corda Workflow, to retrieve the Offers and the information referred in the latter.

The SCLCM also implements a publish-subscribe communication mechanism to enable the 5GZORRO automated Offer Discovery; when an Offer is disseminated among the Corda Nodes of the Marketplace Network the SCLCM instance that started the publishing flow also takes case to publish the disseminated offer in a Kafka Bus (acting as a publisher), this offer will be consumed by the RSOCs in the other domains in order to synchronize their internal catalogues. Figure 3-7 summarizes the SCLCM software architecture.

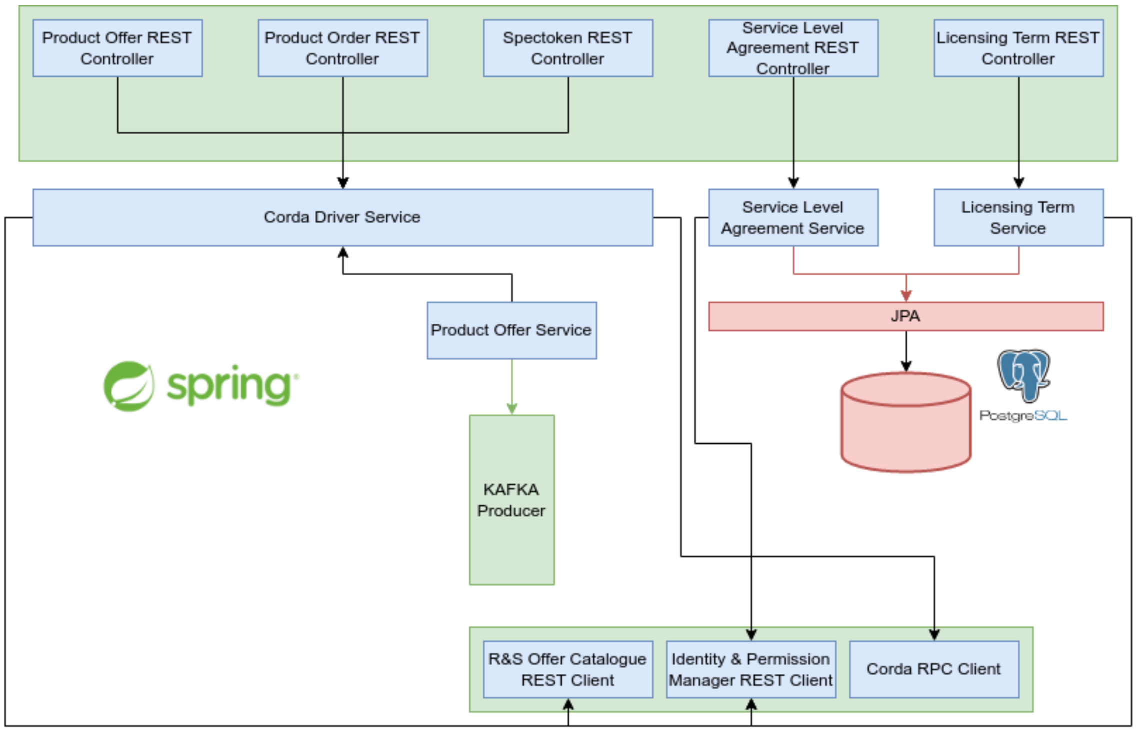
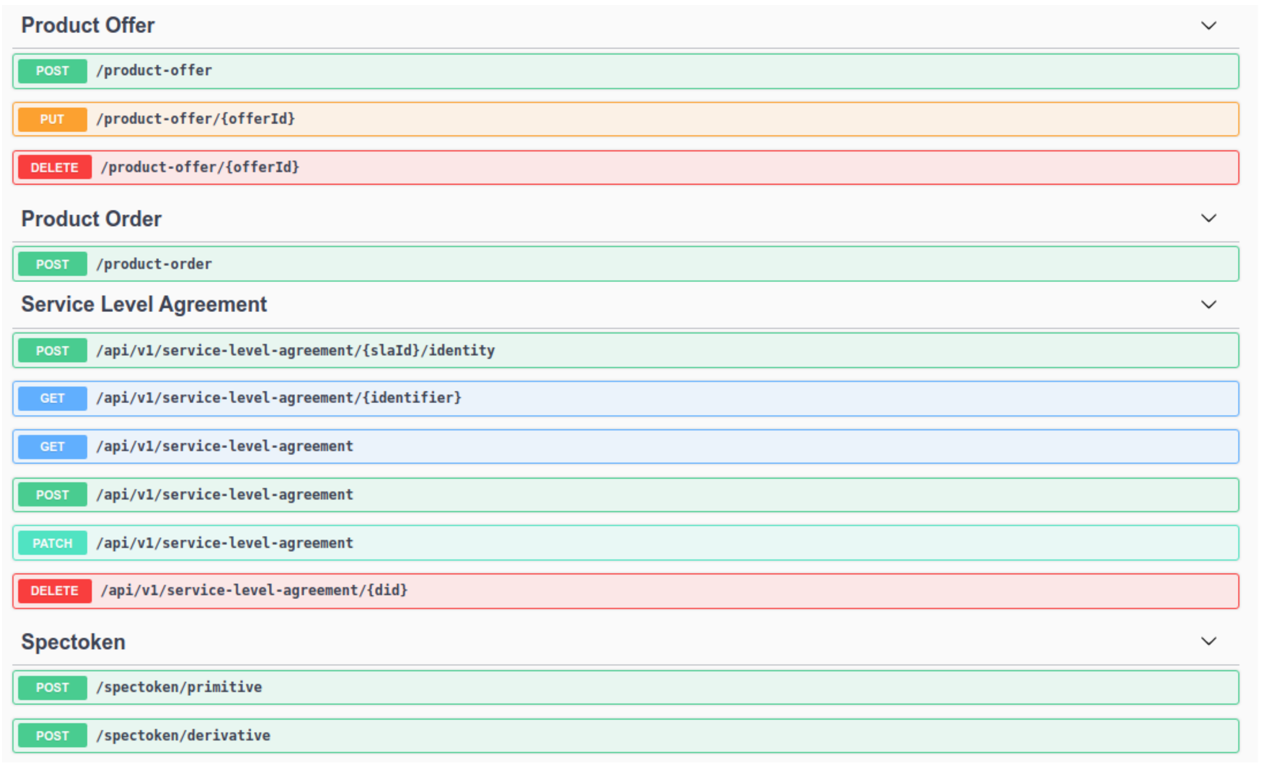

Product Offer REST Controller
Product Order REST Controller
Spectoken REST Controller
Service Level Agreement REST Controller
Licensing Term REST Controller
Corda Driver Service
Service Level Agreement Service
Licensing Term Service
JPA
Product Offer Service
spring
PostgreSQL
KAFKA Producer
R&S Offer Catalogue REST Client
Identity & Permission Manager REST Client
Corda RPC Client

Figure 3-7: SCLCM architecture

The swagger interface of the Product Offering REST Controller is depicted in Figure 3-8.



Product Offer

POST /product-offer

PUT /product-offer/{offerId}

DELETE /product-offer/{offerId}

Product Order

POST /product-order

Service Level Agreement

POST /api/v1/service-level-agreement/{slaId}/identity

GET /api/v1/service-level-agreement/{identifier}

GET /api/v1/service-level-agreement

POST /api/v1/service-level-agreement

PATCH /api/v1/service-level-agreement

DELETE /api/v1/service-level-agreement/{did}

Spectoken

POST /spectoken/primitive

POST /spectoken/derivative

Figure 3-8: SCLCM Swagger Interface

In terms of deployment, the SCLCM application can run directly on a machine, Virtual or Physical, or as a software container. The containerized solution is the selected one to be able to deploy the module as a docker container orchestrated by k8s.

The Corda DLT Driver, instead, is packaged as JARs (CordApp) that are copied to the Corda Nodes that compose the Corda Marketplace Network to enable the lifecycle management operations that are activated by the SCLCM Application as Corda Workflows.

The Code is released as open source and available under Apache 2.0 License in a public repository into the 5GZORRO GitHub Organization (see [24]).

3.4. **Marketplace Portal**

The Marketplace Portal is a component of 5GOZRRO's web GUI which exposes the Marketplace functionalities by interacting directly with the RSOC (Section 3.2), the SCLCM (Section 3.3) and Smart Resource and Service Discovery (SRSD) (Section 4.5). This component allows 5GZORRO's stakeholders to submit and browse the different offers to be available in the marketplace, as well as to place and review related orders.

While other web portals facilitate the onboarding of different services into a catalogue (mainly VNFs or general compute, storage and RAM resources), 5GZORRO's novelty lies in the fact that: (i) this model is extended to other resources such as spectrum (providing the system's ability to track and monitor its usage across different RAN elements); (ii) there is a real marketplace where multi-party business interactions can happen with strong support of SLA enforcement leveraging DLT; and, finally, (iv) there is alignment with standard initiatives such as TM Forum's APIs for a Telecom Marketplace, exposing them through a user-friendly web application.

3.4.1. **Main Functionalities**

The Marketplace Portal provides the views and interfaces needed for the stakeholders to directly interact with the 5GZORRO implemented marketplace functionalities. These functionalities are made available by the Portal through the interaction with the previously identified modules.

By interacting with the RSOC, the Marketplace Portal allows the stakeholders to onboard, fetch, and order marketplace products. These transactions take place through the onboarding of resources and the creation of Product Offerings and Product Orders. The interaction with the SCLCM enables the creation of customized SLA and License terms that are attached to the Product Offers.

Additionally, the interaction with the SRSD allows the stakeholders to intelligently retrieve catalogue offers based on high-level intents, abstracting the underlying complex ML-based offers classification and intent recognition.

The provided functionalities can be divided between those available for a resource/service provider and those available for a resource/service consumer and are described below.

From the Marketplace Portal a resource/service Provider is able to:

- Onboard resources/services into the catalogue.
- Create customized SLAs and License terms.
- Create a product offering price with attached license terms.
- Create a product offering and publish it to the marketplace.
- Terminate a Product order.

From the Marketplace Portal a resource/service Consumer is able to:

- Browse the most suitable offerings based on specific criteria.
- Place an order on a particular Product Offering.
- Browse all active Product Orders.

One of the major innovations brought by 5GZORRO is the consideration of spectrum resources to be traded as any other network resource. In this framework, the 5GZORRO Marketplace provides the following capabilities to Regulator stakeholders:

- Browse all submitted Product Orders which reference a Product Offering where spectrum is included as a resource.
- Terminate any order whose Product Offering contains the spectrum as an asset.

3.4.2. Prototype Implementation

Since the Marketplace Portal is also a component of 5GOZRRO's web GUI, it follows the same implementation already described in section 2.4. All the data and functionality provided by this module is made available through the interaction with 5GZORRO Marketplace components' APIs.

3.5.Validation Tests

This subsection reports the tests performed to validate the different operations supported by the Marketplace applications. For each software prototype, we first document the set of functional tests executed per component, which are in line with the supported operations declared in D3.3 [2], and then we provide some usage scenarios illustrating the joint operation of the components to handle the considered requests from the user-facing Marketplace Portal.

3.5.1.Functional Tests

Table 3-1 lists the set of tests performed to validate the functionalities implemented by the RSOC. The set of tests involves any functionality exposed by the NBI and the interactions implemented with the other modules whose service RSOC consumes, i.e., ID&P, SRSD and SCLCM.

Table 3-1: RSOC functional test set

Name	Description	Passed (Yes/No/Partially)
Stakeholder on-boarding & offer categories	Creation of the Organization Object (stakeholder information) and the categories that can be used for the Product Offering creation. The procedure is initiated by the Identity and Permission Manager through a POST request.	Yes
Resource Specification CRUD	Create, Read, Update and Delete tests of a Resource Specification from the NBI	Yes
Service Specification CRUD	Create, Read, Update and Delete tests of a Service Specification from the NBI	Yes
Product Specification CRUD	Create, Read, Update and Delete tests of a Product Specification from the NBI	Yes
Product Offering Prices CRUD	Create, Read, Update and Delete tests of a Product Offering Prices from the NBI	Yes

Product Offering CRUD (local catalogue)	Create, Read, Update and Delete tests of a Product Offering from the NBI	Yes
DID request	GET request of the DID to the Identity and Permission Manager for Product Offerings and Product Orders	Yes
Product Offering classification	POST request to the SRSD for the classification of the created Product Offering	Yes
Product Offering publication	POST request to the SCLCM for the publishing of the created Product Offering	Yes
Automatic Product Offerings discovery	Retrieval, from Kafka Bus, of Product Offerings created in other stakeholder domains through their local catalogues	Yes
Product Order CRUD (local catalogue)	Create, Read, Update and Delete tests of a Product Order from the NBI	Yes
Product Order publication	POST request to the SCLCM for the publishing of the created Product Order	Yes

Table 3-2 lists the set of tests performed to validate the functionalities implemented by the SCLCM. These tests validate the services exposed by the SCLCM NBI being consumed by other 5GZORRO platform modules (such as the Marketplace Portal and RSOC), as well as the proper integration with underlying components involved during the operations (i.e., the Marketplace DLT).

Table 3-2: SCLCM functional test set

Name	**Description**	**Passed (Yes/No/Partially)**
Product Offering CRUD	Create, Read (from Vault, after DLT dissemination), Update and Delete tests of a Product Offering from the NBI and as consequence of a RSOC Product Offering creation (POST)	Yes
Product Order publish and retrieve	Create and Read (from Vault, after DLT dissemination) tests of a Product Order from the NBI and as consequence of a RSOC Product Order creation (POST)	Yes
Automatic offer discovery	Publishing of the disseminated Product Offering (after POST request from the NBI) on the Kafka Bus in order to be retrieved by the RSOC in other stakeholder domains	Yes
SLA CRUD	Create, Read, Update and Delete tests of a SLA from the NBI	Yes
SLA DLT dissemination	Dissemination test of the SLA referred by a Product Offering purchased through a Product Order	Yes
Licensing Terms CRUD	Create, Read, Update and Delete tests of a Licensing Term from the NBI	Yes

Licensing Term DLT dissemination	Dissemination test of the SLA referred by a Product Offering purchased through a Product Order	Yes
Spectoken publish and retrieve	Create and Read (from Vault, after DLT dissemination) tests of a Spectoken from the NBI.	Yes

Table 3-3 lists the set of tests performed to validate the functionalities implemented by the Portal related to Marketplace capabilities. The tests reported have been performed through the Portal interface and validate also the integration when interacting with the RSOC (for operations related to resources, services and offers management), with the SCLCM (for operations related to SLA, License terms and spectoken management) and with SRSD (for operations related to intent-based discovery).

Table 3-3: Portal functional test set related to Marketplace

Name	**Description**	**Passed (Yes/No/Partially)**
On-board resources into the RSOC	Fetch available assets and on-board new resources into the RSOC from the Provider’s Portal interface	Yes
On-board services into the RSOC	Fetch available assets and on-board new services into the RSOC from the Provider’s Portal interface	Yes
Create customized SLA and License Terms	Create a customized SLA or License Term from the Provider’s Portal interface by editing a Legal Prose Template already submitted in the Portal	Yes
Create Product Offering Price	Create a product offering price with attached license terms from the Provider’s Portal interface	Yes
Create Product Offering	Create a product offering and publish it to the marketplace from the Provider’s Portal interface	Yes
Browse Product offering	Browse the most suitable offerings based on specific criteria from the Consumer’s Portal interface	Yes
Create Product Order	Place an order on a particular Product Offering from the Consumer’s Portal interface	Yes
Browse Product Order	Browse all active Product Orders from the Consumer’s Portal interface	Yes
Terminate Product Order	Terminate a Product order request from the Provider’s Portal interface	Yes
Browse Product Orders with Spectrum resource	Browse all submitted Product Orders which reference a Product Offering where spectrum is included as a resource from the Regulator’s Portal interface	Yes
Terminate Product Order with Spectrum resource	Terminate any order whose Product Offering contains the spectrum as an asset from the Regulator’s Portal interface	Yes

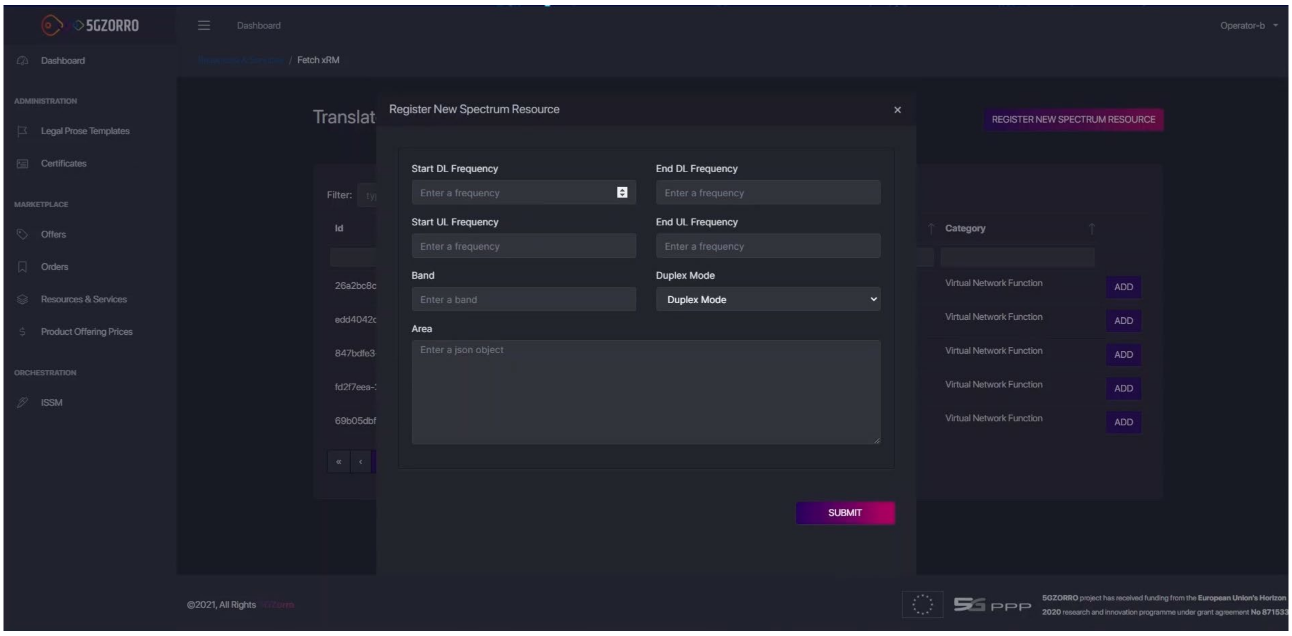
3.5.2. Platform Operations

Next, some usage scenarios of the Marketplace prototypes, in conjunction with the underlying DLT, are described to validate the system capabilities related to the onboarding of resource and service specifications, the creation of additional offers-related components such as prices and customized SLA/eLicense contracts, the composition and dissemination of offers, and the ordering of published offers. This report serves not only to demonstrate the attainment of expected capabilities, but to illustrate the usage of the platform through the 5GZORRO Portal.

3.5.2.1. Registration of a spectrum asset

Spectrum is the only resource type in 5GZORRO that is not self-discoverable and needs to be defined by the spectrum resource provider. The spectrum resource provider (in this sample scenario represented by Operator-b) with a valid spectrum certificate is allowed to register spectrum assets from the 5GZORRO Portal.

As it can be seen in Figure 3-9, the spectrum resource provider completes a form containing information on the start and end frequencies for downlink and uplink, the standardized band, spectrum operation mode, and area of application of the to-be-created spectrum asset.

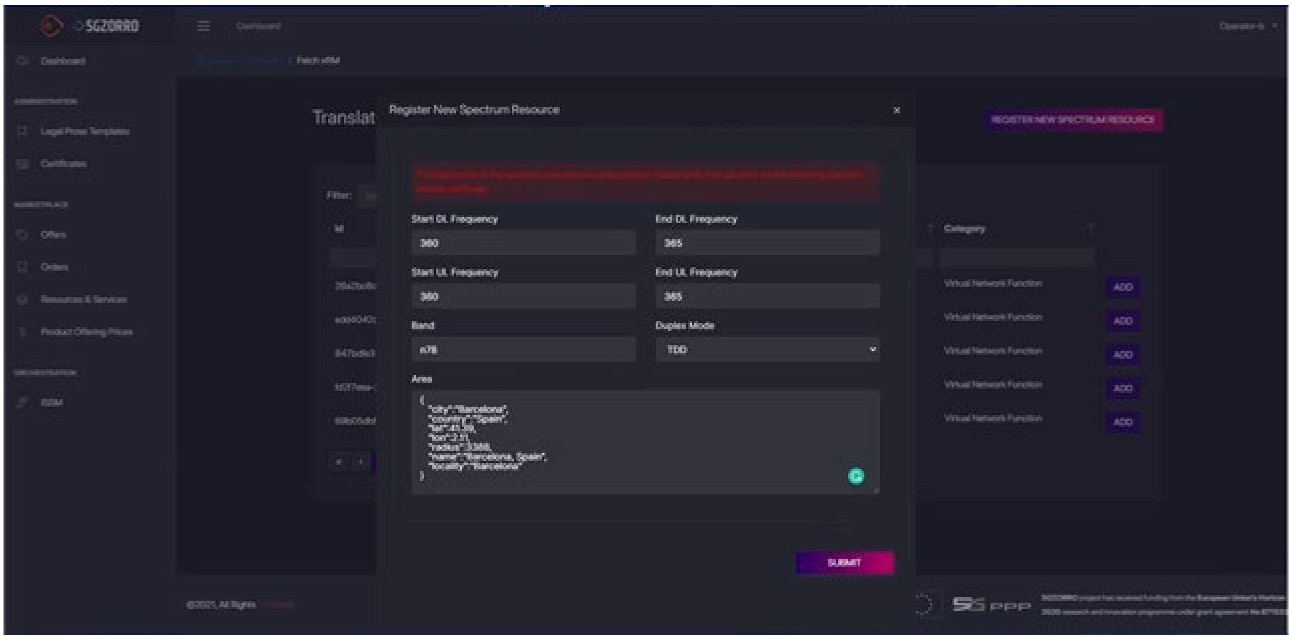


Register New Spectrum Resource

Start DL Frequency: Enter a frequency	End DL Frequency: Enter a frequency
Start UL Frequency: Enter a frequency	End UL Frequency: Enter a frequency
Band: Enter a band	Duplex Mode: Duplex Mode
Area: Enter a json object	

Figure 3-9: Spectrum asset creation view

This information is submitted to the Spectrum Resource Manager (sRM) component of the 5GZORRO platform, which compares this information with all the spectrum certificates registered by the ID&P. If the information of the spectrum asset is not found to be within any spectrum certificate, the sRM will return an error that will be displayed by the Portal (see Figure 3-10), avoiding the registration of such invalid spectrum asset.



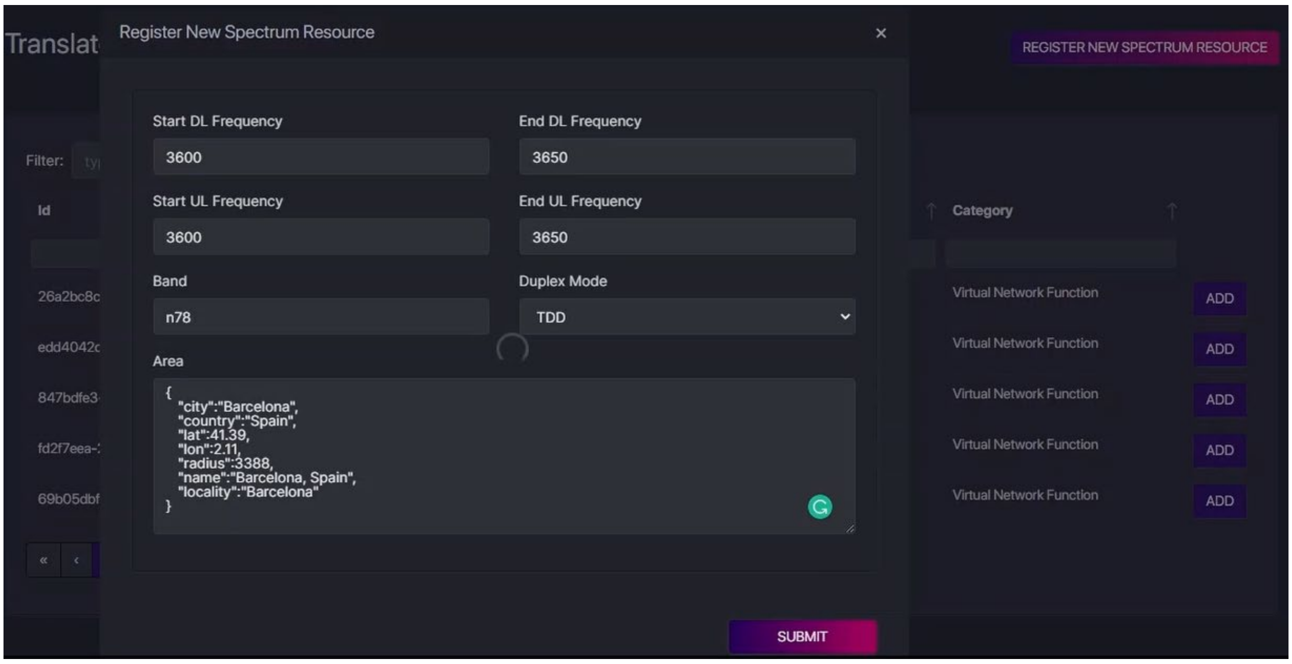
[illegible]

Start DL Frequency: 360
End DL Frequency: 365
Start UL Frequency: 360
End UL Frequency: 365
Band: n78
Duplex Mode: TDD
Area:

```
{
  "city":"Barcelona",
  "country":"Spain",
  "lat":41.39,
  "lon":2.11,
  "radius":3388,
  "name":"Barcelona, Spain",
  "locality":"Barcelona"
}
```

Figure 3-10: Spectrum asset creation rejected with invalid frequency range error

On the contrary, if the information in the spectrum asset is compatible with a valid spectrum certificate, as depicted in Figure 3-11, the asset is successfully created and stored in the sRM, for its future retrieval.



Start DL Frequency: 3600
End DL Frequency: 3650
Start UL Frequency: 3600
End UL Frequency: 3650
Band: n78
Duplex Mode: TDD
Area:

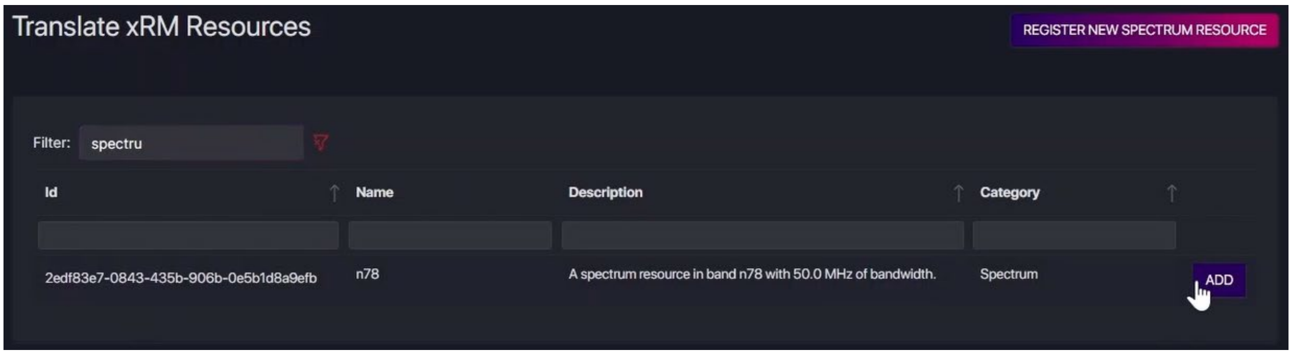
```
{
  "city":"Barcelona",
  "country":"Spain",
  "lat":41.39,
  "lon":2.11,
  "radius":3388,
  "name":"Barcelona, Spain",
  "locality":"Barcelona"
}
```

Figure 3-11: Portal view on the registration of a valid spectrum asset

3.5.2.2. Addition of resource and service specifications

Stakeholder's assets are defined in their own specific data models and need to be translated to TM Forum's data models to be included in the RSOC as resource or service specifications. To do this, available assets are fetched from the Portal, which interacts with the xRM, acting as an abstraction layer on top of the different underlying managers, to retrieve such available assets and upon request of onboarding, to translate the given asset into a TM Forum resource or service specification.

Figure 3-12 shows the 5GZORRO Portal view for onboarding a given asset, spectrum in this case, to the Provider domain internal storage of available resources. This operation will trigger the translation and subsequent persistence in the Provider RSOC instance.

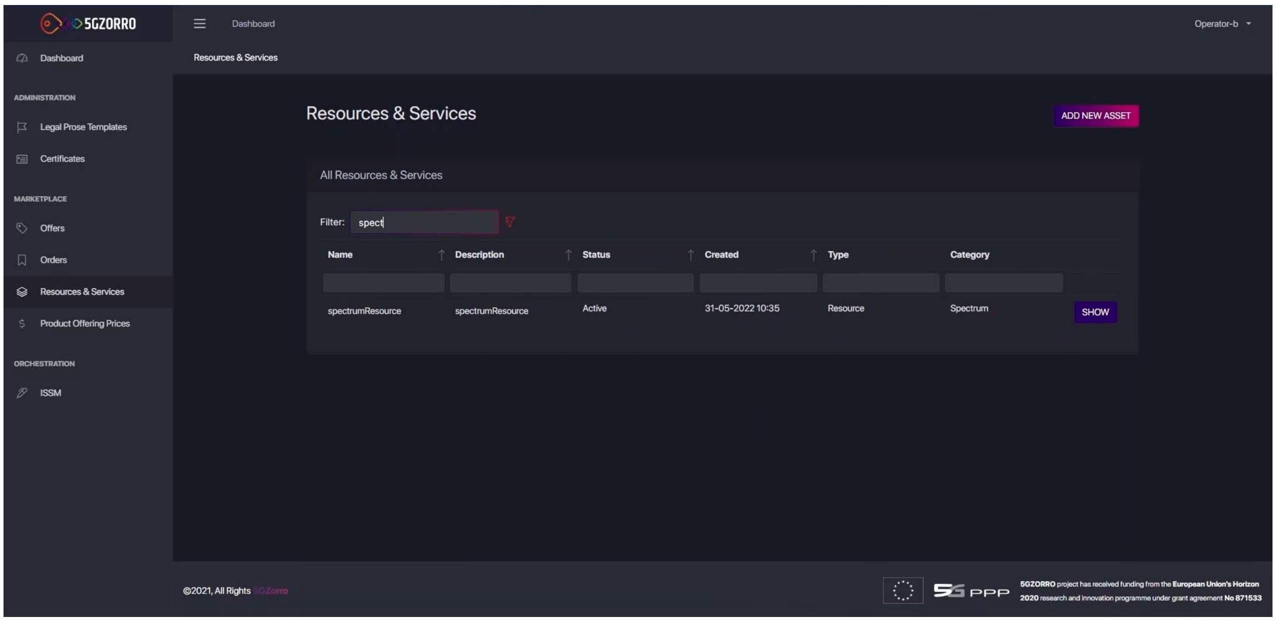

Translate xRM Resources

Filter: spectru

Id	Name	Description	Category	
2edf83e7-0843-435b-906b-0e5b1d8a9efb	n78	A spectrum resource in band n78 with 50.0 MHz of bandwidth.	Spectrum	ADD

Figure 3-12: Portal view on the translation of (spectrum) assets to include them as resources or services in the RSOC

As a result, the onboarded (spectrum) resource can be seen in the Portal under the "Resources and Services" view, as illustrated in Figure 3-13, becoming available for its inclusion in a subsequent product offer.



Resources & Services

All Resources & Services

Filter: spect

Name	Description	Status	Created	Type	Category	
spectrumResource	spectrumResource	Active	31-05-2022 10:35	Resource	Spectrum	SHOW

©2021, All Rights 5GZorro

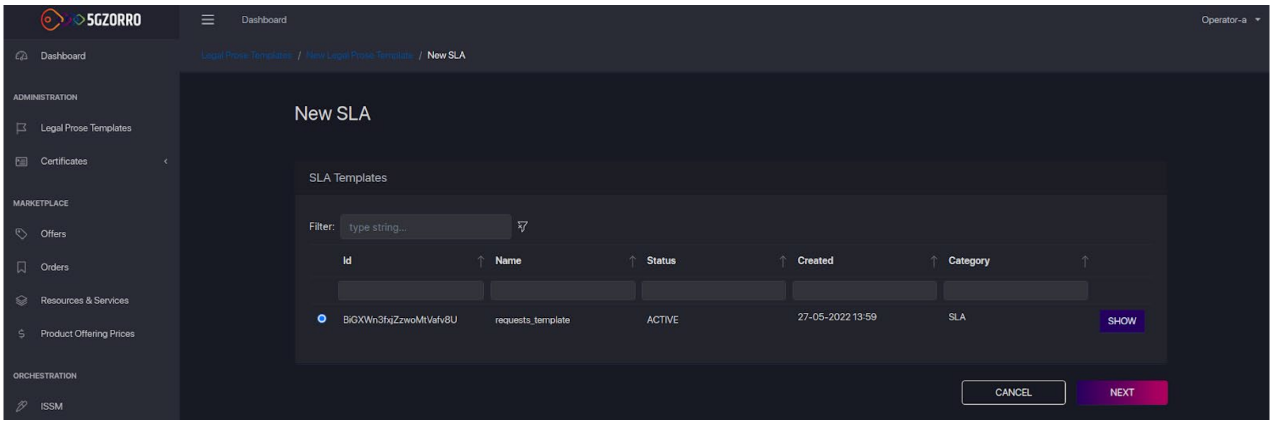
5GZORRO project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871533

Figure 3-13: View on the list of available (spectrum) resources in the Provider RSOC

3.5.2.3. Creation of customized SLA contract

As previously reported, agreement assets, namely SLAs and Licensing Terms, are created off-ledger and then they could be referred by a Product Offering; once this offer is purchased, through a Product Order, the SLA correspondent DLT state is created and disseminated among the parties involved in the order transaction. Following, the creation of an SLA, from the off-ledger creation on the local-domain SCLCM to its usage in a Product Offering and its dissemination in the Marketplace Corda DLT when the mentioned offer is purchased (order).

From the Legal Prose Templates interface in the Portal, a new SLA can be created by filing a previously on-boarded Legal Prose Template. Upon selecting a template among the ones available (see Figure 3-14), it can be filled and a correspondent SLA created through a POST request to the SCLCM (see Figure 3-15).

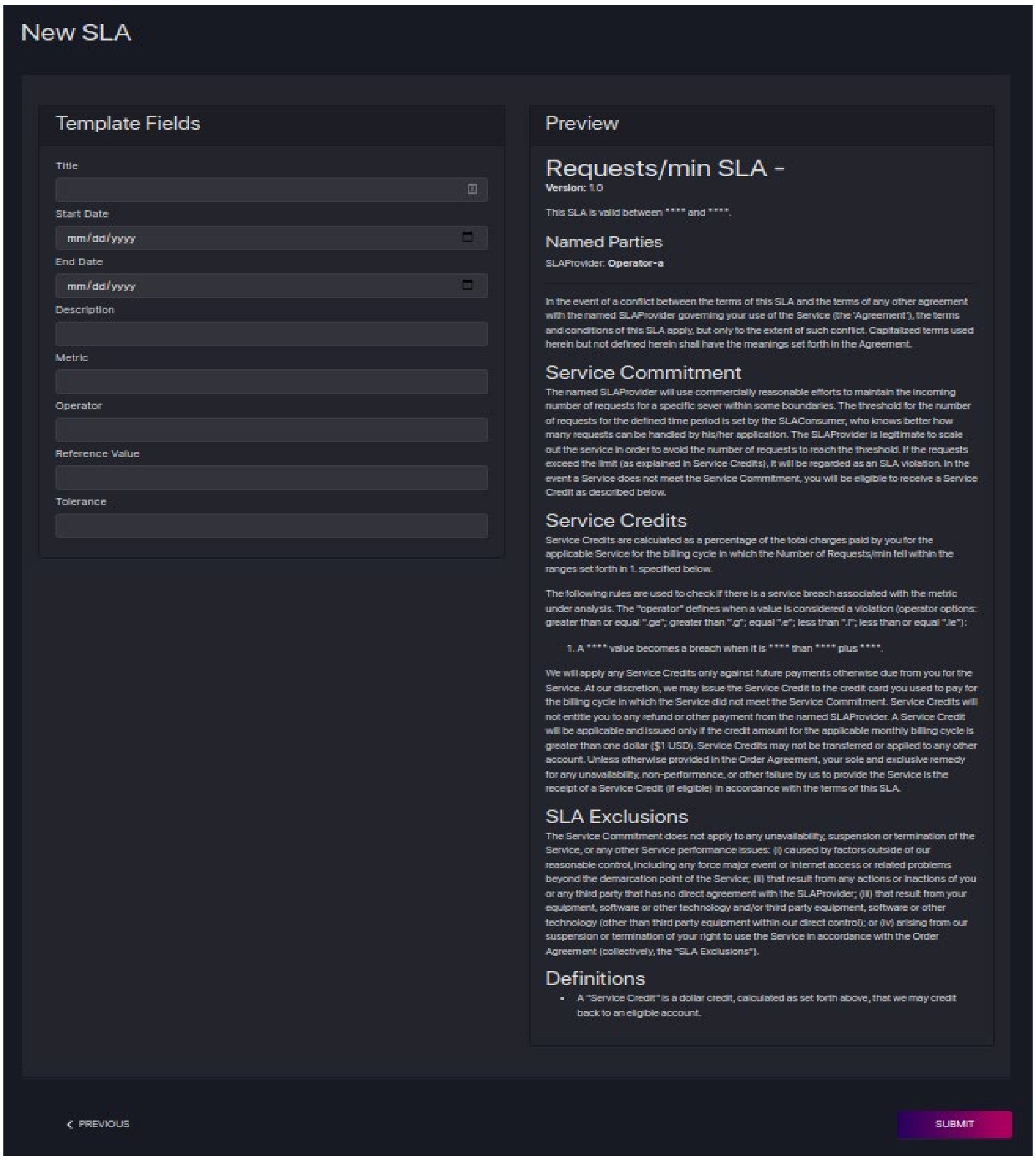


New SLA

SLA Templates

Id	Name	Status	Created	Category
BiGXWn3fxjZzwoMtVafv8U	requests_template	ACTIVE	27-05-2022 13:59	SLA

Figure 3-14: Marketplace Portal, SLA creation, available Legal Prose Templates



New SLA

Template Fields

Title

Start Date: mm/dd/yyyy

End Date: mm/dd/yyyy

Description

Metric

Operator

Reference Value

Tolerance

Preview

Requests/min SLA -

Version: 1.0

This SLA is valid between **** and ****.

Named Parties

SLAProvider: **Operator-a**

In the event of a conflict between the terms of this SLA and the terms of any other agreement with the named SLAProvider governing your use of the Service (the 'Agreement'), the terms and conditions of this SLA apply, but only to the extent of such conflict. Capitalized terms used herein but not defined herein shall have the meanings set forth in the Agreement.

Service Commitment

The named SLAProvider will use commercially reasonable efforts to maintain the incoming number of requests for a specific sever within some boundaries. The threshold for the number of requests for the defined time period is set by the SLAConsumer, who knows better how many requests can be handled by his/her application. The SLAProvider is legitimate to scale out the service in order to avoid the number of requests to reach the threshold. If the requests exceed the limit (as explained in Service Credits), it will be regarded as an SLA violation. In the event a Service does not meet the Service Commitment, you will be eligible to receive a Service Credit as described below.

Service Credits

Service Credits are calculated as a percentage of the total charges paid by you for the applicable Service for the billing cycle in which the Number of Requests/min fell within the ranges set forth in 1. specified below.

The following rules are used to check if there is a service breach associated with the metric under analysis. The "operator" defines when a value is considered a violation (operator options: greater than or equal ".ge"; greater than ".g"; equal ".e"; less than ".l"; less than or equal ".le"):

1. A **** value becomes a breach when it is **** than **** plus ****.

We will apply any Service Credits only against future payments otherwise due from you for the Service. At our discretion, we may issue the Service Credit to the credit card you used to pay for the billing cycle in which the Service did not meet the Service Commitment. Service Credits will not entitle you to any refund or other payment from the named SLAProvider. A Service Credit will be applicable and issued only if the credit amount for the applicable monthly billing cycle is greater than one dollar ($1 USD). Service Credits may not be transferred or applied to any other account. Unless otherwise provided in the Order Agreement, your sole and exclusive remedy for any unavailability, non-performance, or other failure by us to provide the Service is the receipt of a Service Credit (if eligible) in accordance with the terms of this SLA.

SLA Exclusions

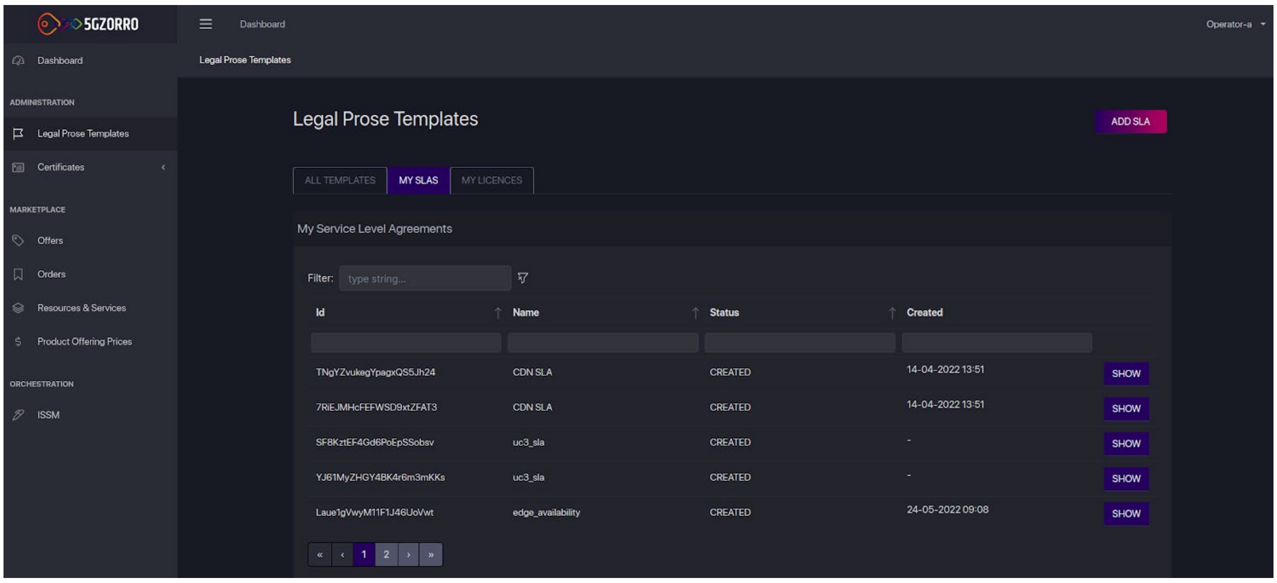
The Service Commitment does not apply to any unavailability, suspension or termination of the Service, or any other Service performance issues: (i) caused by factors outside of our reasonable control, including any force major event or Internet access or related problems beyond the demarcation point of the Service; (ii) that result from any actions or inactions of you or any third party that has no direct agreement with the SLAProvider; (iii) that result from your equipment, software or other technology and/or third party equipment, software or other technology (other than third party equipment within our direct control); or (iv) arising from our suspension or termination of your right to use the Service in accordance with the Order Agreement (collectively, the "SLA Exclusions").

Definitions

- A "Service Credit" is a dollar credit, calculated as set forth above, that we may credit back to an eligible account.

Figure 3-15: Marketplace Portal, SLA creation, template filling

Once the template has been filled and submitted an SLA will be created; from the Marketplace Portal all the created SLAs are displayed.



Legal Prose Templates

My Service Level Agreements

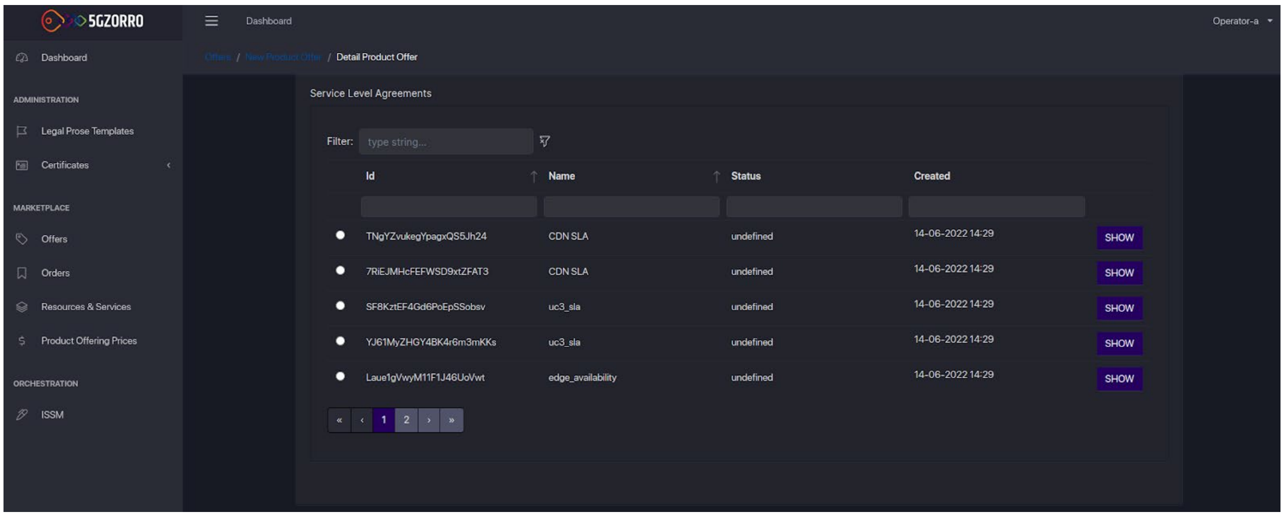
Filter: type string...

Id	Name	Status	Created	
TNgYZvukegYpagxQS5Jh24	CDN SLA	CREATED	14-04-2022 13:51	SHOW
7RiEJMHcFEFWSD9xtZFAT3	CDN SLA	CREATED	14-04-2022 13:51	SHOW
SF8KztEF4Gd6PoEpSSobsv	uc3_sla	CREATED	-	SHOW
YJ61MyZHGY4BK4r6m3mKKs	uc3_sla	CREATED	-	SHOW
Laue1gVwyM11F1J46UoVwt	edge_availability	CREATED	24-05-2022 09:08	SHOW

« ‹ 1 2 › »

Figure 3-16: Marketplace Portal, SLA creation, created SLAs

During the creation of a Product Offering, we can select one of the SLA previously created (Figure 3-17):



Service Level Agreements

Filter: type string...

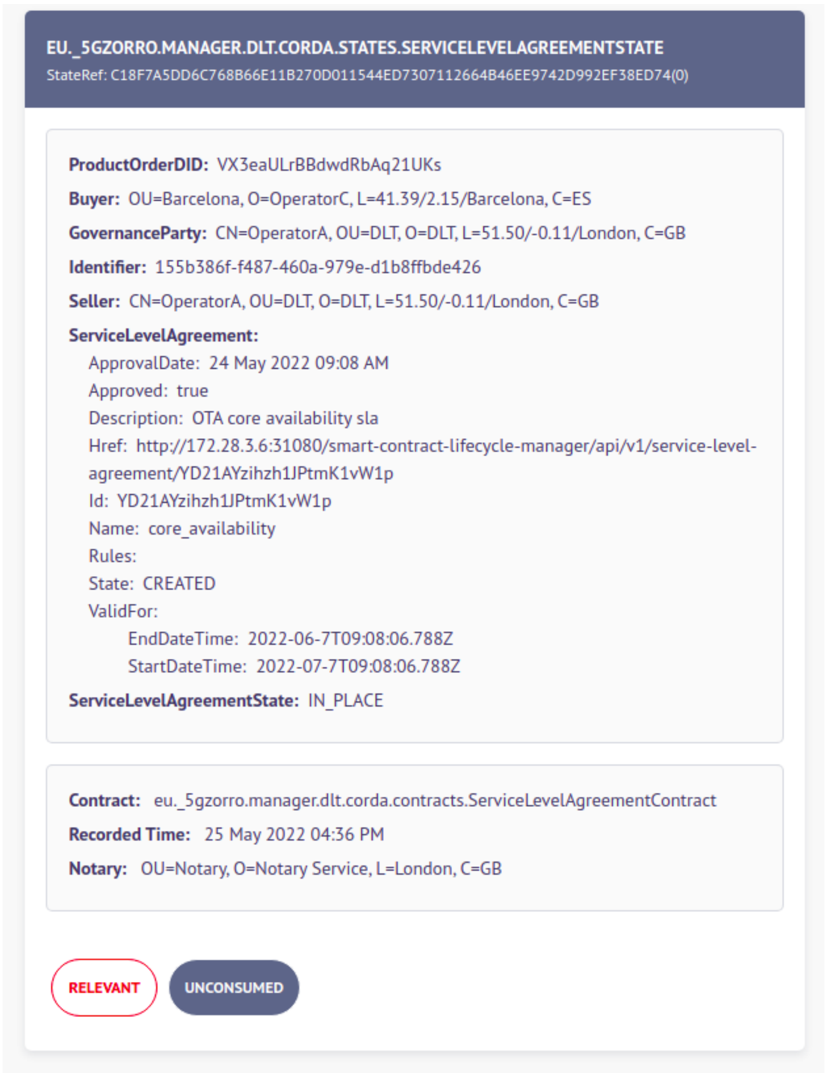
	Id	Name	Status	Created	
●	TNgYZvukegYpagxQS5Jh24	CDN SLA	undefined	14-06-2022 14:29	SHOW
●	7RiEJMHcFEFWSD9xtZFAT3	CDN SLA	undefined	14-06-2022 14:29	SHOW
●	SF8KztEF4Gd6PoEpSSobsv	uc3_sla	undefined	14-06-2022 14:29	SHOW
●	YJ61MyZHGY4BK4r6m3mKKs	uc3_sla	undefined	14-06-2022 14:29	SHOW
●	Laue1gVwyM11F1J46UoVwt	edge_availability	undefined	14-06-2022 14:29	SHOW

« ‹ 1 2 › »

Figure 3-17: Marketplace Portal, Product Offering creation, SLA selection

Once a Product Offering has been created and then purchased with the creation of the corresponding Product Order, the latter will be sent to the SCLCM that will start a Workflow towards its own Corda Node in order to start the dissemination of the submitted Product Order and the referred agreements, i.e., the dissemination of the created SLA on the DLT.

Figure 3-18 shows the DLT state that has been created and disseminated for the SLA *YD21AYzihzh1JPtmK1vW1p* after an Order that involved a Product Offering with that particular SLA has been persisted into the DLT.



EU._5GZORRO.MANAGER.DLT.CORDA.STATES.SERVICELEVELAGREEMENTSTATE
StateRef: C18F7A5DD6C768B66E11B270D011544ED7307112664B46EE9742D992EF38ED74(0)

ProductOrderDID: VX3eaULrBBdwdRbAq21UKs
Buyer: OU=Barcelona, O=OperatorC, L=41.39/2.15/Barcelona, C=ES
GovernanceParty: CN=OperatorA, OU=DLT, O=DLT, L=51.50/-0.11/London, C=GB
Identifier: 155b386f-f487-460a-979e-d1b8ffbde426
Seller: CN=OperatorA, OU=DLT, O=DLT, L=51.50/-0.11/London, C=GB
ServiceLevelAgreement:
ApprovalDate: 24 May 2022 09:08 AM
Approved: true
Description: OTA core availability sla
Href: http://172.28.3.6:31080/smart-contract-lifecycle-manager/api/v1/service-level-agreement/YD21AYzihzh1JPtmK1vW1p
Id: YD21AYzihzh1JPtmK1vW1p
Name: core_availability
Rules:
State: CREATED
ValidFor:
EndDateTime: 2022-06-7T09:08:06.788Z
StartDateTime: 2022-07-7T09:08:06.788Z
ServiceLevelAgreementState: IN_PLACE

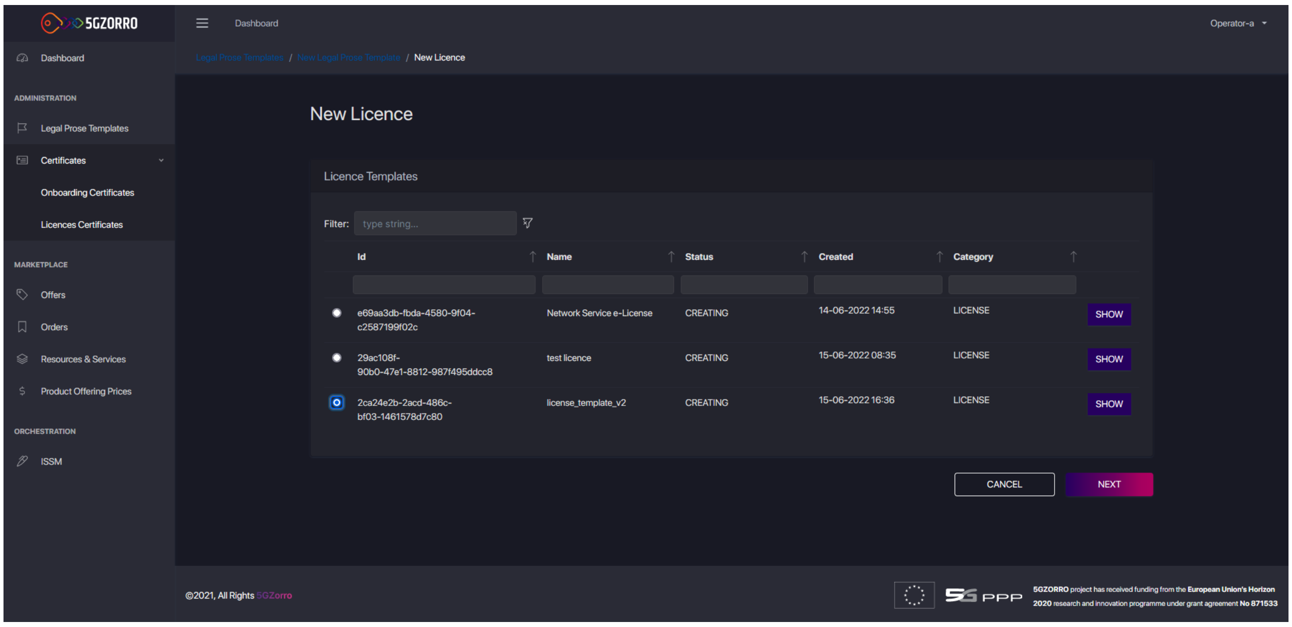
Contract: eu._5gzorro.manager.dlt.corda.contracts.ServiceLevelAgreementContract
Recorded Time: 25 May 2022 04:36 PM
Notary: OU=Notary, O=Notary Service, L=London, C=GB

Figure 3-18: Corda DLT SLA State

3.5.2.4. Creation of customized e-License contract

Any Network Function (xNF) as software component onboarded on the 5GZORRO Marketplace can be associated with a e-License contract that fully defines the terms and conditions that apply to an individual transaction for a xNF. This process takes two steps: first providers create a license contract that will define the agreement associated to the transaction, and second, they will define a Product Offering Price which is documented on the next validation item.

The creation process of a customized e-License contract is similar to the one already described for the SLA. From the Legal Prose Templates interface in the Portal, a new e-License contract can be created by filing a previously on-boarded Legal Prose Template. Chosen a template among the ones available (see Figure 3-19), the selected one can be filled and a corresponding e-License created through a POST request to the SCLCM (see Figure 3-20).



New Licence

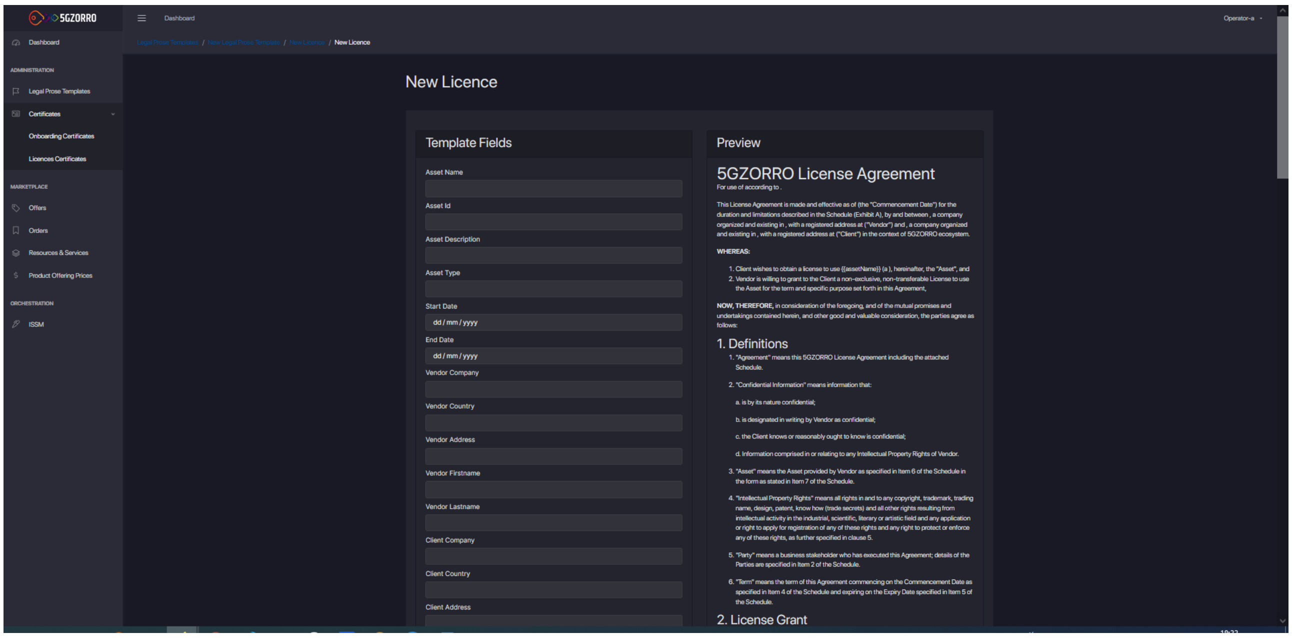
Licence Templates

Filter: type string...

	Id	Name	Status	Created	Category	
○	e69aa3db-fbda-4580-9f04-c2587199f02c	Network Service e-License	CREATING	14-06-2022 14:55	LICENSE	SHOW
○	29ac108f-90b0-47e1-8812-987f495ddcc8	test licence	CREATING	15-06-2022 08:35	LICENSE	SHOW
●	2ca24e2b-2acd-486c-bf03-1461578d7c80	license_template_v2	CREATING	15-06-2022 16:36	LICENSE	SHOW

CANCEL
NEXT
©2021, All Rights 5GZorro
5G PPP
5GZORRO project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871533

Figure 3-19: Marketplace Portal, e-License creation, available Legal Prose Templates



New Licence

Template Fields

Asset Name

Asset Id

Asset Description

Asset Type

Start Date: dd / mm / yyyy

End Date: dd / mm / yyyy

Vendor Company

Vendor Country

Vendor Address

Vendor Firstname

Vendor Lastname

Client Company

Client Country

Client Address

Preview

5GZORRO License Agreement

For use of according to .

This License Agreement is made and effective as of (the "Commencement Date") for the duration and limitations described in the Schedule (Exhibit A), by and between , a company organized and existing in , with a registered address at ("Vendor") and , a company organized and existing in , with a registered address at ("Client") in the context of 5GZORRO ecosystem.

WHEREAS:

1. Client wishes to obtain a license to use {{assetName}} (a), hereinafter, the "Asset", and
2. Vendor is willing to grant to the Client a non-exclusive, non-transferable License to use the Asset for the term and specific purpose set forth in this Agreement,

NOW, THEREFORE, in consideration of the foregoing, and of the mutual promises and undertakings contained herein, and other good and valuable consideration, the parties agree as follows:

1. Definitions

1. "Agreement" means this 5GZORRO License Agreement including the attached Schedule.
2. "Confidential Information" means information that:

   a. is by its nature confidential;

   b. is designated in writing by Vendor as confidential;

   c. the Client knows or reasonably ought to know is confidential;

   d. Information comprised in or relating to any Intellectual Property Rights of Vendor.
3. "Asset" means the Asset provided by Vendor as specified in Item 6 of the Schedule in the form as stated in Item 7 of the Schedule.
4. "Intellectual Property Rights" means all rights in and to any copyright, trademark, trading name, design, patent, know how (trade secrets) and all other rights resulting from intellectual activity in the industrial, scientific, literary or artistic field and any application or right to apply for registration of any of these rights and any right to protect or enforce any of these rights, as further specified in clause 5.
5. "Party" means a business stakeholder who has executed this Agreement; details of the Parties are specified in Item 2 of the Schedule.
6. "Term" means the term of this Agreement commencing on the Commencement Date as specified in Item 4 of the Schedule and expiring on the Expiry Date specified in Item 5 of the Schedule.

2. License Grant

Figure 3-20: Marketplace Portal, e-License creation, template filling

Once the template has been filled and submitted an e-License will be created and it will be made available from the Marketplace Portal.

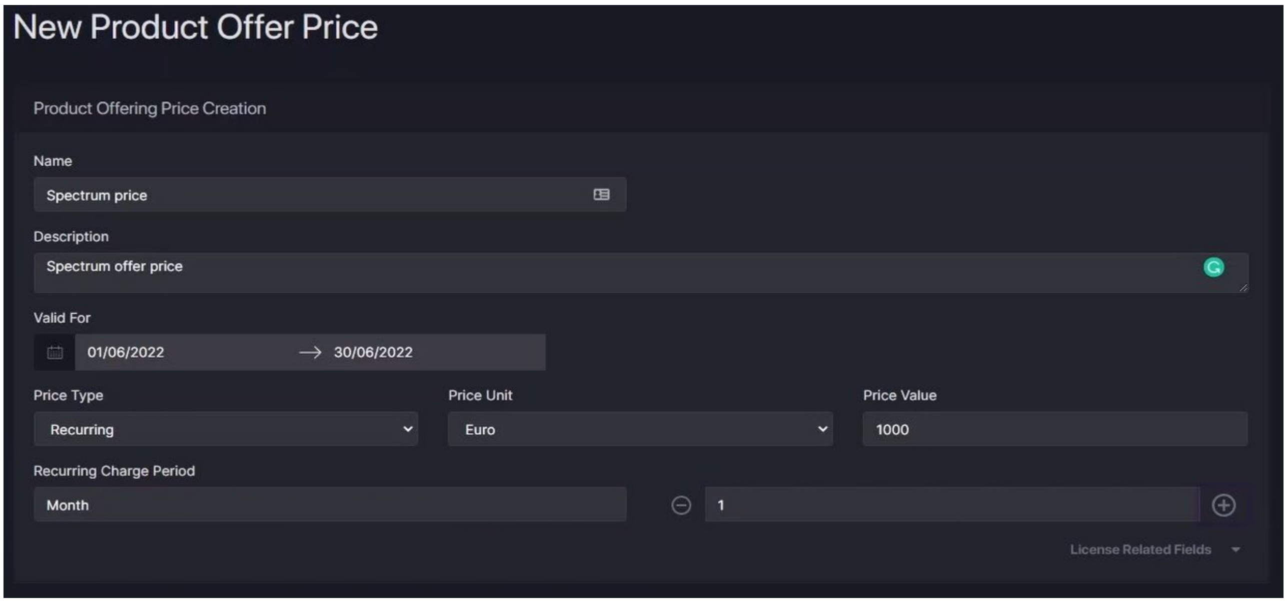
3.5.2.5. *Creation of offering prices*

A Product Offering Price is a document that defines a pricing mechanism for a service or resource attached to a product offering. Essentially, it adds a pricing strategy for the observed usage of the resource or service which can be configured to best describe the intended business model of the service provider. This allows vendors to configure the best approach to monetize their product:

- Flat rate is defined by a fixed price for a set of features of the product. Configured via a ONE TIME Price Type dropdown in the Product Offering Price view of the Portal.

- Pay-as-you-Grow is defined as a price that varies based on the usage of the product. Usage can be determined based on various metrics of the product like number of instances of a xNF or an application-level metric. Configured via a USAGE Price Type dropdown in the Product Offering Price view of the Portal.
- Subscriptions are similar to flat rates but the right to use the purchased product offer needs to be renewed at a known interval. Configured via a RECURRING Price Type dropdown in the Product Offering Price view of the Portal.

Figure 3-21 shows an example that is used for a spectrum offer, considered a subscription type of plan.


New Product Offer Price

Product Offering Price Creation

Name: Spectrum price

Description: Spectrum offer price

Valid For: 01/06/2022 → 30/06/2022

Price Type: Recurring

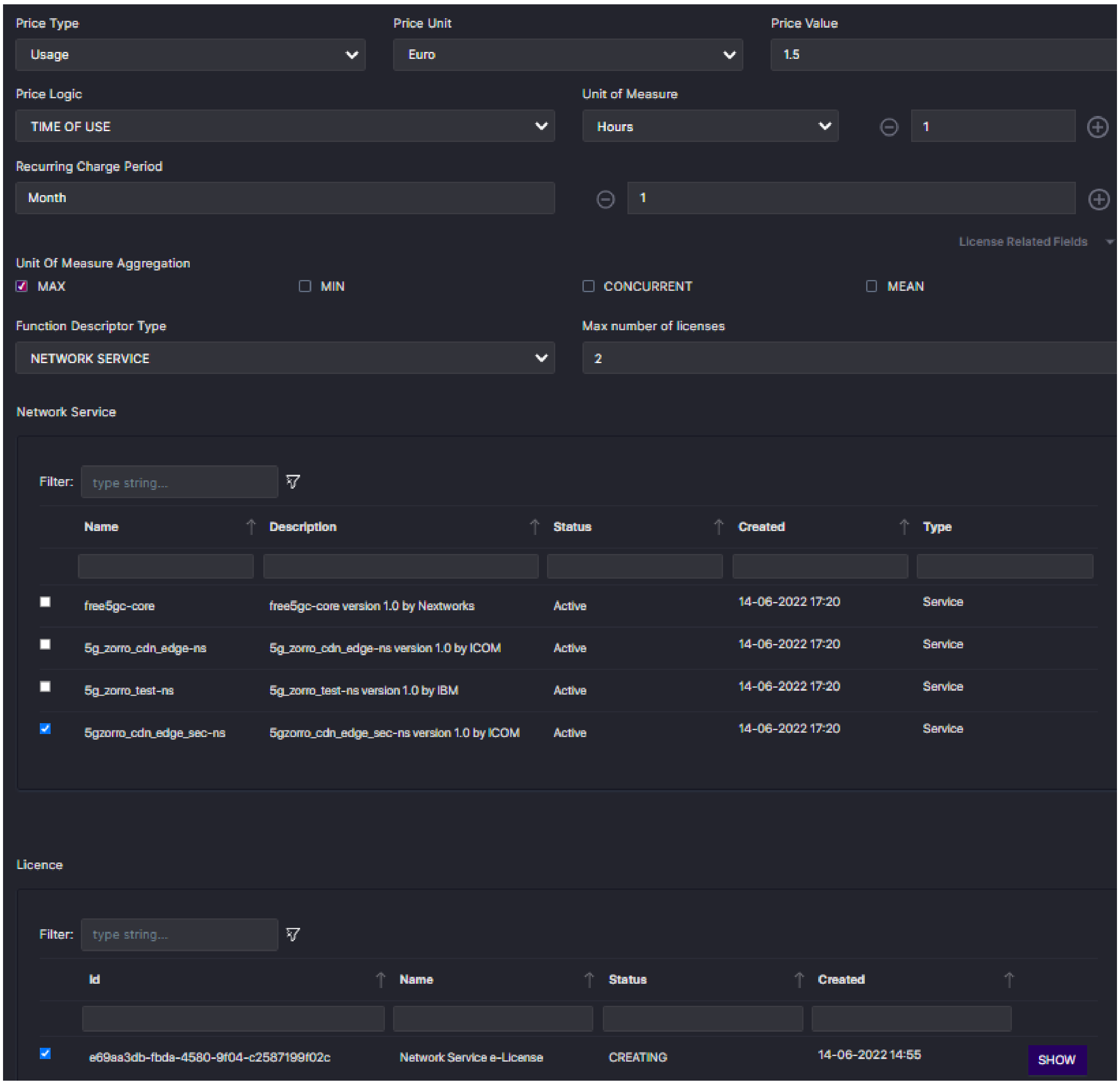
Price Unit: Euro

Price Value: 1000

Recurring Charge Period: Month 1

Figure 3-21: Creation of a Product Offering Price in the Portal

In the case of a software service or resource, the portal offers an additional section under the “License Related Fields” dropdown visible on the bottom right corner of Figure 3-21. These advanced fields show what kind of limits, if any, are applicable to the current transaction. Figure 3-22 shows the complete UI for the case of a network service which is configured based on a pay-as-you-grow plan. In this case, the network service (5gzorro_cdn_edge_sec-ns) usage is monitored based on the uptime of the service, one hour equals to 1.5€, billed monthly.



Price Type: Usage
Price Unit: Euro
Price Value: 1.5
Price Logic: TIME OF USE
Unit of Measure: Hours 1
Recurring Charge Period: Month 1

License Related Fields

Unit Of Measure Aggregation
[✓ MAX] MIN CONCURRENT MEAN

Function Descriptor Type: NETWORK SERVICE
Max number of licenses: 2

Network Service

Filter: type string...

	Name	Description	Status	Created	Type
☐	free5gc-core	free5gc-core version 1.0 by Nextworks	Active	14-06-2022 17:20	Service
☐	5g_zorro_cdn_edge-ns	5g_zorro_cdn_edge-ns version 1.0 by ICOM	Active	14-06-2022 17:20	Service
☐	5g_zorro_test-ns	5g_zorro_test-ns version 1.0 by IBM	Active	14-06-2022 17:20	Service
✓	5gzorro_cdn_edge_sec-ns	5gzorro_cdn_edge_sec-ns version 1.0 by ICOM	Active	14-06-2022 17:20	Service

Licence

Filter: type string...

	Id	Name	Status	Created	
✓	e69aa3db-fbda-4580-9f04-c2587199f02c	Network Service e-License	CREATING	14-06-2022 14:55	SHOW

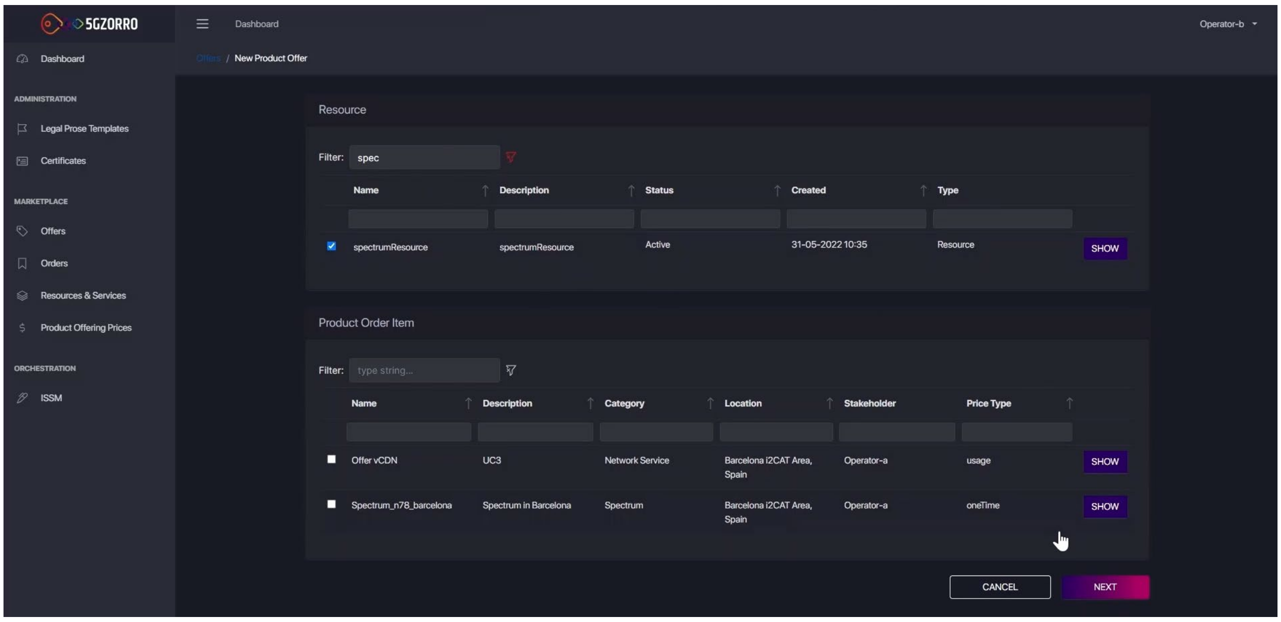
Figure 3-22: Creation of a Product Offering Price for a xNF

Figure 3-22 also shows how a software vendor can introduce limitations on the usage of the xNF by selecting a maximum number of instances. Lastly, an e-License is selected to define the agreement in a human readable document and to include any additional terms, conditions, purpose or definition. The creation of this e-License has already been shown in Figure 3-20.

3.5.2.6. Composition and dissemination of a product offer

When creating an offer, 5GZORRO stakeholders browse their available resources and services, being able to revise their technical specifications in the process. Upon selecting the corresponding resource or service specification to be offered over the Marketplace, providers need to indicate complementary information, such as name, description, location, category and validity period. Already created offering prices can be browsed and attached to the offer. Similarly, an existing SLA contract can be associated to the offer. With all this info assembled a creation request is forwarded from the Portal to the RSOC, which interacts with ID&P for the generation of a unique DID and with SRSD for the classification of offer. Subsequently this offer creation request is delegated to the SCLCM for its commitment into the Marketplace ledger and the dissemination of this new offer cross-domain towards all the Marketplace parties.

Figure 3-23 shows the first part of the aforementioned offer creation flow, in which the provider selects a spectrum resource for the composition of a spectrum type of offer.



Resource

Filter: spec

	Name	Description	Status	Created	Type	
✓	spectrumResource	spectrumResource	Active	31-05-2022 10:35	Resource	SHOW

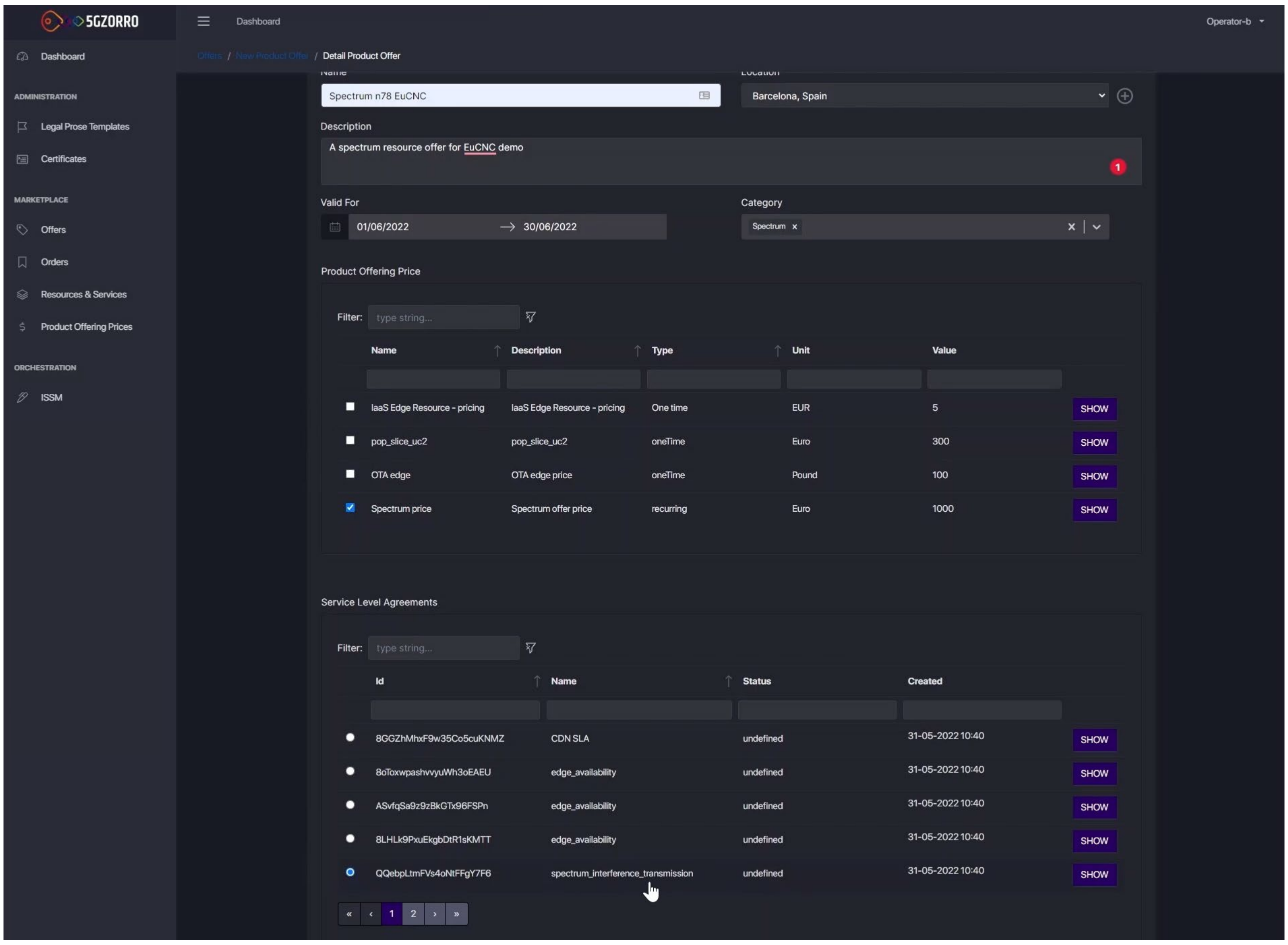
Product Order Item

Filter: type string...

	Name	Description	Category	Location	Stakeholder	Price Type	
	Offer vCDN	UC3	Network Service	Barcelona i2CAT Area, Spain	Operator-a	usage	SHOW
	Spectrum_n78_barcelona	Spectrum in Barcelona	Spectrum	Barcelona i2CAT Area, Spain	Operator-a	oneTime	SHOW

Figure 3-23: Selection of a spectrum resource for composing a spectrum product offer

The subsequent view of this flow is shown in Figure 3-24, in which general offer information is provided together with the selection of the offering price and the SLA contract to be associated with such spectrum offer.



Name: Spectrum n78 EuCNC

Location: Barcelona, Spain

Description: A spectrum resource offer for EuCNC demo

Valid For: 01/06/2022 → 30/06/2022

Category: Spectrum

Product Offering Price

Filter: type string...

	Name	Description	Type	Unit	Value	
☐	IaaS Edge Resource - pricing	IaaS Edge Resource - pricing	One time	EUR	5	SHOW
☐	pop_slice_uc2	pop_slice_uc2	oneTime	Euro	300	SHOW
☐	OTA edge	OTA edge price	oneTime	Pound	100	SHOW
☑	Spectrum price	Spectrum offer price	recurring	Euro	1000	SHOW

Service Level Agreements

Filter: type string...

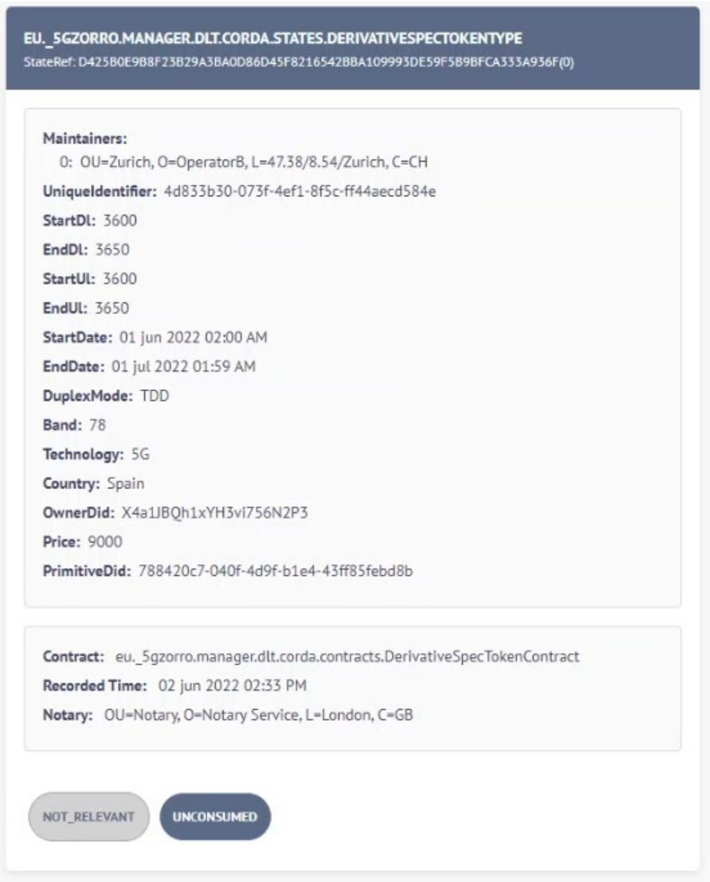
	Id	Name	Status	Created	
○	8GGZhMhxF9w35Co5cuKNMZ	CDN SLA	undefined	31-05-2022 10:40	SHOW
○	8oToxwpashvvyuWh3oEAEU	edge_availability	undefined	31-05-2022 10:40	SHOW
○	ASvfqSa9z9zBkGTx96FSPn	edge_availability	undefined	31-05-2022 10:40	SHOW
○	8LHLk9PxuEkgbDtR1sKMTT	edge_availability	undefined	31-05-2022 10:40	SHOW
●	QQebpLtmFVs4oNtFFgY7F6	spectrum_interference_transmission	undefined	31-05-2022 10:40	SHOW

« ‹ 1 2 › »

Figure 3-24: Associating product price and SLA to spectrum product offer

In the case of spectrum product offers, a Derivative SpectokenType is also created at the provider domain as part of the offer creation flow. This Derivative Spectoken stems from the Primitive Spectoken the spectrum product offer was bound to. As the offer creation request gets to the SCLCM, this module performs some validations on the spectrum product offer information against the associated Primitive Spectoken to ensure that the Derivative Spectoken information is contained within the ranges of the Primitive Spectoken. Once the SCLCM ensures that the information in the offer is correct, the Derivative SpectokenType is generated and the flow for offer creation can be completed. On the contrary, if the validation fails the offer creation is not performed returning an indicative error.

Figure 3-25 shows the Derivative SpectokenType information as it can be retrieved from the Marketplace DLT vault, containing the spectrum technical information that includes starting and ending frequencies, duplex mode, start and expiration dates of the license, radio access technology bounded to the band, and country; together with information specific to the generated SpectokenType such as its unique ID, its maintainer DLT node (i.e. the spectrum offer provider, which is this sample scenario is represented by the OperatorB), and its contract ID.



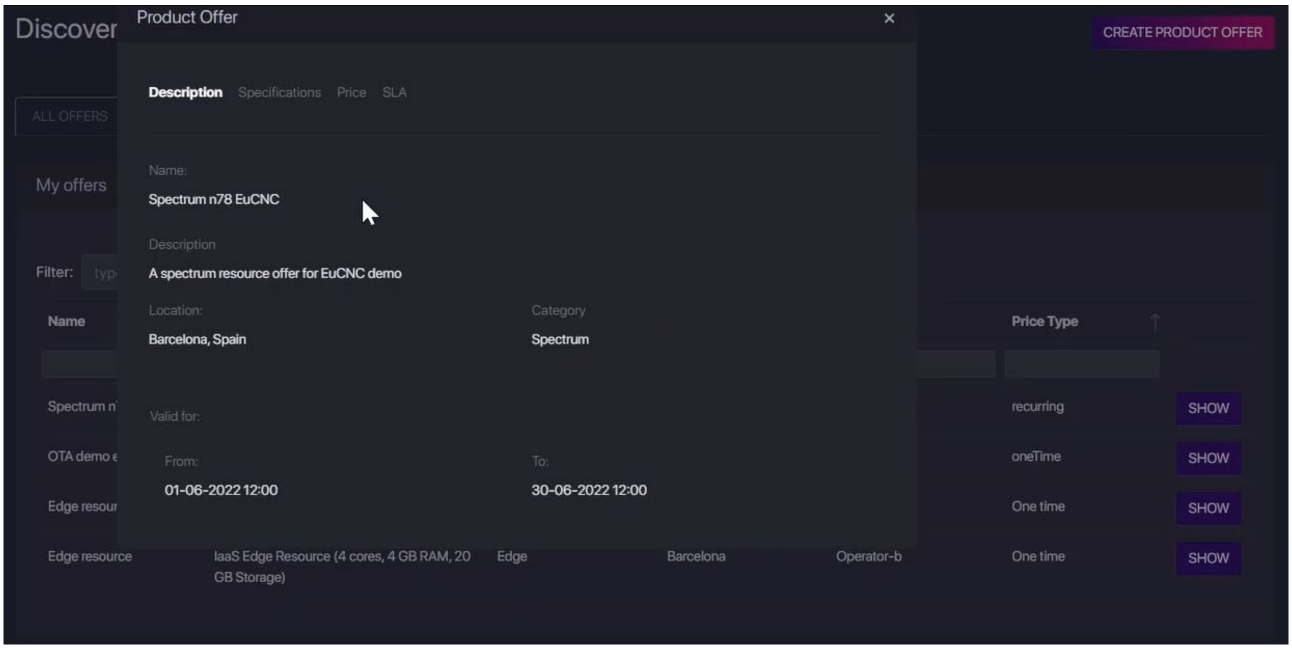
EU._5GZORRO.MANAGER.DLT.CORDA.STATES.DERIVATIVESPECTOKENTYPE
StateRef: D425B0E9B8F23B29A3BA0D86D45F8216542BBA109993DE59F5B9BFCA333A936F(0)

Maintainers:
0: OU=Zurich, O=OperatorB, L=47.38/8.54/Zurich, C=CH
UniqueIdentifier: 4d833b30-073f-4ef1-8f5c-ff44aecd584e
StartDl: 3600
EndDl: 3650
StartUl: 3600
EndUl: 3650
StartDate: 01 Jun 2022 02:00 AM
EndDate: 01 Jul 2022 01:59 AM
DuplexMode: TDD
Band: 78
Technology: 5G
Country: Spain
OwnerDid: X4a1JBQh1xYH3vi756N2P3
Price: 9000
PrimitiveDid: 788420c7-040f-4d9f-b1e4-43ff85febd8b

Contract: eu._5gzorro.manager.dlt.corda.contracts.DerivativeSpecTokenContract
Recorded Time: 02 Jun 2022 02:33 PM
Notary: OU=Notary, O=Notary Service, L=London, C=GB

Figure 3-25: Derivative SpectokenType information retrieved from the DLT vault

After the successful creation of a product offer, the details of the considered offer can be reviewed from the Portal. Figure 3-26, Figure 3-27 and Figure 3-28 depict the information provided in the different tabs view for the created product offer.

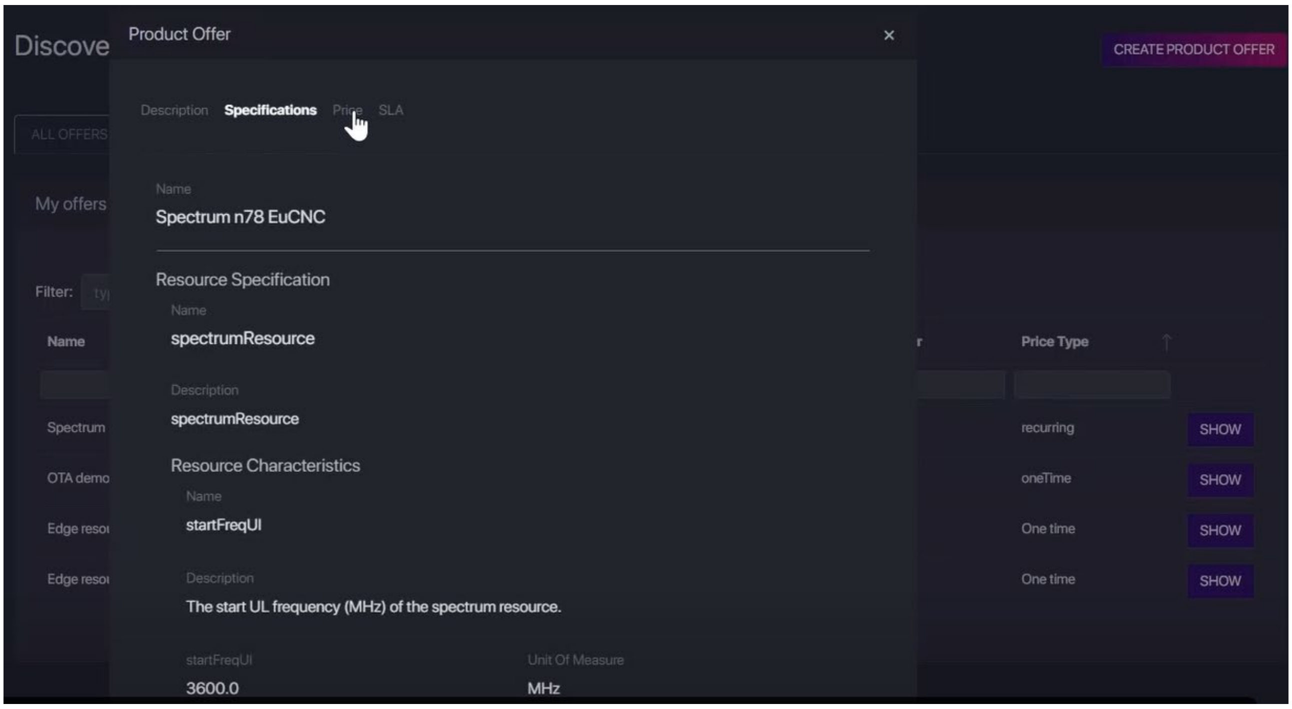


Product Offer

Name: Spectrum n78 EuCNC
Description: A spectrum resource offer for EuCNC demo
Location: Barcelona, Spain
Category: Spectrum
Valid for:
From: 01-06-2022 12:00
To: 30-06-2022 12:00

Spectrum n					recurring	SHOW
OTA demo e					oneTime	SHOW
Edge resour					One time	SHOW
Edge resource	IaaS Edge Resource (4 cores, 4 GB RAM, 20 GB Storage)	Edge	Barcelona	Operator-b	One time	SHOW

Figure 3-26: Details of the created spectrum product offer under the tab for general description


Product Offer

Description **Specifications** Price SLA

Name
Spectrum n78 EuCNC

Resource Specification

Name
spectrumResource

Description
spectrumResource

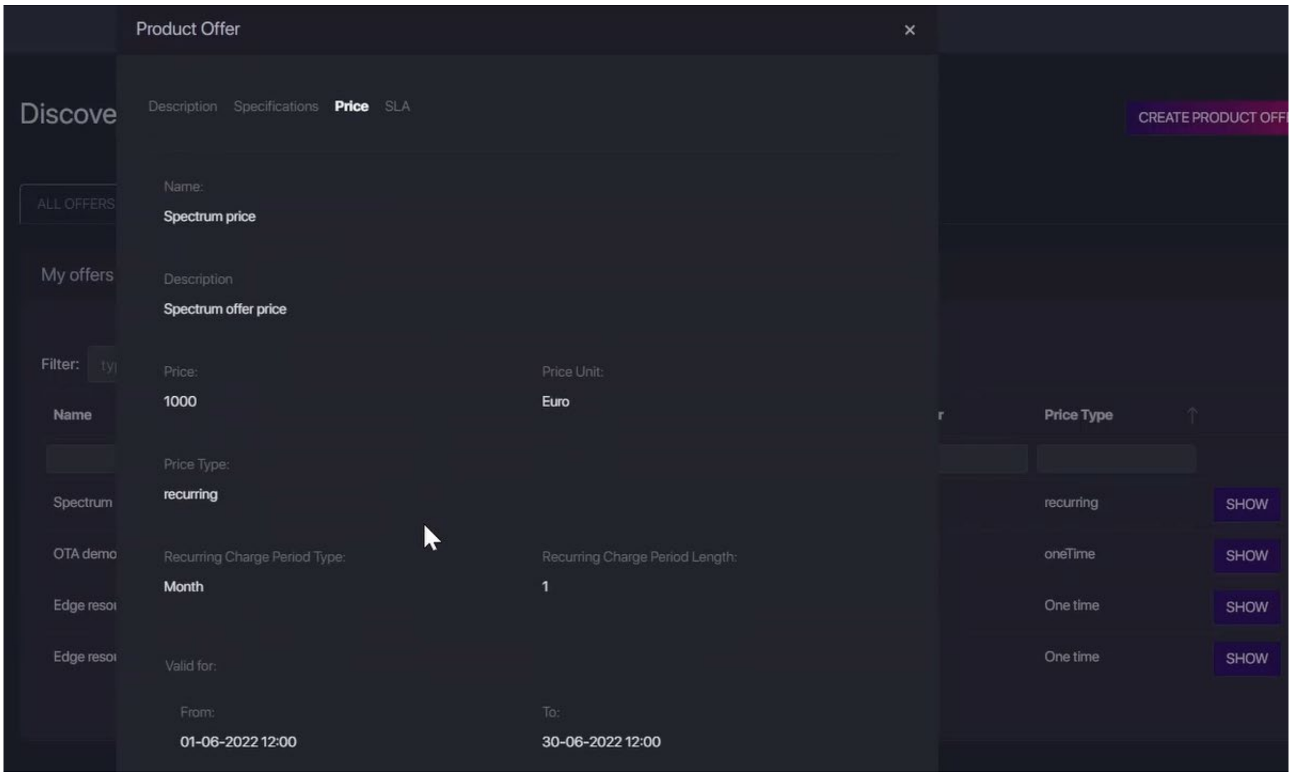
Resource Characteristics

Name
startFreqUl

Description
The start UL frequency (MHz) of the spectrum resource.

startFreqUl	Unit Of Measure
3600.0	MHz

Figure 3-27: Details of the created spectrum product offer under the tab for resource specification


Product Offer

Description Specifications **Price** SLA

Name:
Spectrum price

Description
Spectrum offer price

Price:	Price Unit:
1000	Euro

Price Type:
recurring

Recurring Charge Period Type:	Recurring Charge Period Length:
Month	1

Valid for:

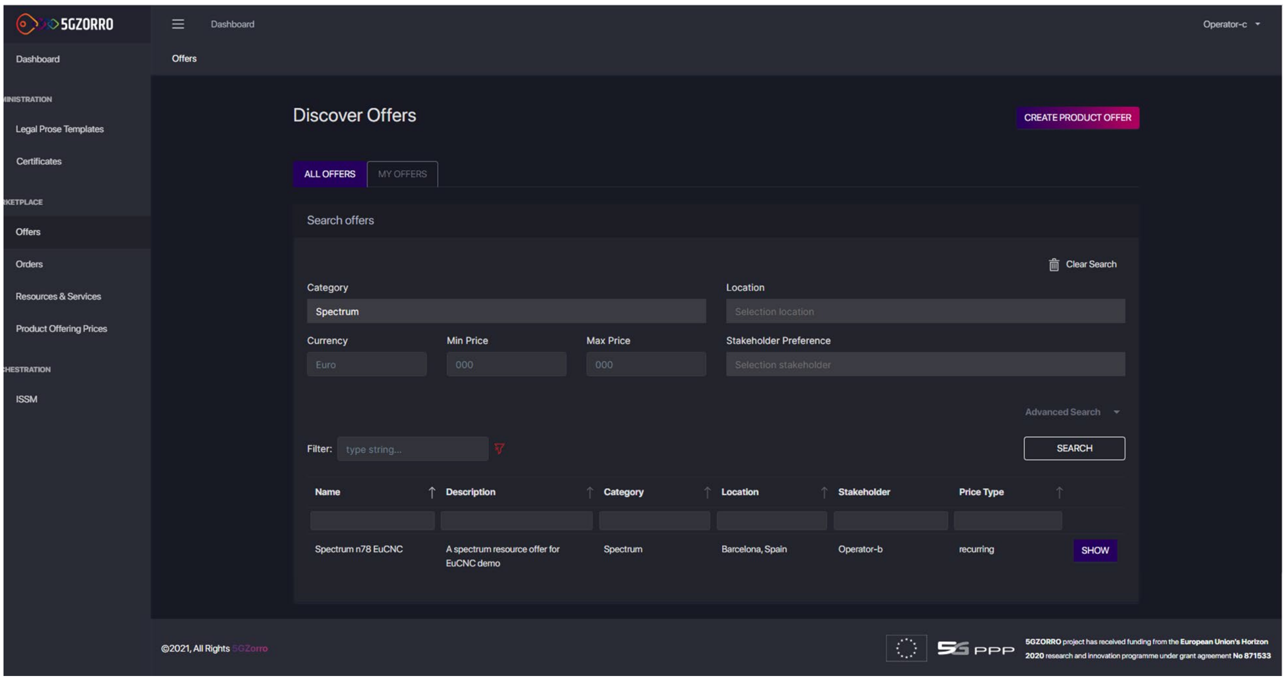
From:	To:
01-06-2022 12:00	30-06-2022 12:00

Figure 3-28: Details of the created spectrum product offer under the tab for price

The successful creation of a product offer also implies the execution of additional integration actions that are performed by the platform under the hood. This is the case of the generation of a unique DID assigned by ID&P to the offers and its online classification performed by the SRSD, module that is further detailed in Section 4.5.

Likewise, once offers are created, they are disseminated cross-domain via a combination of DLT updates and events notifications in the Kafka bus, representing the Intra-Domain Communication Fabric, pushed by the

SCLCM, to which the different RSOC instances are subscribed and act upon in order to populate their databases with the new offer information. To corroborate this procedure, Figure 3-29 shows the same Spectrum Offer but from the Portal view of another party, in this case the Operator-c. Note that in the performed search, a filter is indicated from the Portal to discover only available offers with the Spectrum Category, which is implemented at the RSOC.



Discover Offers

Search offers

Category: Spectrum
Location: Selection location
Currency: Euro
Min Price: 000
Max Price: 000
Stakeholder Preference: Selection stakeholder
Filter: type string...

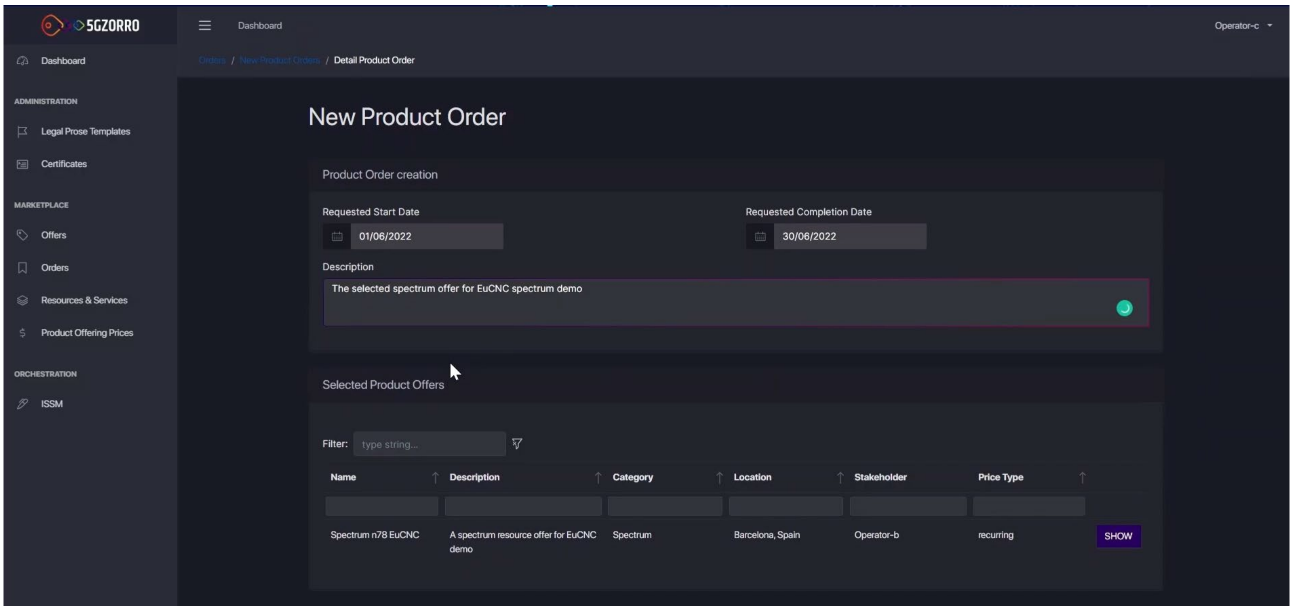
Name	Description	Category	Location	Stakeholder	Price Type
Spectrum n78 EuCNC	A spectrum resource offer for EuCNC demo	Spectrum	Barcelona, Spain	Operator-b	recurring

5GZORRO project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871533

Figure 3-29: Spectrum product offer retrieved from the Operator-c Portal

3.5.2.7. *Ordering of published product offer*

At the consumer domain, published offers can be discovered as previously showcased and subsequently ordered. After performing a selection, the consumer places a request specifying the corresponding offer together with the order description and requested offer consumption period. This procedure is depicted in Figure 3-30.



New Product Order

Product Order creation

Requested Start Date: 01/06/2022
Requested Completion Date: 30/06/2022
Description: The selected spectrum offer for EuCNC spectrum demo

Selected Product Offers

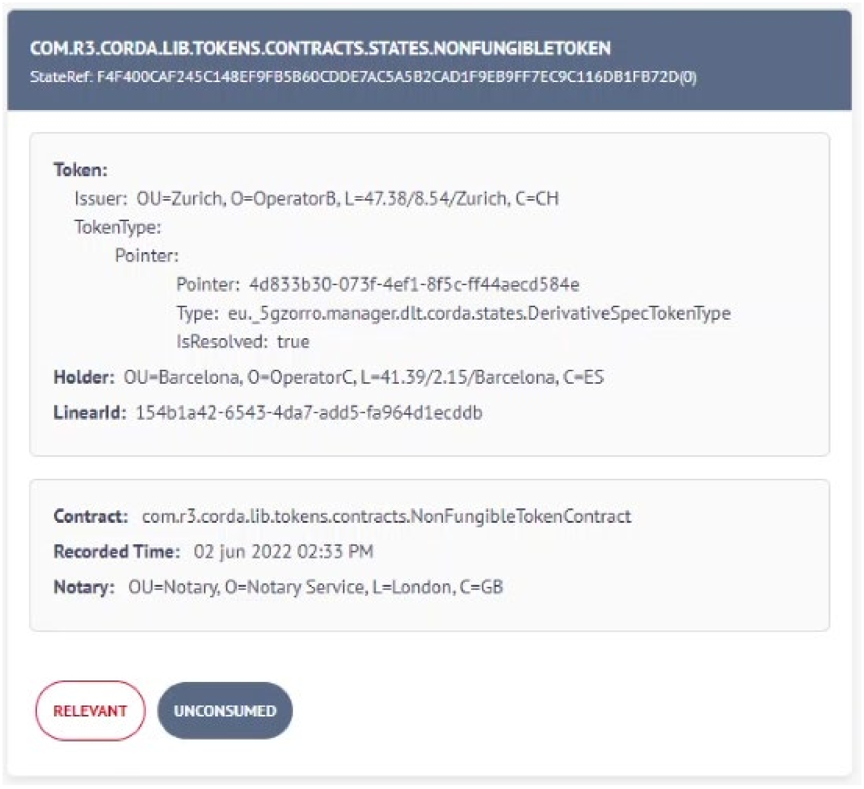
Filter: type string...

Name	Description	Category	Location	Stakeholder	Price Type
Spectrum n78 EuCNC	A spectrum resource offer for EuCNC demo	Spectrum	Barcelona, Spain	Operator-b	recurring

Figure 3-30: Spectrum product order creation view

Unlike product offers, the orders are only disseminated on a need-to-know basis, which means that in addition to the consumer the order is only retrieved and persisted at the RSOC of the provider domain, from where it can be fetched by the Portal. Likewise, in the particular case of spectrum related orders, also the Regulator domain is informed about the new order.

In the case of ordering a spectrum offer, a Derivative Spectoken NFT is issued to the ordering party. This Derivative Spectoken NFT relates to the Derivative SpectokenType associated to the spectrum product offer. Figure 3-31 shows the Derivative Spectoken NFT information as it can be retrieved from the Marketplace DLT vault. We can see that it contains a reference to the associated Derivative SpectokenType and to the holder DLT node (i.e. the spectrum consumer, which is this sample scenario is represented by the OperatorC).


COM.R3.CORDA.LIB.TOKENS.CONTRACTS.STATES.NONFUNGIBLETOKEN
StateRef: F4F400CAF245C148EF9FB5B60CDDE7AC5A5B2CAD1F9EB9FF7EC9C116DB1FB72D(0)

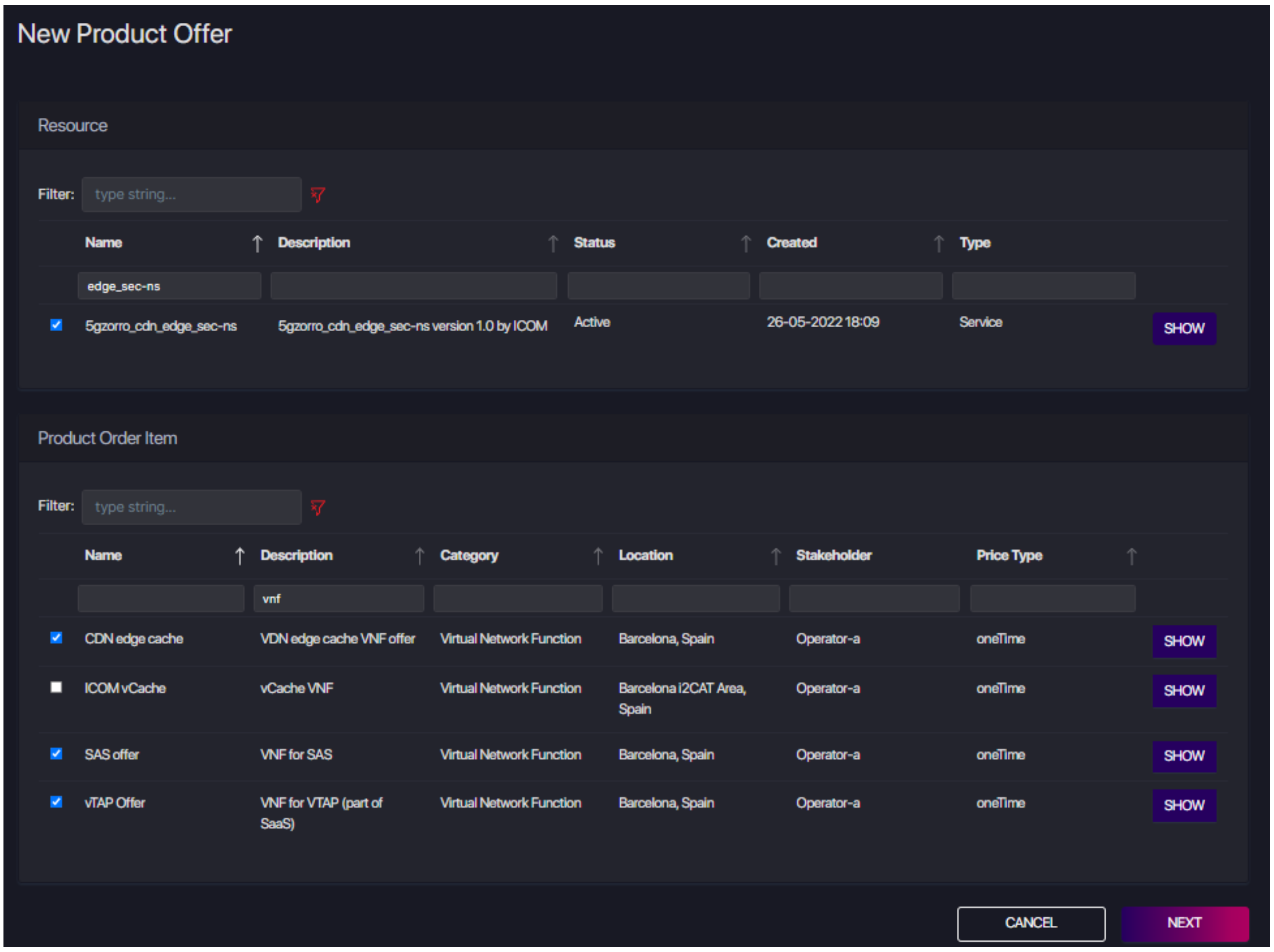
```
Token:
   Issuer:  OU=Zurich, O=OperatorB, L=47.38/8.54/Zurich, C=CH
   TokenType:
      Pointer:
            Pointer:  4d833b30-073f-4ef1-8f5c-ff44aecd584e
            Type:  eu._5gzorro.manager.dlt.corda.states.DerivativeSpecTokenType
            IsResolved:  true
Holder:  OU=Barcelona, O=OperatorC, L=41.39/2.15/Barcelona, C=ES
LinearId:  154b1a42-6543-4da7-add5-fa964d1ecddb
```

```
Contract:   com.r3.corda.lib.tokens.contracts.NonFungibleTokenContract
Recorded Time:   02 jun 2022 02:33 PM
Notary:   OU=Notary, O=Notary Service, L=London, C=GB
```

Figure 3-31: Derivative Spectoken NFT information retrieved from the DLT vault

3.5.2.8. Composition of a Network Service Offer based on previously ordered VNFs

A multi-stakeholder marketplace such as the one offered by 5GZORRO introduces the ability to compose complex offers of services that leverage on resources and services from other providers. Relying on the relationship between orders and offers and the ability to compose bundled offers, the portal allows a way to create, for example, an offer for a network service which is built as the combination of previously ordered VNFs. This process is shown in Figure 3-32 and Figure 3-33 which leverage the fact that the existence of an order means that there is already a smart contract in place that governs the agreement between the VNF provider and the network service provider. Thus, the network service provider can compose a new offer with its own specification, product offering price and e-license to be later used in an order transaction with an end service consumer.


New Product Offer

Resource

Filter: type string...

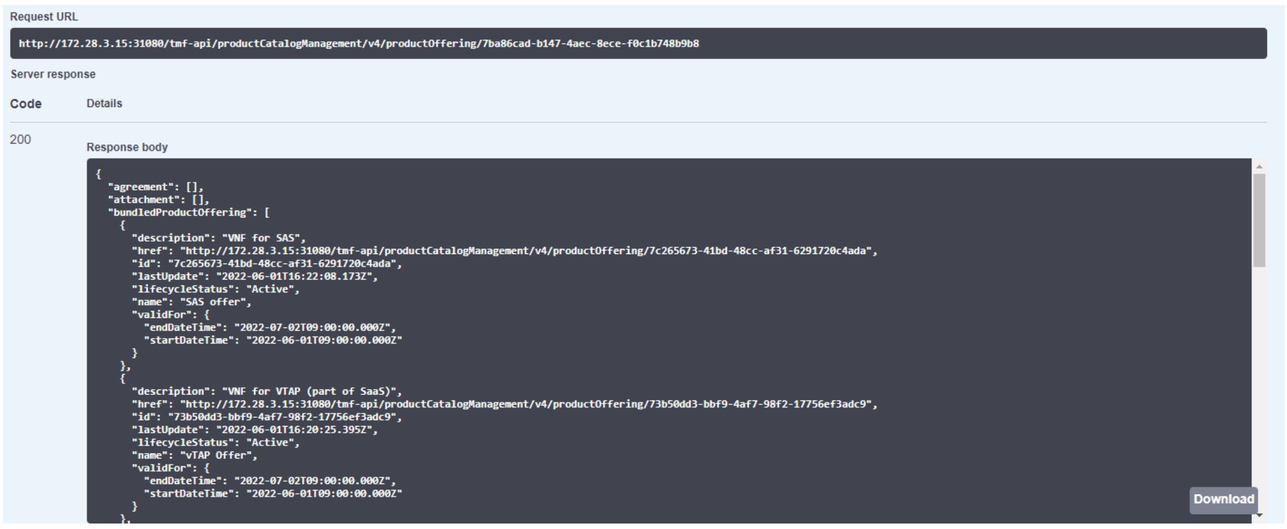
	Name	Description	Status	Created	Type	
	edge_sec-ns					
✓	5gzorro_cdn_edge_sec-ns	5gzorro_cdn_edge_sec-ns version 1.0 by ICOM	Active	26-05-2022 18:09	Service	SHOW

Product Order Item

Filter: type string...

	Name	Description	Category	Location	Stakeholder	Price Type	
		vnf					
✓	CDN edge cache	VDN edge cache VNF offer	Virtual Network Function	Barcelona, Spain	Operator-a	oneTime	SHOW
	ICOM vCache	vCache VNF	Virtual Network Function	Barcelona i2CAT Area, Spain	Operator-a	oneTime	SHOW
✓	SAS offer	VNF for SAS	Virtual Network Function	Barcelona, Spain	Operator-a	oneTime	SHOW
✓	vTAP Offer	VNF for VTAP (part of SaaS)	Virtual Network Function	Barcelona, Spain	Operator-a	oneTime	SHOW

Figure 3-32: Creation of bundled Product Offer


Request URL

```
http://172.28.3.15:31080/tmf-api/productCatalogManagement/v4/productOffering/7ba86cad-b147-4aec-8ece-f0c1b748b9b8
```

Server response

Code	Details
200	Response body

```
{
  "agreement": [],
  "attachment": [],
  "bundledProductOffering": [
    {
      "description": "VNF for SAS",
      "href": "http://172.28.3.15:31080/tmf-api/productCatalogManagement/v4/productOffering/7c265673-41bd-48cc-af31-6291720c4ada",
      "id": "7c265673-41bd-48cc-af31-6291720c4ada",
      "lastUpdate": "2022-06-01T16:22:08.173Z",
      "lifecycleStatus": "Active",
      "name": "SAS offer",
      "validFor": {
        "endDateTime": "2022-07-02T09:00:00.000Z",
        "startDateTime": "2022-06-01T09:00:00.000Z"
      }
    },
    {
      "description": "VNF for VTAP (part of SaaS)",
      "href": "http://172.28.3.15:31080/tmf-api/productCatalogManagement/v4/productOffering/73b50dd3-bbf9-4af7-98f2-17756ef3adc9",
      "id": "73b50dd3-bbf9-4af7-98f2-17756ef3adc9",
      "lastUpdate": "2022-06-01T16:20:25.395Z",
      "lifecycleStatus": "Active",
      "name": "vTAP Offer",
      "validFor": {
        "endDateTime": "2022-07-02T09:00:00.000Z",
        "startDateTime": "2022-06-01T09:00:00.000Z"
      }
    },
```

Figure 3-33: Bundled Product Offerings in the RSOC

4. Cross-domain Analytics & Intelligence for AIOps Platform Prototypes

The 5GZORRO Cross-domain Analytics & Intelligence for AIOps Platform is responsible for realizing the envisioned AIOps for AI-driven operation of 5G network resources, slices, and services as described D3.3 [2]. 5GZORRO Analytics & Intelligence for AIOps Platform Prototype described in this deliverable is realised as a set of software modules that together provide services to facilitate data-driven zero-touch automation throughout the 5GZORRO environment.

As shown in Figure 4-1, that represents the AIOPs Platform Architecture, some software modules, e.g., the Monitoring Data Aggregator or MDA, logically belong to specific Provider's domain and must be deployed and operated by their owning Provider. Other modules, such as the Operational Datalake Platform and the Cross Domain Analytics and Intelligence Services are logically centralized and belong to 5GZORRO platform as a whole, while the deployments can differ so that each provider runs and operates a local copy of the cross-platform service for reasons of redundancy and high availability. These software modules communicate over 5GZORRO communication fabric (mainly Kafka) and also rely on other cross platform services described in this document, the Governance Platform Services to access identities and permissions and the Marketplace Platform to access details of various marketplace objects such as stakeholder, offers, SLAs, etc.

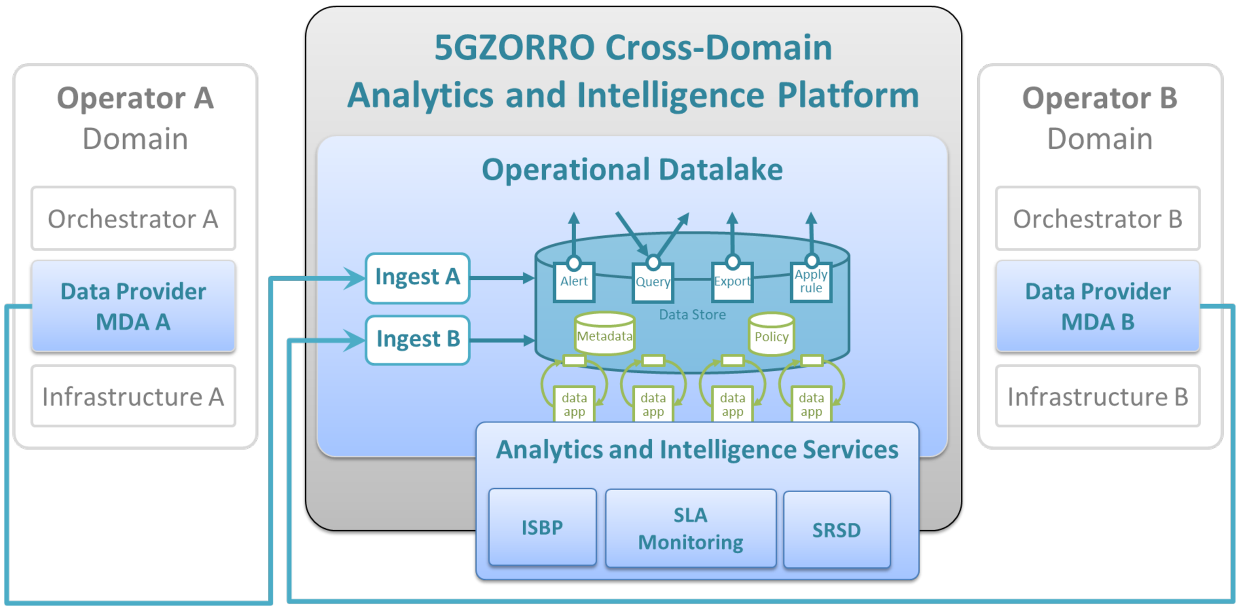


Figure 4-1: 5GZORRO Cross-domain Analytics & Intelligence for AIOps Platform Architecture

In what follows this section describes the software modules implemented as part of Cross-domain Analytics & Intelligence for AIOps Platform Prototypes, namely the Operational Data Lake platform in section 4.1, the Monitoring Data Aggregator Service in section 4.2, the SLA Monitoring Service in section 4.3, the Intelligent SLA Breach Presictor Service in section 4.4, and the Smart Resource and Service Discovert Application in section 4.5. In addition, section 4.6 covers the validation tests performed as part of platfrom integration efforts.

4.1. **Operational Data Lake**

We use the term *Operational Data Lake* to include the Data Store and the surrounding tools to perform all the required data processing operations according to the AIOps approach.

As shown in the following figure, 5GZORRO Operational Data Lake is composed of multiple building blocks, described below:

1. **Data Storage** for storing the data; for simplicity we choose to use the object storage system for all the data types.
2. **Messaging and streaming** capabilities for communication across different components.
3. **Metadata** service for annotating the data and imposing the common domain specific data model.
4. **Analytics** tools to support data scientists in their work creating and training domain specific models, feature engineering, etc.
5. **Data Transformation** tools to efficiently process large amounts of data.
6. **API service** to support data providers that create new data, data consumers that consume data, create new datasets, and operators that manage data processing workflows.
7. **Runtime** system to host all the components; we use a cloud native runtime enriched with advanced capabilities for workflow management, eventing, and serverless execution.

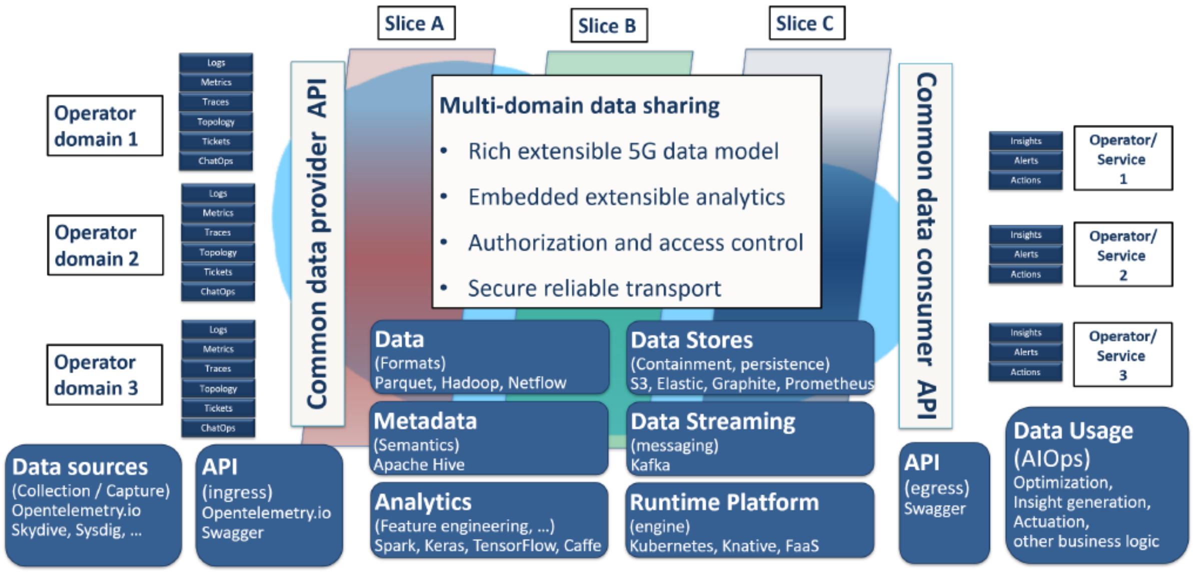

Slice A
Slice B
Slice C
Operator domain 1
Operator domain 2
Operator domain 3
Logs
Metrics
Traces
Topology
Tickets
ChatOps
Common data provider API
Multi-domain data sharing
Rich extensible 5G data model
Embedded extensible analytics
Authorization and access control
Secure reliable transport
Common data consumer API
Insights
Alerts
Actions
Operator/ Service 1
Operator/ Service 2
Operator/ Service 3
Data (Formats) Parquet, Hadoop, Netflow
Data Stores (Containment, persistence) S3, Elastic, Graphite, Prometheus
Metadata (Semantics) Apache Hive
Data Streaming (messaging) Kafka
Analytics (Feature engineering, ...) Spark, Keras, TensorFlow, Caffe
Runtime Platform (engine) Kubernetes, Knative, FaaS
Data sources (Collection / Capture) Opentelemetry.io Skydive, Sysdig, ...
API (ingress) Opentelemetry.io Swagger
API (egress) Swagger
Data Usage (AIOps) Optimization, Insight generation, Actuation, other business logic

Figure 4-2: 5GZORRO Operational Data Lake

The following specific choices were made to implement the Operational Datalake Prototype in 5GZORRO.

For runtime, we employ k8s [18], an open-source system for automating deployment, scaling, and management of containerized applications and Argo Workflows [28], an open-source container-native workflow engine for orchestrating parallel jobs on k8s.

For Data Storage, we use Minio [29], an open-source software storage platform which implements object storage on a single distributed computer cluster.

For messaging and streaming, we use Apache Kafka [30], a distributed streaming platform for publishing and subscribing records as well as storing and processing streams of records. This is used as part of the 5GZORRO communication fabric.

For data transformation and analytics, we use Apache Hadoop [26], a framework that allows for the distributed processing of large datasets across clusters of computers using simple programming models and Apache Spark™ [27], a unified analytics engine for large-scale data processing.

For metadata service, we have created a simple custom data catalogue component sufficient for demonstrating capabilities required by 5GZORRO use cases. We use a PostgreSQL [15] database to store the index of the metadata.

For the API service, we expose the URLs of the Data Lake components and provide additional API server endpoints to serve the Data Lake specific APIs (details below).

4.1.1. Main Functionalities

In 5GZORRO, we have the following specific use case:

1. Ingest monitoring data.
2. Provide services for easy consumption of ingested monitoring data.
3. Perform some analytics to detect or predict violation of SLA.
4. Perform actions to be performed upon detection or prediction of SLA violation.

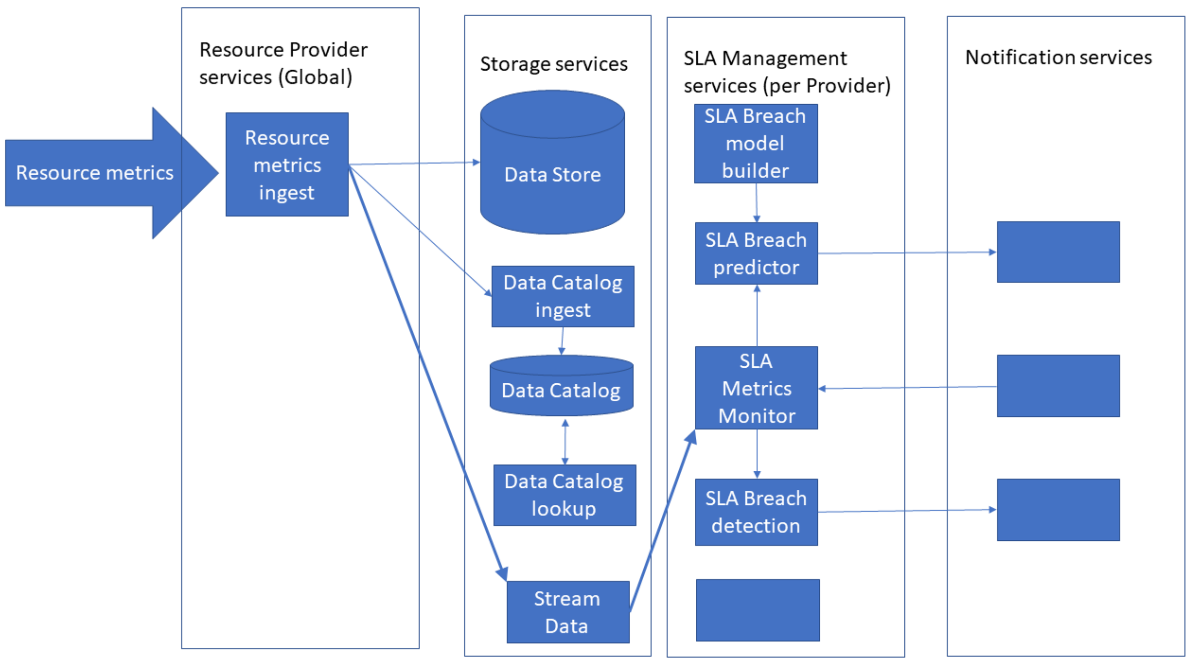


Figure 4-3: Datalake Usage Example

We implemented several services specific to the 5GZORRO environment to support the above scenario.

- Data Ingestion
- Data Lookup
- Data streaming

Metrics data is sent to Data Ingestion Service. This data is stored directly in the Object Store for future reference. In parallel, the data is sent to the Data Catalog/Lookup Service to be indexed so that specific data can be found in the future based on various identifiers (such as productID). An additional data path is provided by the Data Streaming Service, in which users of the Data Lake may receive filtered data (based or productID) in real-time, as the data arrives. The SLA Monitoring and ISBP use this streamed data, in real-time, to detect or predict an imminent SLA breach. These services are discussed in more detail in the following subsections.

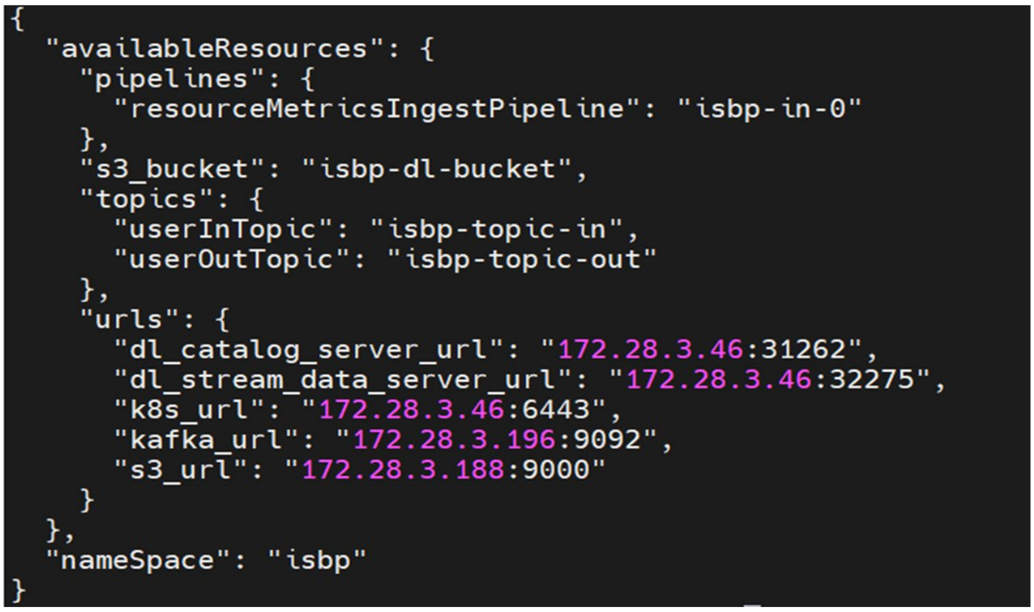
4.1.2. Prototype Implementation

4.1.2.1. Datalake Server

The Datalake Server is implemented as a Swagger/OpenAPI REST API in python. The main operations supported by the Datalake Server are:

- Create, Get, Delete a user (a 5GZORRO stakeholder or operator)
- Create, Get, Delete a pipeline
- Create, Get, Delete a service

A user of the Datalake services may be any 5GZORRO stakeholder, including Operators, Service Providers, Infrastructure Providers, etc. Once a user is registered with the Datalake Server, the user is provided with a list of predefined services available in the Datalake, and the user may then manage pipelines and services on top of the Datalake infrastructure. The full API is specified in reference [31]. Upon user registration, typical information returned looks like depicted in Figure 4-4.



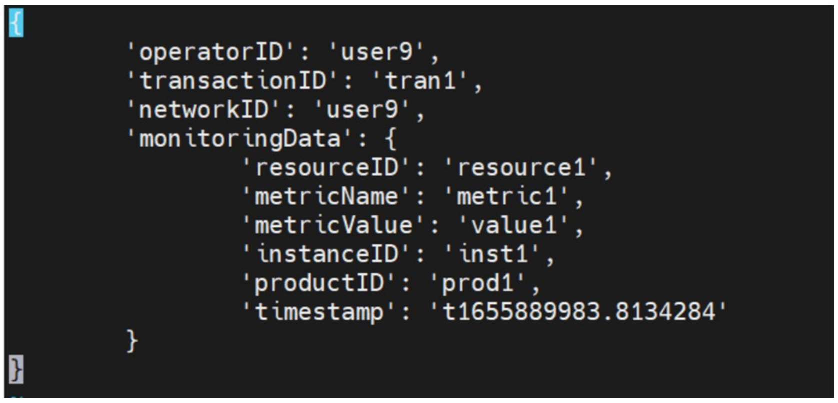
```
{
  "availableResources": {
    "pipelines": {
      "resourceMetricsIngestPipeline": "isbp-in-0"
    },
    "s3_bucket": "isbp-dl-bucket",
    "topics": {
      "userInTopic": "isbp-topic-in",
      "userOutTopic": "isbp-topic-out"
    },
    "urls": {
      "dl_catalog_server_url": "172.28.3.46:31262",
      "dl_stream_data_server_url": "172.28.3.46:32275",
      "k8s_url": "172.28.3.46:6443",
      "kafka_url": "172.28.3.196:9092",
      "s3_url": "172.28.3.188:9000"
    }
  },
  "nameSpace": "isbp"
}
```

Figure 4-4: Datalake user registration API response

Each user is provided with a pair of (Kafka) *topics* (userInTopic and userOutTopic) which he may use for any kind of communication. There is a predefined pipeline to ingest data to be stored in the Datalake, and the user is provided with a (Kafka) topic to which the data may be sent for ingestion (resourceMetricsIngestPipeline). Upon ingestion, the data is indexed and is then stored in a user-specific bucket in object store. The various services available from the Datalake are returned to the user in the form of URLs. These include the addresses of the k8s cluster, the Kafka server, the S3 object store, as well as other services that the Datalake provides. Source code is available at [32].

4.1.2.2. Data Ingestion Pipeline

The Data Ingestion Pipeline receives json data on its Kafka topic in the following format.



```
{
        'operatorID': 'user9',
        'transactionID': 'tran1',
        'networkID': 'user9',
        'monitoringData': {
                'resourceID': 'resource1',
                'metricName': 'metric1',
                'metricValue': 'value1',
                'instanceID': 'inst1',
                'productID': 'prod1',
                'timestamp': 't1655889983.8134284'
        }
}
```

Figure 4-5: Example of Data received by Data Ingestion Pipeline

The module verifies the proper structure of the provided data and places the data in the object store location (bucket). The data is then output to be processed by the next stage in the pipeline, which performs indexing of the data, so that it may be easily located. This module is written in python and is packaged as a Docker container. Source code is available at [33].

4.1.2.3. Data Lookup Service

The index of ingested data is stored in a local PostgreSQL database and allows lookup based on the various fields: productID, resourceID, instanceID, transactionID. Given an ID, the server returns a list of matching monitoring data entries and their locations in the object store. The full API is available and described at [34]. The Data Lookup Service is implemented as a Swagger/OpenAPI REST API in python and is deployed as a Docker container on the k8s cluster. Source code is available at [35].

4.1.2.4. Data Streaming Service

It is sometimes desirable (as for the ISBP module to detect an imminent SLA breach) to receive the monitoring data is received, rather than extracting history from the Datalake Lookup service. This is provided by the Data Streaming Service (DSS).

The DSS is implemented as a Swagger/OpenAPI REST API in python and is deployed as a Docker container on the k8s cluster. Its API allows the registering and unregistering a module to receive streamed data based on a specified productID. Whenever monitoring data with the specified productID is detected, that monitoring data is streamed directly to a (Kafka) topic provided by the registered module. In this way the registered module receives exactly those monitoring data items that are of interest in a timely manner.

The full API is at [36]. Source code is available at [37].

4.2. **Monitoring Data Aggregator**

The Monitoring Data Aggregator (MDA) is a module that has emerged to have all related data fetching, processing, synthetization, and aggregation functionalities into a single, dedicated intelligent functional block, starting from the initially conceived "Data Ingestion and Transformation Application" module design concept.

MDA is responsible for collecting data provided by each Resource and Service Provider of the platform for the resources and services that are controlled by said Operator. This data is then aggregated and stored in the Data Lake in a suitable manner to perform the desired analytics to keep track of resource usage and to predict/track SLA violations.

4.2.1. Main Functionalities

The MDA component is characterized by having the following main functionalities:

- Collect provided data by a Resource/Service Provider.
- Enable aggregation of data in customizable ways, according to a received configuration.
- Encrypt data to be sent by using signatures.
- Push monitoring data over time (if specified) towards the Data Lake, to enable monitoring operations made by other components.

One of the main workflows where the MDA has a key participation, is in regard to SLA Monitoring and SLA breach detection/prediction. The following Figure 4-6, showcases the interactions the MDA module has with some 5GZORRO components, and the operations that it does, in the context of the SLA Breach Detection scenario.

A configuration containing the desired metrics to be processed is performed by the Network Slice and Service Orchestration (NSSO) component. Then, monitoring data is collected from the xRM for each relevant resource and service. Afterwards, the data is aggregated over time and, after storing it on the Data Lake, it's used to detect or predict possible SLA violation.

The interfaces and software components must provide the aggregated data and the ability to match the metrics from a provided resource to the SLA it supports. For that, the ISSM receives a new product offer instantiation request from actor, then it requests the resource instantiation from the NSSO. The NSSO will then trigger the whole workflow that was previously described.

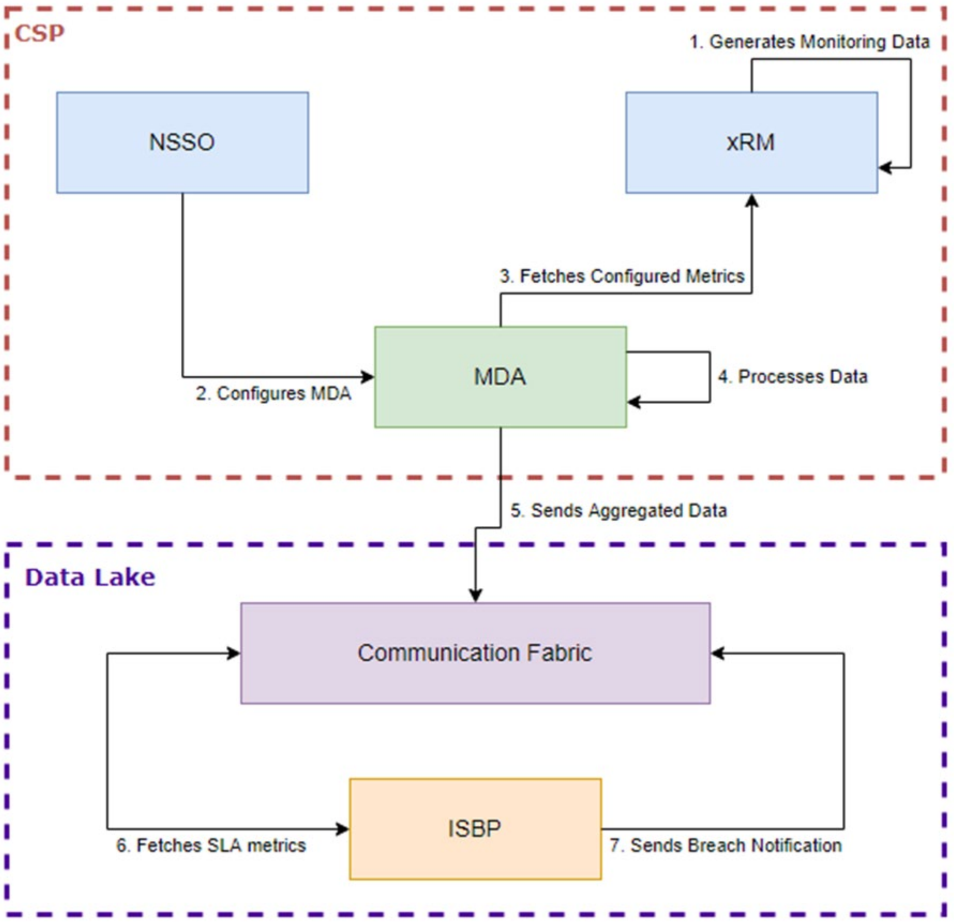


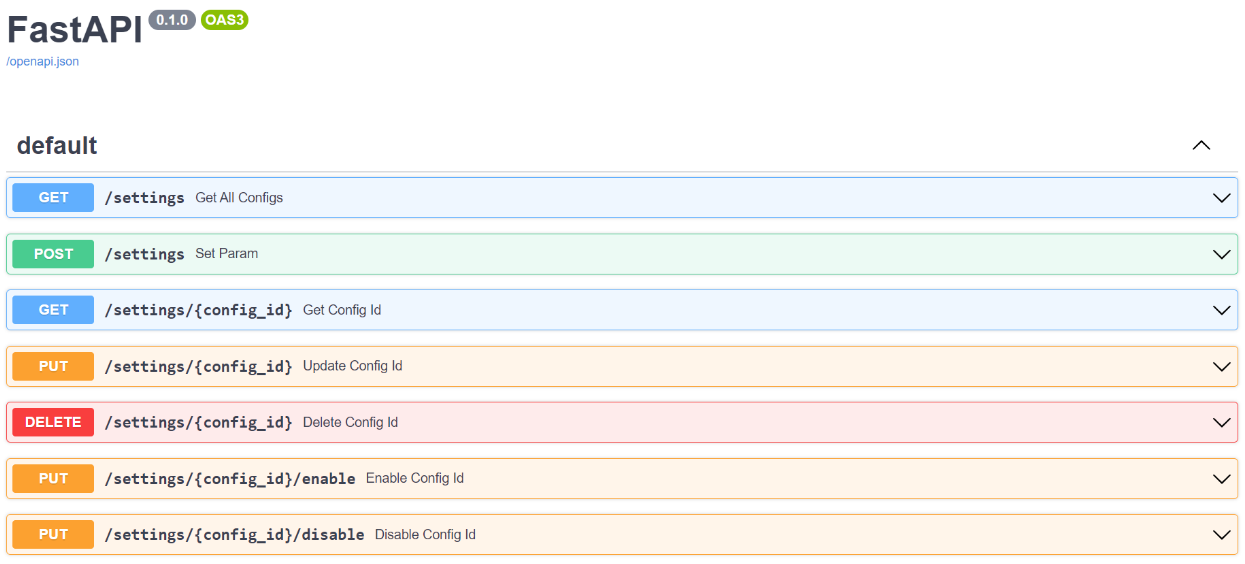
Figure 4-6: MDA applied to SLA Breach Detection

4.2.2. Prototype Implementation

In this section are presented and described the main interfaces for management of monitoring configurations, and respective information models.

The module source code, in regard to the technical implementation, is available online at [38]. Moreover, it includes the GitHub workflows for an automatic update of the component packages, the used OpenAPI, and some 'dummy' components for testing the workflows locally. Deployment files related to Docker and k8s deployments are also available. Inside GitHub's wiki, there's the 'Deploy MDA per Operator' page, where it's described how to deploy the MDA per operator in k8s, using a script.

MDA relies on FastAPI's OpenAPI support to expose the API REST Controller and its provided services to be used primarily by the NSSO. The interactive endpoints can be overviewed by accessing the root of the hosted module, as shown in Figure 4-7. These operations are related with configs management by NSSO, like *startAggregateMetric*, *stopAggregateMetric* and *lookupMetricsByReference*. The other functionalities, like *postMonitoringData*, are run in automatic processes. These processes receive the configs and execute the readers, aggregators and send all data to the Data Lake.



GET /settings Get All Configs
POST /settings Set Param
GET /settings/{config_id} Get Config Id
PUT /settings/{config_id} Update Config Id
DELETE /settings/{config_id} Delete Config Id
PUT /settings/{config_id}/enable Enable Config Id
PUT /settings/{config_id}/disable Disable Config Id

Figure 4-7: MDA OpenAPI

4.3. SLA Monitoring

The SLA Monitoring service collects and analyses monitoring data from the Datalake and uses this data to detect SLA violations in real time. Furthermore, it keeps a record of the SLAs of all the active contracts to periodically consume resource metrics. Upon occurring SLA violations, the service provider can be notified, and countermeasures can be proactively taken to maintain the desired QoS for an offered service. This service works closely with the DLT and Smart Contracts to realise trustworthy SLA governance.

4.3.1. Main Functionalities

SLA Monitoring starts operating as soon as a contractual agreement is signed in the marketplace. The MDA module is configured to collect resource monitoring data. These data are then submitted to a Kafka streaming service, exposed by the DataLake module. SLA Monitoring service waits for new SLA events in the Datalake and then subscribes to the Datalake streaming service to receive the respective monitoring data. SLAs of all the active contracts are kept internally. SLA Monitoring analyses the monitoring data and compares the metrics (SLIs) such as availability or response time, with the thresholds in SLAs, in order to detect SLA violations. Such violation notifications are subsequently propagated to the smart contract for the purposes of re-calculating SLA status.

We can divide the SLA Monitoring functionalities and operational workflow into two phases:

- **The preparation phase** (Figure 4-8) **-** is used to define what SLAs are being monitored. This flow begins with an SLA event being sent to the Datalake (more specifically to the Kafka streaming service exposed) and it includes fetching the product from the TMF Catalogue and consequently the SLA from the SCLCM. Additionally, the component subscribes to the Datalake in order to receive the respective monitoring data. The SLA is stored in an intermediate storage so as to reduce the necessary calls to external components.

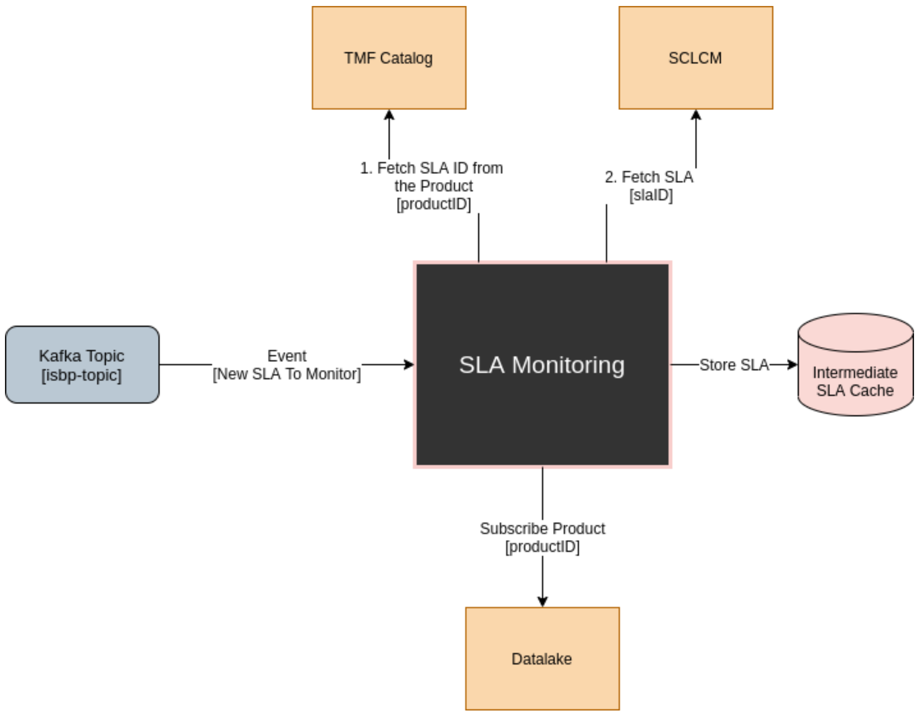


Figure 4-8: SLA Monitoring Preparation Phase

- **The monitoring phase** (Figure 4-9) - at this stage, the Data Lake streams the monitoring data to SLA Monitoring and the latter analyses this data by comparing the metrics (SLIs) with the thresholds defined in SLOs of SLAs, in order to detect SLA violations. Other components can subscribe to the SLA Monitoring output topic in the Datalake to receive notifications about the SLA Violations.

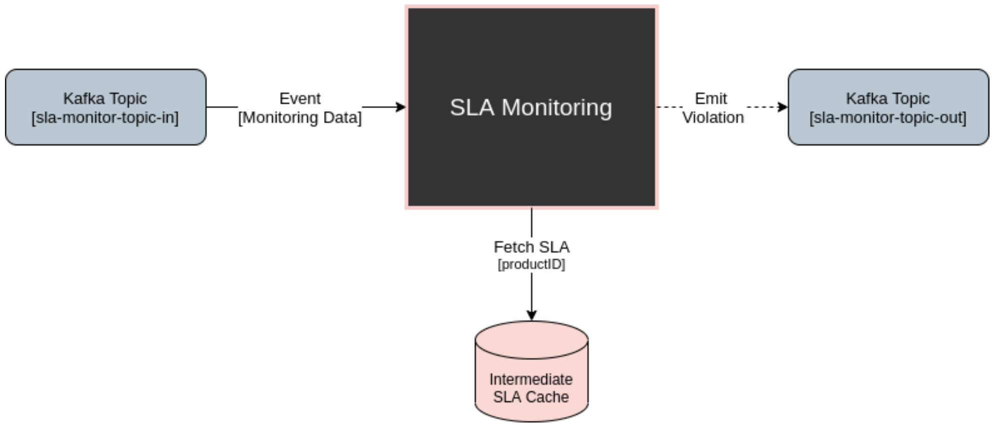


Figure 4-9: SLA Monitoring Monitorization Phase

4.3.2. Prototype Implementation

SLA Monitor is a web service written in JavaScript, using Node.js open-source server environment. It uses a publish/subscribe communication paradigm and is divided into 4 submodules:

1) consumer.js which contains two kafka consumers that receive and process SLA Events and Monitoring data events respectively;

2) producer.js that contains logic for a kafka producer;

3) redis.js that serves as an intermediate storage for the SLAs;

4) external.js that contains logic that connect to external components (fetch SLAs and subscribe to Datalake).

There are two SLA Monitoring topics: 1) sla-monitoring-topic-in used to consume Monitoring data events; and 2) sla-monitoring-topic-out to where the violation event is pushed.

The SLA Monitor module is deployed in each k8s cluster of 5GZORRO testbeds.

The source code along with the details of SLA Monitor functionalities and instructions on how to deploy can be found on 5GZORRO Github page, using the reference [39]. Deployment files related to k8s deployments are also available.

The SLA violation references the product offer and SLA in which the violation was detected, the specific rule that was violated and the actual value that was verified.

Table 4-1: SLA violation Information Model

Parameter		Type	Description
id		string	Id of the SLA violation
productDID		DID	The identifier of the product offer to which the violation relates
sla			
	id	String	Id of the SLA
	href	String	Url value of the SLA
rule			Details of the rule breached
	id	String	Unique identifier of the metric
	metric	String	Reference of metrics stored in the Service Provider “Metric Library”
	unit	String	Unit of measure of metric
	referenceValue	String	Reference value of the metric
	operator	String	Operator used when calculating the rule
	tolerance	String	The tolerance value when the threshold is crossed.
	consequence	String	Action to be applied as a result of a threshold crossing
violation			Details of the violation
	actualValue	String	The value recorded for the breach

4.4. Intelligent SLA Breach Predictor

The creation of an SLA triggers a closed-loop automation cycle whose intent is to provide quality of service insights concerning the service provisioned by the provider. The cycle begins with the MDA continuously requesting monitoring data on the resource specified in the SLA as discussed above. The cycle is complimented by the Intelligent SLA Breach Predictor (ISBP) module.

4.4.1. Main Functionalities

The ISBP module is a Data Lake component, whose main task is to analyse the monitoring data provided by the MDA. Should the analysis show that a deviation of the subject service’s performance is likely, an alert is produced that is dispatched to the relevant component for possible further actions. This analysis is conducted using a Machine Learning (ML) model which is the main actor behind the module's main and secondary feature. These features, in order, are:

- **SLA Breach Prediction:** Each of the monitoring data provided by the MDA correspond to only one SLA. Therefore, ISBP needs to be aware that a new SLA has been activated, in order to start a prediction pipeline based on this SLA's attributes. For that reason, ISBP is actively waiting for new SLA events on a message queue to consume. Upon such an event, ISBP creates an internal structure to hold the details of the SLA in order to correlate incoming monitoring data, and potentially to adjust the ML model and/or the SLA details, if the SLA itself is modified during its lifecycle. After that, ISBP periodically requests monitoring data of that SLA from the Data Lake storage service. As discussed, these monitoring data contain metrics on the resource specified in the SLA. These metrics, which are accumulated in a first-come-first-served basis by ISBP, are then used as input on the ML model which then predicts the next metric in the sequence. Lastly, the predicted value is compared against the threshold determined by the SLA. If that threshold is found to be violated, a breach notification is then constructed that contains all the relevant information needed for the component that will receive it to decide whether to take action or not. This workflow is visible in Figure 4-10.

- **ML Model training:** In parallel with the above operation, every new prediction is also compared against the next metric that is received by ISBP. This is because for every prediction, there is an actual metric captured by MDA that it corresponds to. This comparison yields an accuracy value that is added to the accuracy values of the previous steps. After enough number (e.g., 10) of these values are aggregated, their median is used to assess the ML model's performance. If this performance is found to be less than what can be accepted, ISBP triggers an operation that uses a fraction of the monitoring data to re-train the model in order to achieve better performance. Otherwise, no action is taken.

Additionally, a REST interface is also available that accept requests concerning one or more prediction pipelines. The response, in JSON format, contain the requested pipeline's details like the current status of execution, SLA details and ML model details.

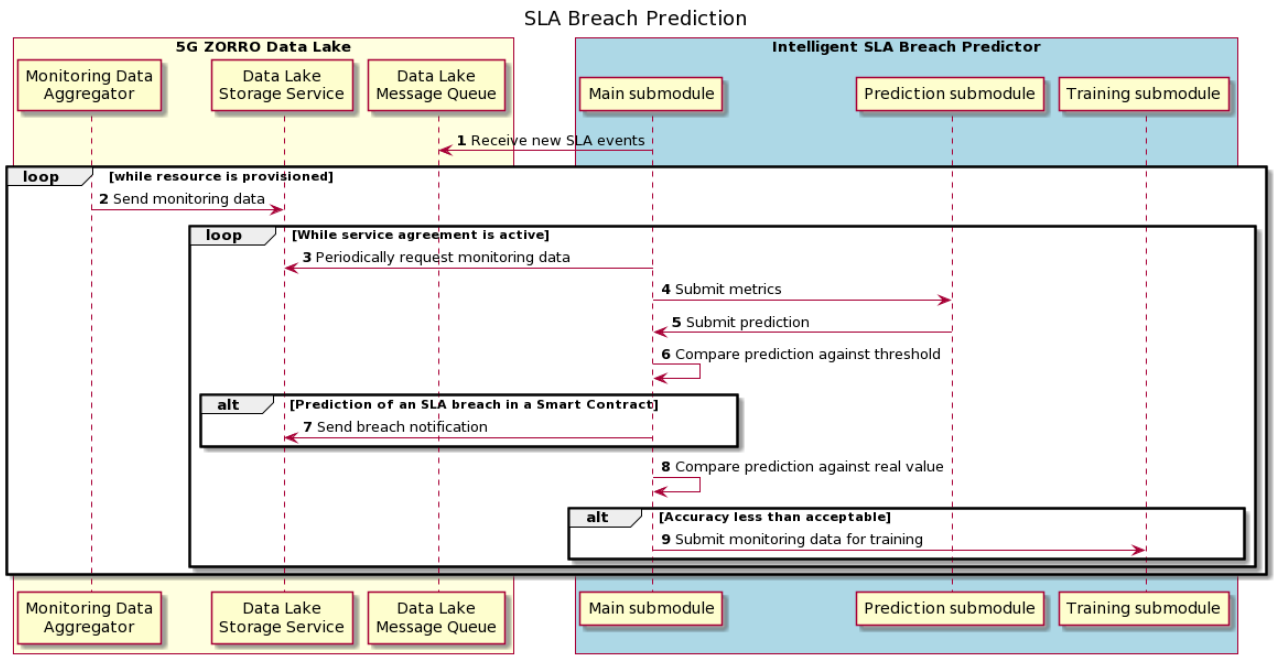

SLA Breach Prediction
5G ZORRO Data Lake
Intelligent SLA Breach Predictor
Monitoring Data Aggregator
Data Lake Storage Service
Data Lake Message Queue
Main submodule
Prediction submodule
Training submodule
1 Receive new SLA events
loop [while resource is provisioned]
2 Send monitoring data
loop [While service agreement is active]
3 Periodically request monitoring data
4 Submit metrics
5 Submit prediction
6 Compare prediction against threshold
alt [Prediction of an SLA breach in a Smart Contract]
7 Send breach notification
8 Compare prediction against real value
alt [Accuracy less than acceptable]
9 Submit monitoring data for training

Figure 4-10: SLA Breach Prediction Workflow

4.4.2. Prototype Implementation

The ISBP module is a web service written in Python, using the FastAPI library for developing REST applications. It consists of three loosely coupled submodules that interact with each other via a REST interface: 1) the main submodule that contains the REST interface, the Kafka message queue consumer and the functions for managing the SLA pipeline, 2) the submodule that conducts the prediction operation which it subsequently sends to the main submodule and 3) the submodule that executes the training operation. All submodules have access to the same ML model database from which they execute read/write operations. The above description is depicted in Figure 4-11.

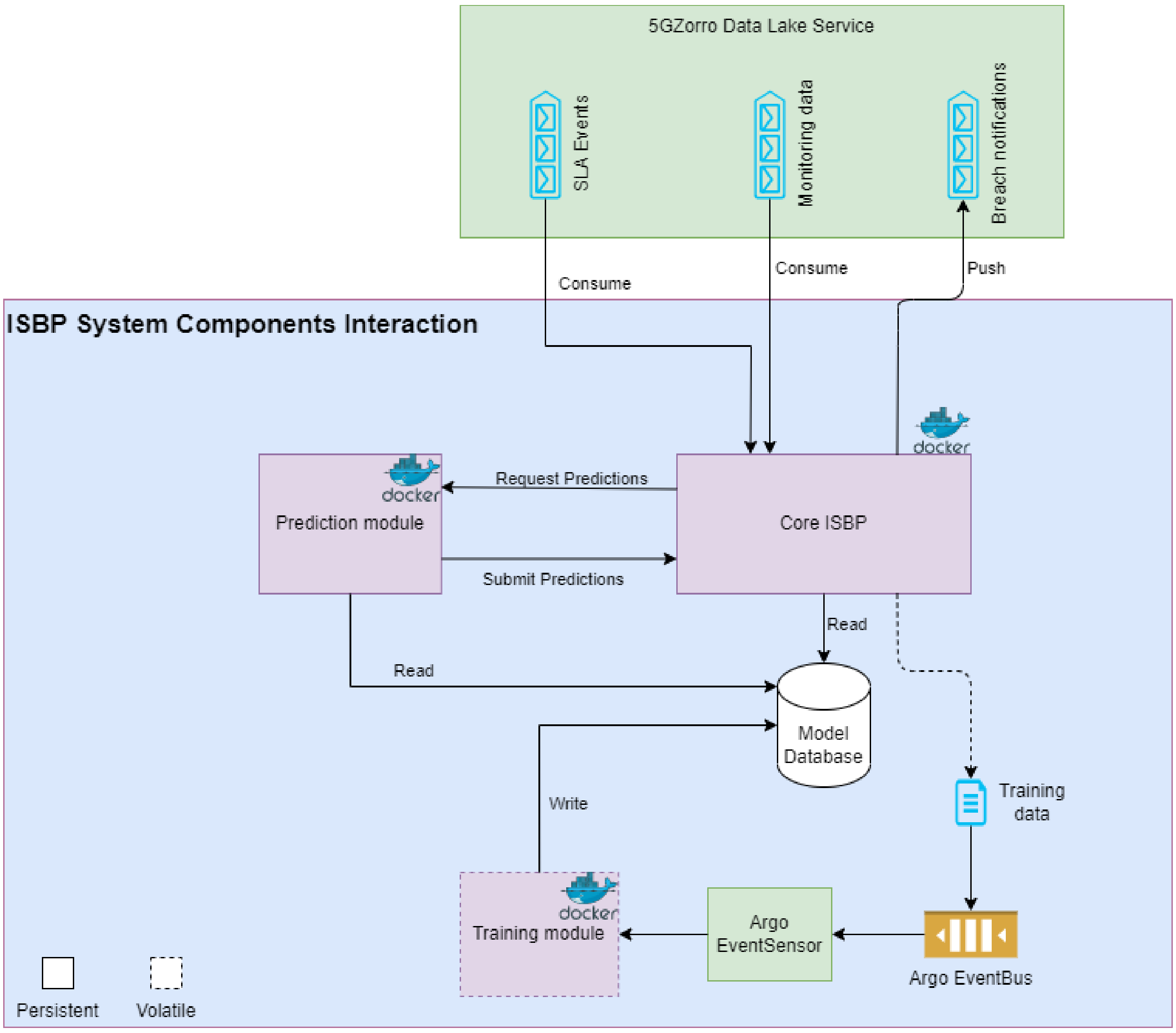
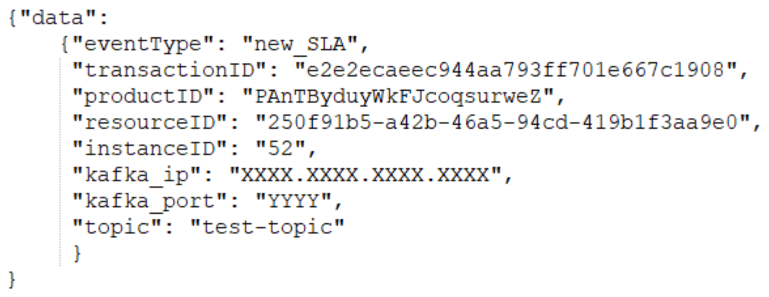


Figure 4-11: ISBP Component Architecture & Interactions

It should be noted that the training submodule is implemented as a python job that is triggered upon a certain event, enabled by the Argo Workflows framework. Additionally, the prediction submodule is also a FastAPI service with a single REST endpoint that accepts a configuration in order to generate predictions. This service has all ML libraries loaded in memory during start-up to minimize the latency between consecutive predictions. The main submodule of ISBP as well as the prediction module, are deployed on a k8s cluster as deployments, while the training module is triggered as an individual pod every time there is a need for the operation.

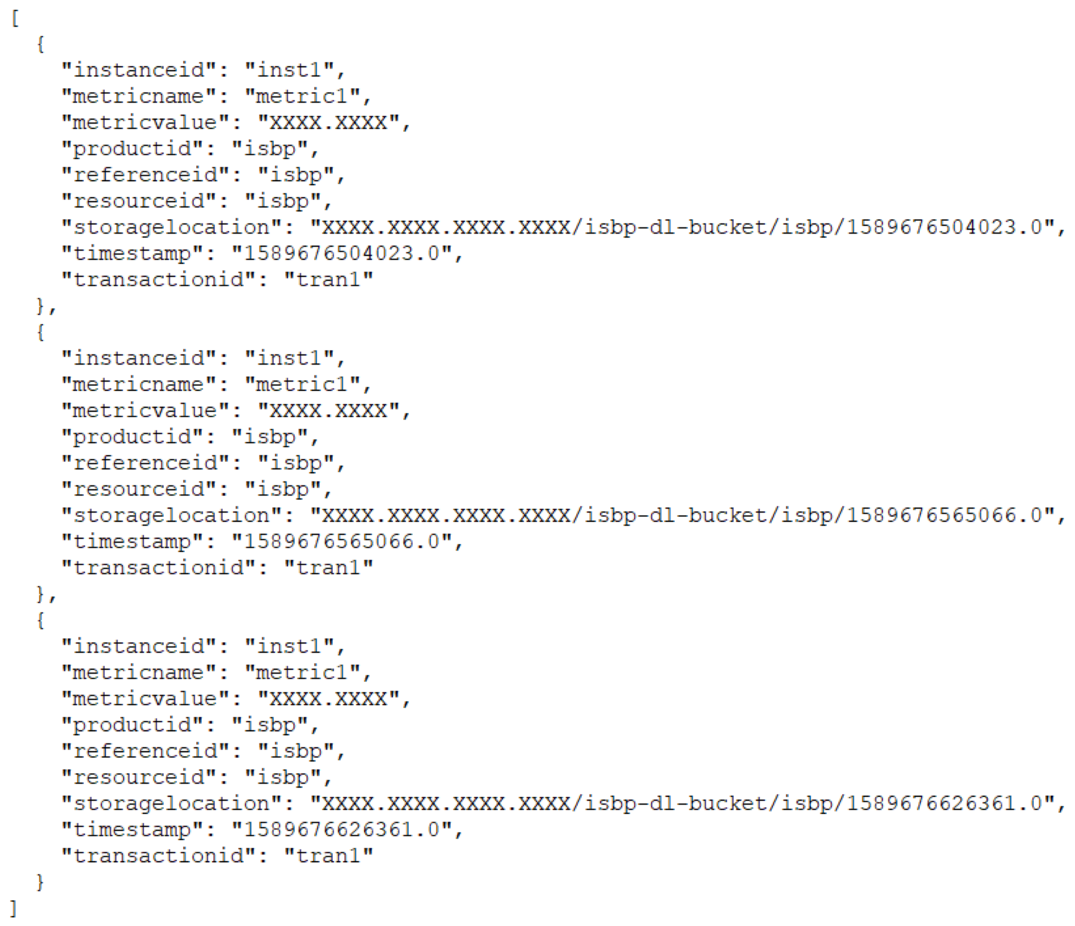
The source code along with instructions on how to deploy and use ISBP can be found at [40].

Additionally, in the following figures, the data format of the communication paradigm described in the previous section is provided. Figure 4-12 depicts the format of the SLA event that is initially received by ISBP in order to trigger a prediction pipeline. Figure 4-13 depicts the response of the Data Lake storage service containing the latest monitoring data acquired by the MDA. Finally, Figure 4-14 shows the breach notification pushed by ISBP. All messages are formatted in JSON.



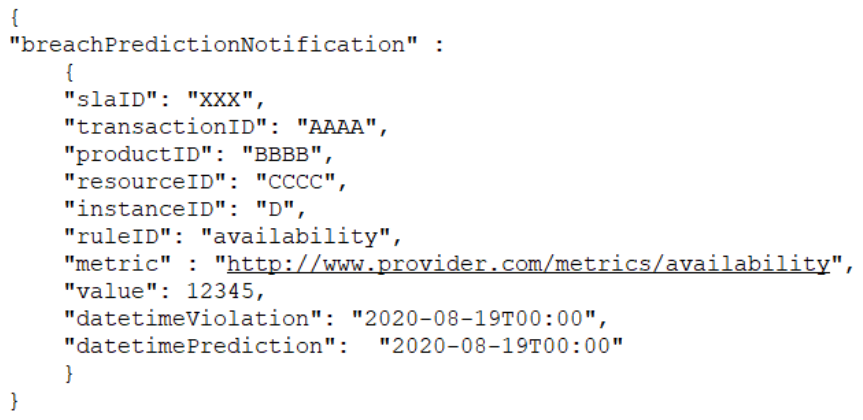
```
{"data":
    {"eventType": "new_SLA",
     "transactionID": "e2e2ecaeec944aa793ff701e667c1908",
     "productID": "PAnTByduyWkFJcoqsurweZ",
     "resourceID": "250f91b5-a42b-46a5-94cd-419b1f3aa9e0",
     "instanceID": "52",
     "kafka_ip": "XXXX.XXXX.XXXX.XXXX",
     "kafka_port": "YYYY",
     "topic": "test-topic"
     }
}
```

Figure 4-12: SLA format



```
[
  {
    "instanceid": "inst1",
    "metricname": "metric1",
    "metricvalue": "XXXX.XXXX",
    "productid": "isbp",
    "referenceid": "isbp",
    "resourceid": "isbp",
    "storagelocation": "XXXX.XXXX.XXXX.XXXX/isbp-dl-bucket/isbp/1589676504023.0",
    "timestamp": "1589676504023.0",
    "transactionid": "tran1"
  },
  {
    "instanceid": "inst1",
    "metricname": "metric1",
    "metricvalue": "XXXX.XXXX",
    "productid": "isbp",
    "referenceid": "isbp",
    "resourceid": "isbp",
    "storagelocation": "XXXX.XXXX.XXXX.XXXX/isbp-dl-bucket/isbp/1589676565066.0",
    "timestamp": "1589676565066.0",
    "transactionid": "tran1"
  },
  {
    "instanceid": "inst1",
    "metricname": "metric1",
    "metricvalue": "XXXX.XXXX",
    "productid": "isbp",
    "referenceid": "isbp",
    "resourceid": "isbp",
    "storagelocation": "XXXX.XXXX.XXXX.XXXX/isbp-dl-bucket/isbp/1589676626361.0",
    "timestamp": "1589676626361.0",
    "transactionid": "tran1"
  }
]
```

Figure 4-13: Data Lake response containing the latest monitoring data acquired by the MDA



```
{
"breachPredictionNotification" :
    {
    "slaID": "XXX",
    "transactionID": "AAAA",
    "productID": "BBBB",
    "resourceID": "CCCC",
    "instanceID": "D",
    "ruleID": "availability",
    "metric" : "http://www.provider.com/metrics/availability",
    "value": 12345,
    "datetimeViolation": "2020-08-19T00:00",
    "datetimePrediction":  "2020-08-19T00:00"
    }
}
```

Figure 4-14: Breach notification message

4.5. Smart Resource and Service Discovery Application

The 5GZORRO architecture aims to facilitate multi-party collaboration in dynamic 5G environments where Operators and Service Providers often need to employ 3rd party resources to satisfy a contract. To do so, Resource Providers make their resource offers available for sharing by advertising them through the 5GZORRO Marketplace. These resources belong to different parts of the 5G network, such as the Mobile Core/Cloud, the RAN, as well as the mobile or infrastructure edge resources.

As previously stated, stakeholders acting as assets consumers interact with the Marketplace, and in particular with the services provided by the RSOC (Section 3.2), to obtain the set of product offers available and suitable to satisfy their need. While this approach may be sufficient, it leaves to the customer, not only the decision of what 3rd party resources are the most appropriate to use, but also the fine-grained searching over provided offers in order to find the ones that best match some particular scenario or optimization criteria, which quite often may imply more complex trade-offs.

To complement this and enforce the zero-touch capabilities of the 5GZORRO platform, the Smart Resource and Service Discovery (SRSD) application allows obtaining a customized subset of resources that best satisfy the consumer expectations. Specifically, the aim of this component is to enable a programmatic and intent-based discovery by using smart selection techniques based on ML.

4.5.1. Main Functionalities

The SRSD module aims to provide the marketplace with functionalities that allow the intelligent discovery of offers available. More in particular, the module enables the automated identification and selection of multi-party resources on a marketplace composed by various types of resources and services. To fulfil this objective, the functionalities of this module are mainly provided by two main submodules:

- Intent-based framework, which allows translating high-level requests specified by the end-users in human language into particular technical specifications to be mapped into offers available in the marketplace across various stakeholders. The translation phase is an AI-driven process, powered by a Natural Language Processing (NLP) model.
- Offline clustering and online classification ML models, which aim to first (offline) identify patterns across a set of available training offers, and then (online) classify and identify the incoming offers to be registered into one of those patterns, in order to speed up the search process in the marketplace started by the intent-based framework when a new resource demand request is received.

These two submodules satisfy the specifications, functional and technical requirements defined in D3.3 [2]. The support for both AI-based model for off-line clustering and on-line prediction, together with the intent-based API, enable the automated discovery and smart categorization of resources offers per type (compute, storage, network, RAN), format (physical, virtual), slice segment (core, RAN, transport, edge), location, offered price and other set of distinct capabilities such as specific indicators related to SLA.

A sample of the internal workflow associated to the submodules composed de SRSD application is displayed in Figure 4-15, in the particular case of holding two stakeholders in the marketplace. The modules interact through REST APIs and share a set of non-structured databases for the exchange and storing of information with respect to the clusters defined and the synthetic categories assigned to the incoming offers at the marketplace, and that are used by the intent-based module to attend the users' requests. The next subsection provides a more detailed information on the definition of such databases.

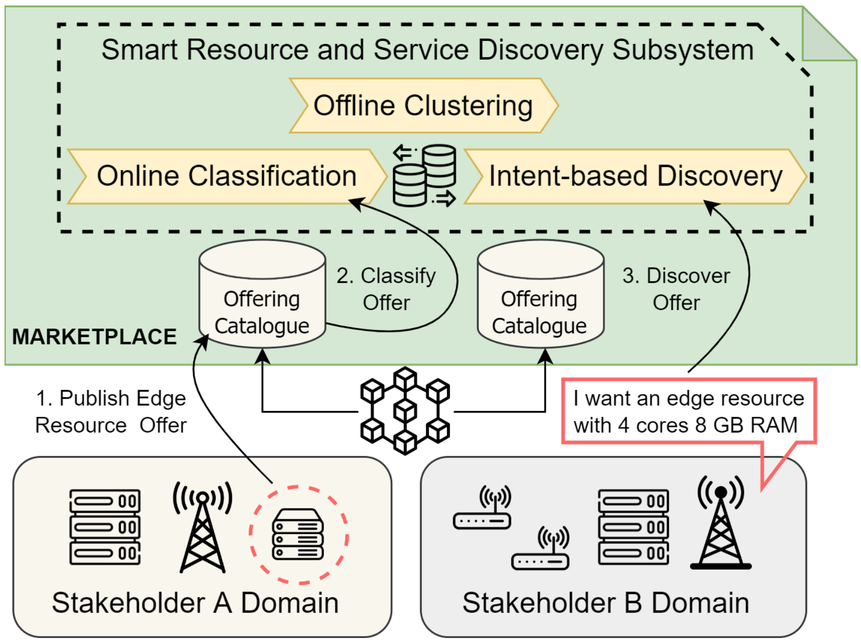


Figure 4-15: Sample workflow of the SRSD

4.5.2. Prototype Implementation

In SRSD application two subsystems are utilized. One for the realization-translation of the user intent and one for the clusterization of the product offers. Also, the mapping between the two submodules is also an integral part of the application, as the intent is matched to suitable clusters and then the corresponding offers are returned to the user. Note that for each cluster item its distance from the initial query is estimated, which in combination with trust scores or other arbitrary criteria, is used to rank returned results.

For the realization of the user intent, a Domain specific language (DSL) approach was followed. DSL frameworks are utilized for building language models, which focus on the specifics of the language used by the application, rather than generating a more generic natural language. This approach allows for a more robust, efficient and less error prune platform than traditional NLP techniques. RITA [44] (NLP) framework based on Apache UIMA RUTA was used, a language tool focused on writing manual definition or rules that are application specific. The pattern recognition subsystem used, was the spaCy [45] ML back-end system, an open-source software library for advanced NLP, written in the programming languages Python and Cython, that compiles the intents into spaCy compatible patterns.

The GitHub repository of the main SRSD application containing the intent recognition and the mapping of the user intent to the corresponding clusters is provided at [41].

The returned product offers are returned sorted by their trust score which is computed by sending the selected offers to 5G-TRMF (5G-enabled Trust and Reputation Management Framework) module, by an integral Rest http communication bus.

For the ML classification, both the clustering and the classification ML models are developed in Python, using open-source and widely available libraries. The clustering model is based on the conjunction of two main types of models, namely Factorial Analysis of Fixed Data (FAMD) and K-Means. The former aims to reduce the dimensionality of the wide range of features of each offer type, while the second takes the reduced dataset produced by the FAMD algorithm to generate the clusters. The algorithms are implemented taking as basis the Scikit-learn [46] (sklearn) library for ML pre-processing, training and validation. Conversely, the classification method is based on a set of random forest ML models built using the H2O library [42]. Each random forest model is associated to a specific offer type, as it was the case of the clustering algorithm. Different from other libraries, H2O allows defining classification models able to process at the same time both numerical and categorical libraries, which makes it an excellent choice for the purpose of this smart SRSD.

For information storing, both the classification and the intent-based ML models rely on MongoDB databases. Such databases are often shared between both modules, since the interaction across them is key for the success of the application. In particular, the management of the databases is performed as described in the operations detailed in Table 4-2.

Table 4-2: Database relationship on the SRSD application

Database name	Description	Data type	Operations
srsd-clusters-db	- Contains a set of collections, one per each offer category in the RSOC. - Each collection contains a table showing, as many rows as the number of clusters identified, and as many columns as the number of features used to create the clusters. - Each cell in the table represents the centroid of the cluster (average of all the points contained in the cluster) for a given cluster number and feature.	Tables, Structured data	- The clustering ML model creates the database and updates the information when the clusters are regenerated. - The intent-based module reads the values of the clusters when a request is received at the marketplace to match the user's translation with cluster features.
srsd-clusters-dev-db	- Contains a set of collections, one per each offer category in the RSOC. - Each collection contains a table showing, as many rows as the number of clusters identified, and as many columns as the number of features used to create the clusters. - Each cell in the table represents the average deviation of the cluster for a given cluster number and feature (computed as the average deviation between	Tables, Structured data	- The clustering ML model creates the database and updates the information when the clusters are regenerated. - The intent-based module reads the values of the clusters when a request is received at the marketplace to calculate the minimal distance between the translation made and the cluster numbers matching it.

	each point contained with respect to the centroid).		
offers-catalogue-db	- Contains a set of collections, one per each offer category in the RSOC. - Each collection contains the original offer introduced in the marketplace plus an additional field indicating the cluster number on which it has been classified within a certain offer type.	Json files, Structured data	- When receiving a new offer at the marketplace, the smart resource and service discovery application stores the received data in the corresponding collection of the database, adding the cluster assigned according to its offer category. - The intent-based module also gets access to this database to gather the offers related to a certain cluster and return them to the user when receiving a certain request.
srsd-models-db	- Contains a set of collections, one per each offer category in the RSOC. - Each collection contains the clustering and classification ML models generated for each type of offer.	Bytes, Unstructured data	- When receiving a new offer at the marketplace, this database is accessed by the smart resource and service discovery application to gather the required classification ML model according to the offer type. - The database is also accessed during maintenance of the ML models, upgrading first the clustering and then the classification model.

In terms of deployment, the application is deployed in a k8s cluster as a Docker-container endpoint.

4.6. Validation Tests

This subsection reports the tests performed to validate the different operations supported by the Datalake and Cross-domain Analytics & Intelligence for AIOps components and their interactions. For each component, we document the set of functional tests executed, and we provide some usage scenarios illustrating the joint operation of the components.

4.6.1. Functional Tests

Table 4-3 lists the set of tests performed to validate the functionalities implemented by the Operational Data Lake. The basic API allows users to register/unregister. Once registered, the user is able to use other Datalake services, such as data ingestion and lookup, and data streaming services. Shell scripts are provided to test the basic operations of each of these services. The Datalake provides the ability for a user to define their own pipelines and services. These same capabilities are used to set up some predefined pipelines and services, such as the streaming service.

Table 4-3: Operational Datalake functional test set

Name	Description	Passed (Yes/No/Partially)
Datalake Server API	Create/Delete/List registered users; Create/Delete/List pipelines; Create/Delete/List services See shell scripts in https://github.com/5GZORRO/datalake/tree/master/experiments	Yes
Datalake Ingestion Pipeline and Lookup Service	Generate monitoring data to ingest; verify it can be looked up by the various fields: productID, resourceID, transactionID, etc. See shell scripts in https://github.com/5GZORRO/datalake/tree/master/services/catalog	Yes
Datalake Streaming Service	Allow module to register and to receive monitoring data based on productID. Tested by integration with the SLA Monitoring component.	Yes

Table 4-4 lists the set of tests performed to validate the functionalities implemented by the MDA for monitoring data and storage on the Data Lake. The tests reported are the result of the integration with the NSSO component (see D4.3 [21]), which sends the required configuration to enable data collection & aggregation. For the data storage part, MDA triggered an automatic process to send the aggregated data to Data Lake, according to the topic on the configuration sent by NSSO.

Table 4-4: MDA functional test set

Name	Description	Passed (Yes/No/Partially)
Post Monitoring Data	Operator provides monitoring data, which is saved in the Data Store and which is also forwarded to the aggregator.	Yes
Aggregate Monitoring Data	Process accumulated monitoring data and aggregate into a form that is consumable by other components.	Yes
Start Aggregate Metric	Specify a particular metric that we want to aggregate.	Yes
Stop Aggregate Metric	Specify a particular metric that we want to stop to aggregate.	Yes
Lookup Metrics By Reference	Obtain list of aggregated metrics that pertain to specified *productID*.	Yes

Table 4-5 lists the set of tests performed to validate the functionalities implemented by the SLA Monitoring module. The tests performed validate the integration of the SLA-monitoring component with the necessary external components, namely the ISBP to start monitoring new SLAs, the RSOC to retrieve the SLA information based on the product, the SCLCM to retrieve the SLA details and last but not least the Datalake to which we subscribe and publish information to.

Table 4-5: SLA Monitoring functional test set

Name	Description	Passed (Yes/No/Partially)
Trigger SLA Monitoring service	Receive an SLA event and trigger the subscription of the Product to the Datalake in order to receive the respective monitoring data.	Yes
SLA Monitoring	Execute the SLA Monitoring process by comparing the metrics (SLIs) with the thresholds defined in SLOs of SLAs, in order to detect SLA violations.	Yes
Publish SLA Violation Notification	Notify relevant components when a threshold specified in the SLA is exceeded and a violation is detected.	Yes

Table 4-6 lists the set of tests performed to validate the functionalities implemented by the ISBP. The tests were conducted before, during and after integration sessions that included modules such as Data Lake, ISSM and MDA. The interaction between the modules and ISBP was done mainly through Kafka messages, while some of it also took place via HTTP. The success of the test was determined, by comparing the content of the logs of ISBP against the expected outcome.

Table 4-6: ISBP functional test set

Name	Description	Passed (Yes/No/Partially)
Trigger prediction service	Create a prediction service with every new SLA event. Every metric received is correlated with a service based on its source.	Yes
Generate predictions	Generate time series predictions using pre-trained ML models, taking the incoming metrics as input.	Yes
Notify in case of SLA breach prediction	Notify relevant components when a prediction exceeds the threshold specified in the SLA.	Yes

Table 4-7 lists the set of tests performed to validate the functionalities implemented by the SRSD. These tests validate the services exposed by the SRSD being consumed by other 5GZORRO platform modules (such as the Marketplace Portal and RSOC). In terms of internal interactions, the intent-based discovery submodule queries the SRSD's MongoDB in order to retrieve clustering information and index offers of interest. SRSD also requests trust information about the offers of interest from the 5G-TRMF, which affects the ranking of the offers that SRSD returns.

Table 4-7: SRSD application functional test set

Name	Description	Passed (Yes/No/Partially)
Predict Offer	Classify an offer based on pre-computed clusters, resulting in a new entry added to the application database with classified offers.	Yes

Remove Classified Offer	Remove a classified offer from the application database (because of the offer being no longer available on the Marketplace).	Yes
Clusters Generation	Generate new clusters offline and update the classifier model for online prediction.	Yes
Discover Offers	Discover available offers based on high-level intents that are translated into the corresponding clusters.	Yes
Trust Service	Request the assessment of multiple Product Offers to the 5G-TRMF so as to compute their trust scores and return and sorted list.	Yes

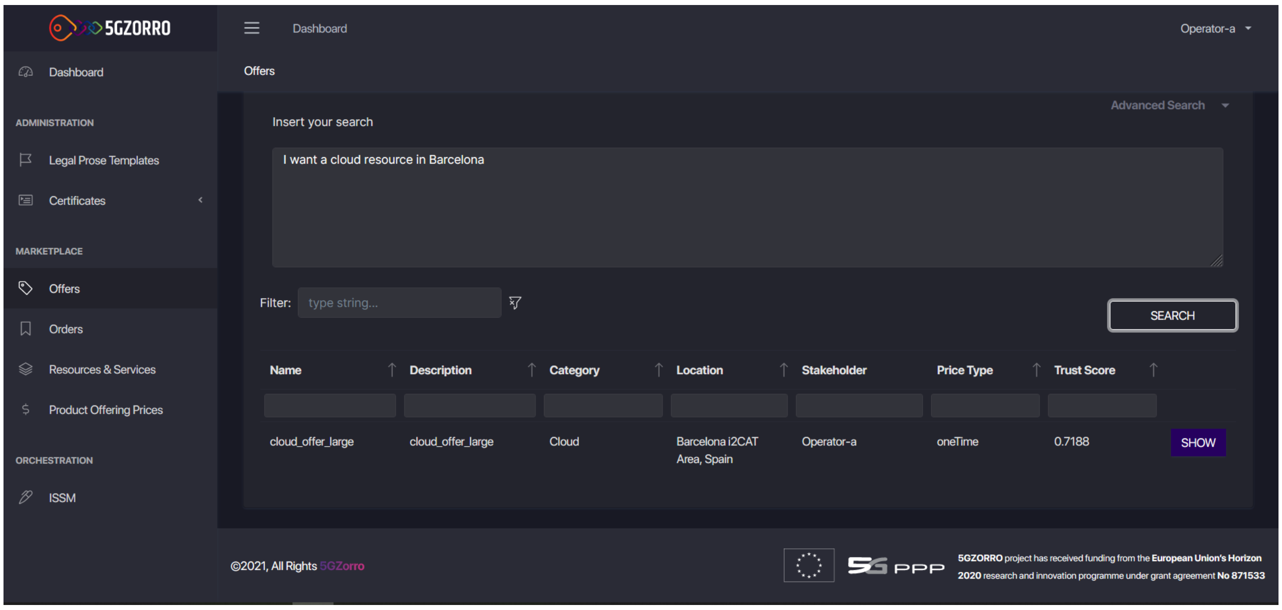
4.6.2.Platform Operations

Next, some usage scenarios of the AIOps prototypes and Data Lake are described to validate the system capabilities related to the intent-based discovery, the aggregation of monitoring data, the SLA monitoring and the intelligent SLA breach prediction.

4.6.2.1. Intent-based discovery of offers

The capability for intent-based discovery supported by the SRSD is directly provided to the Portal users via the integration of this module with the Marketplace Portal (described in Section 3.4). In order to do so, in the Search Offers view, an Advanced Search textbox is available for the user to type an intent and request this intelligent searching service. The request will be made to SRSD's endpoint, which will process the intent, by initially recognizing the user's intention by using the NLP model it utilizes internally. Then it will use the clustering information about the offer category identified in the intent and match the user's intent with the available offers. After processing the available offers, SRSD will reply back to the Portal with a JSON object containing the selected orders, which are displayed in the Portal view.

This module supports the 7 offer categories considered in 5GZORRO, i.e., 5 resource types (cloud, edge, RAN, spectrum, and VNF) and the 2 service types (network service and slice). The following figures from Figure 4-16 to Figure 4-22 illustrate the execution of this operation for each offer category, including also the location in the intent. In the portal, the user clicks on Offers and then Advanced Search. The user types the intent in the textbox and hits the search button. The reply of SRSD will be a list of ranked offers, which will be displayed in the portal, along with relevant information per offer. If no offer is found, then no data will be shown.



Insert your search

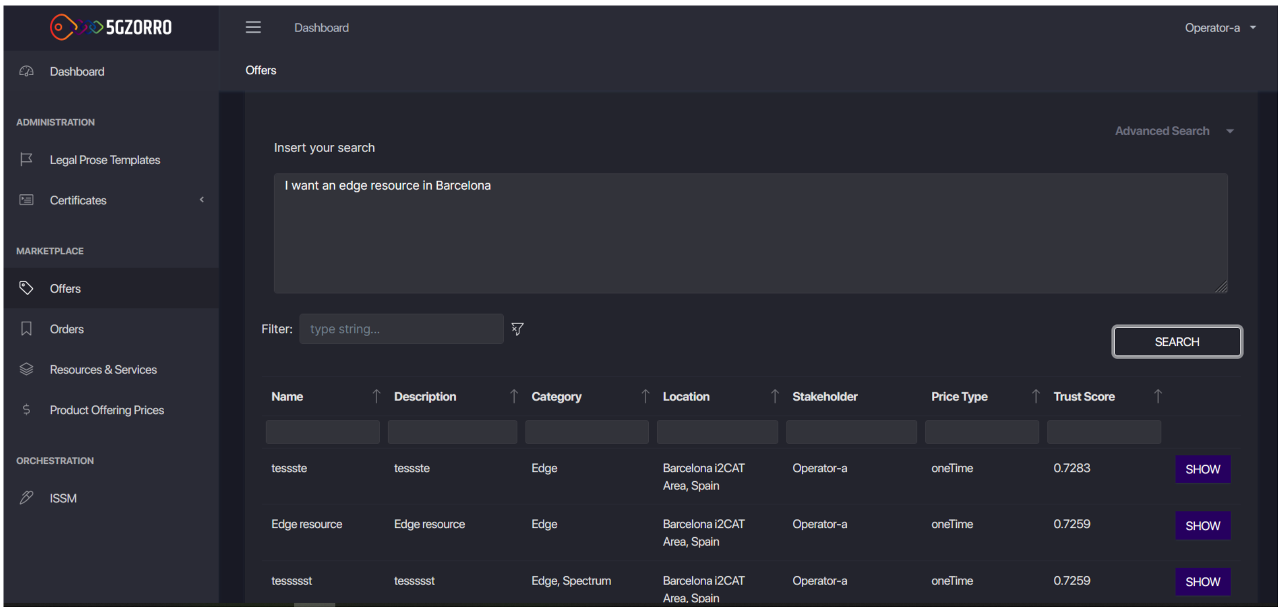
I want a cloud resource in Barcelona

Name	Description	Category	Location	Stakeholder	Price Type	Trust Score
cloud_offer_large	cloud_offer_large	Cloud	Barcelona i2CAT Area, Spain	Operator-a	oneTime	0.7188

©2021, All Rights 5GZorro

5GZORRO project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871533

Figure 4-16: Intent request for a Cloud Offer send to SRSD's endpoint through Operator-A Portal.

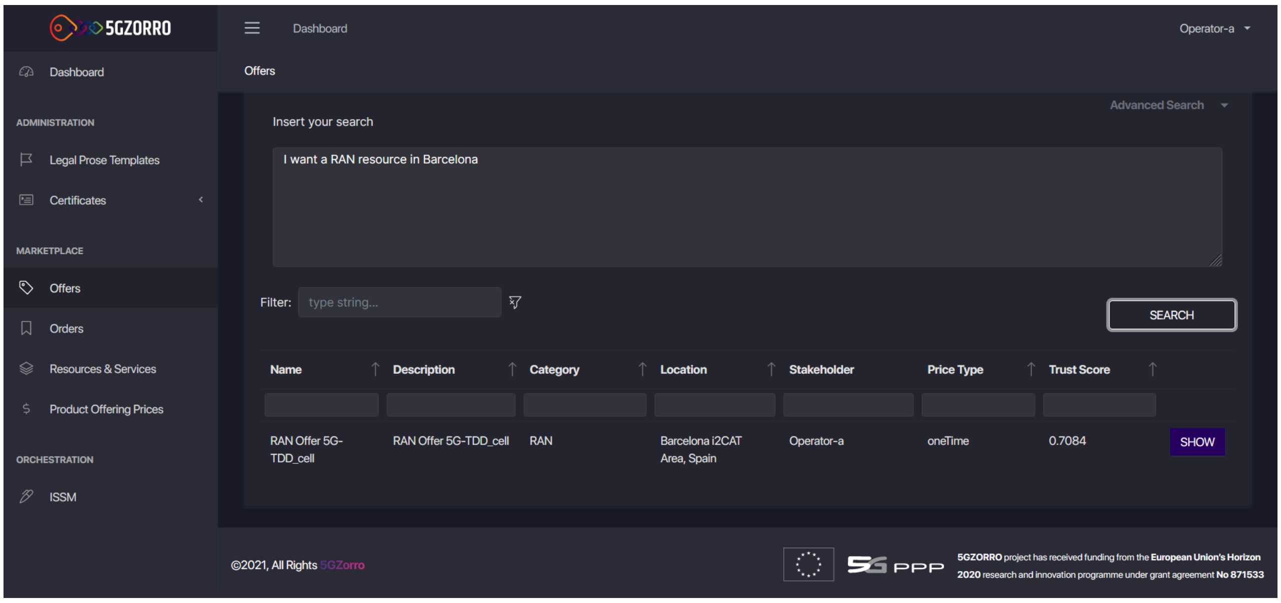


Insert your search

I want an edge resource in Barcelona

Name	Description	Category	Location	Stakeholder	Price Type	Trust Score
tessste	tessste	Edge	Barcelona i2CAT Area, Spain	Operator-a	oneTime	0.7283
Edge resource	Edge resource	Edge	Barcelona i2CAT Area, Spain	Operator-a	oneTime	0.7259
tesssst	tesssst	Edge, Spectrum	Barcelona i2CAT Area, Spain	Operator-a	oneTime	0.7259

Figure 4-17: Intent request for an Edge Offer send to SRSD's endpoint through Operator-A Portal.



Insert your search

I want a RAN resource in Barcelona

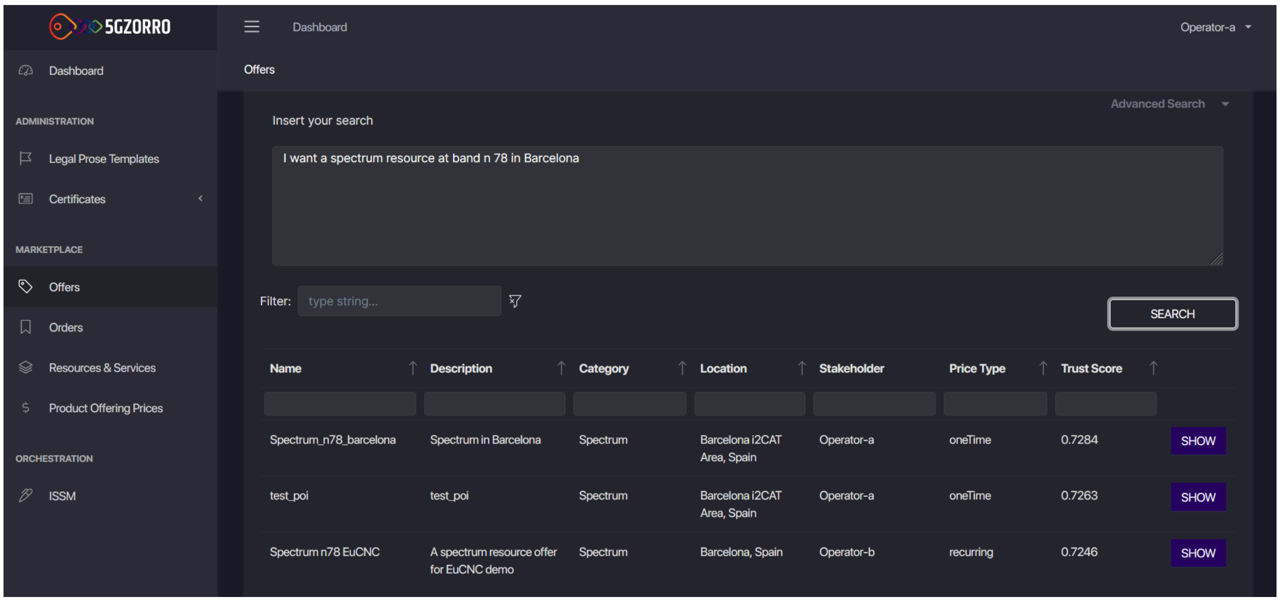
Filter: type string...

Name	Description	Category	Location	Stakeholder	Price Type	Trust Score	
RAN Offer 5G-TDD_cell	RAN Offer 5G-TDD_cell	RAN	Barcelona i2CAT Area, Spain	Operator-a	oneTime	0.7084	SHOW

©2021, All Rights 5GZorro

5GZORRO project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871533

Figure 4-18: Intent request for a RAN Offer send to SRSD's endpoint through Operator-A Portal.



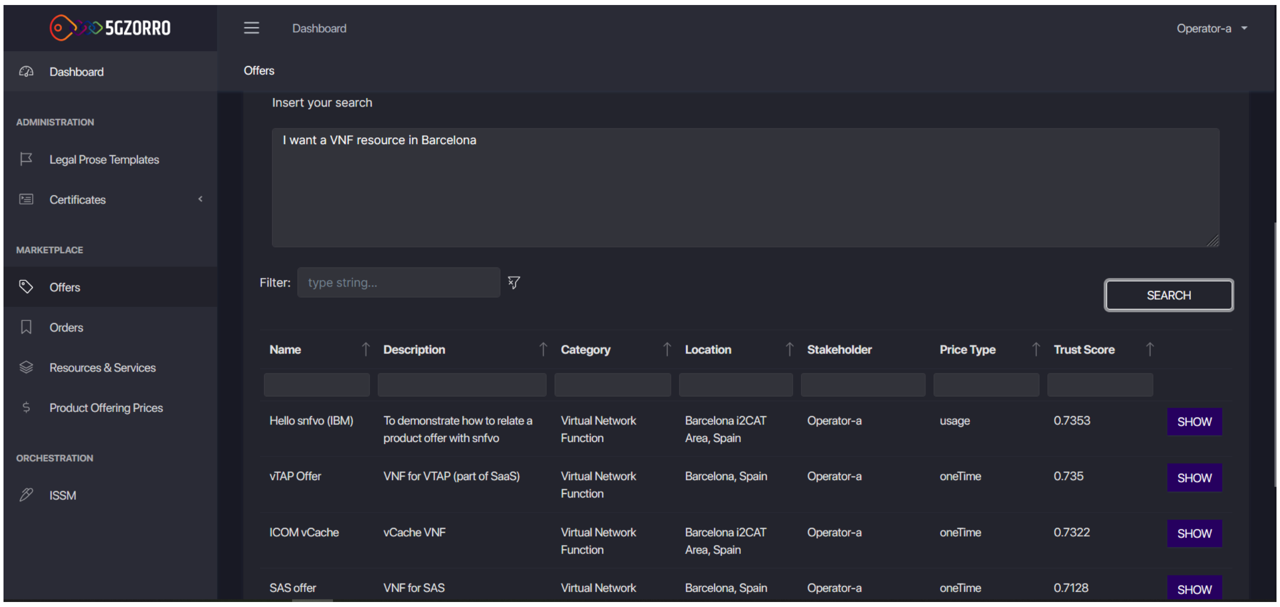
Insert your search

I want a spectrum resource at band n 78 in Barcelona

Filter: type string...

Name	Description	Category	Location	Stakeholder	Price Type	Trust Score	
Spectrum_n78_barcelona	Spectrum in Barcelona	Spectrum	Barcelona i2CAT Area, Spain	Operator-a	oneTime	0.7284	SHOW
test_poi	test_poi	Spectrum	Barcelona i2CAT Area, Spain	Operator-a	oneTime	0.7263	SHOW
Spectrum n78 EuCNC	A spectrum resource offer for EuCNC demo	Spectrum	Barcelona, Spain	Operator-b	recurring	0.7246	SHOW

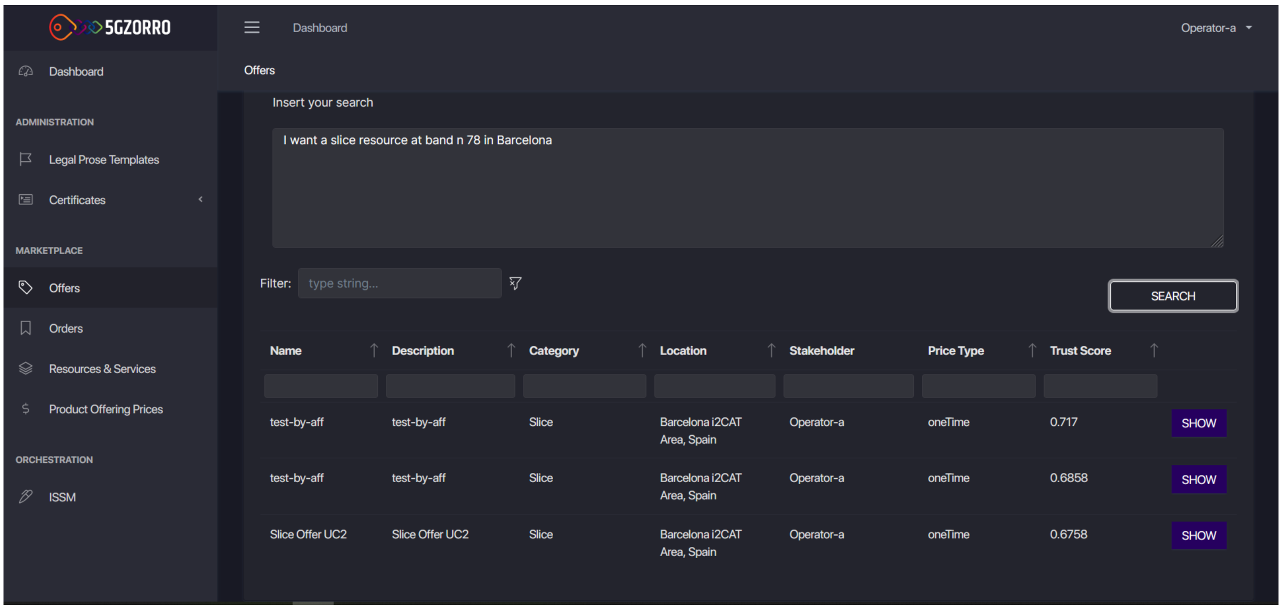
Figure 4-19: Intent request for a Spectrum Offer send to SRSD's endpoint through Operator-A Portal.



I want a VNF resource in Barcelona

Name	Description	Category	Location	Stakeholder	Price Type	Trust Score	
Hello snfvo (IBM)	To demonstrate how to relate a product offer with snfvo	Virtual Network Function	Barcelona i2CAT Area, Spain	Operator-a	usage	0.7353	SHOW
vTAP Offer	VNF for VTAP (part of SaaS)	Virtual Network Function	Barcelona, Spain	Operator-a	oneTime	0.735	SHOW
ICOM vCache	vCache VNF	Virtual Network Function	Barcelona i2CAT Area, Spain	Operator-a	oneTime	0.7322	SHOW
SAS offer	VNF for SAS	Virtual Network	Barcelona, Spain	Operator-a	oneTime	0.7128	SHOW

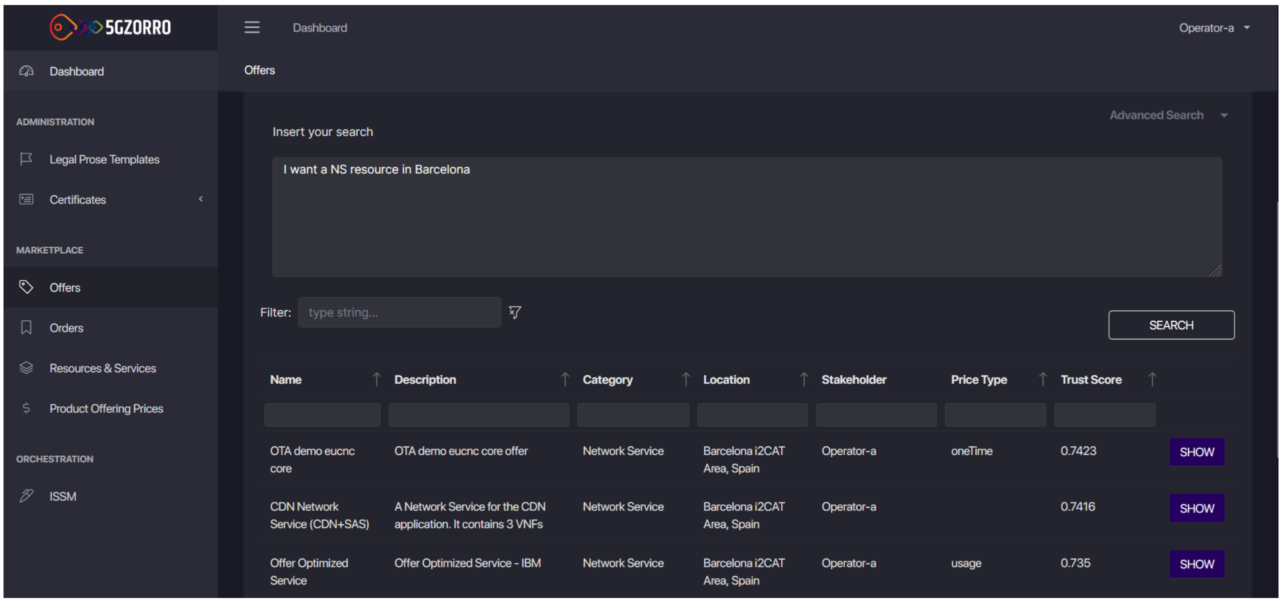
Figure 4-20: Intent request for a VNF Offer send to SRSD's endpoint through Operator-A Portal.



I want a slice resource at band n 78 in Barcelona

Name	Description	Category	Location	Stakeholder	Price Type	Trust Score	
test-by-aff	test-by-aff	Slice	Barcelona i2CAT Area, Spain	Operator-a	oneTime	0.717	SHOW
test-by-aff	test-by-aff	Slice	Barcelona i2CAT Area, Spain	Operator-a	oneTime	0.6858	SHOW
Slice Offer UC2	Slice Offer UC2	Slice	Barcelona i2CAT Area, Spain	Operator-a	oneTime	0.6758	SHOW

Figure 4-21: Intent request for a Slice Offer send to SRSD's endpoint through Operator-A Portal.



Insert your search

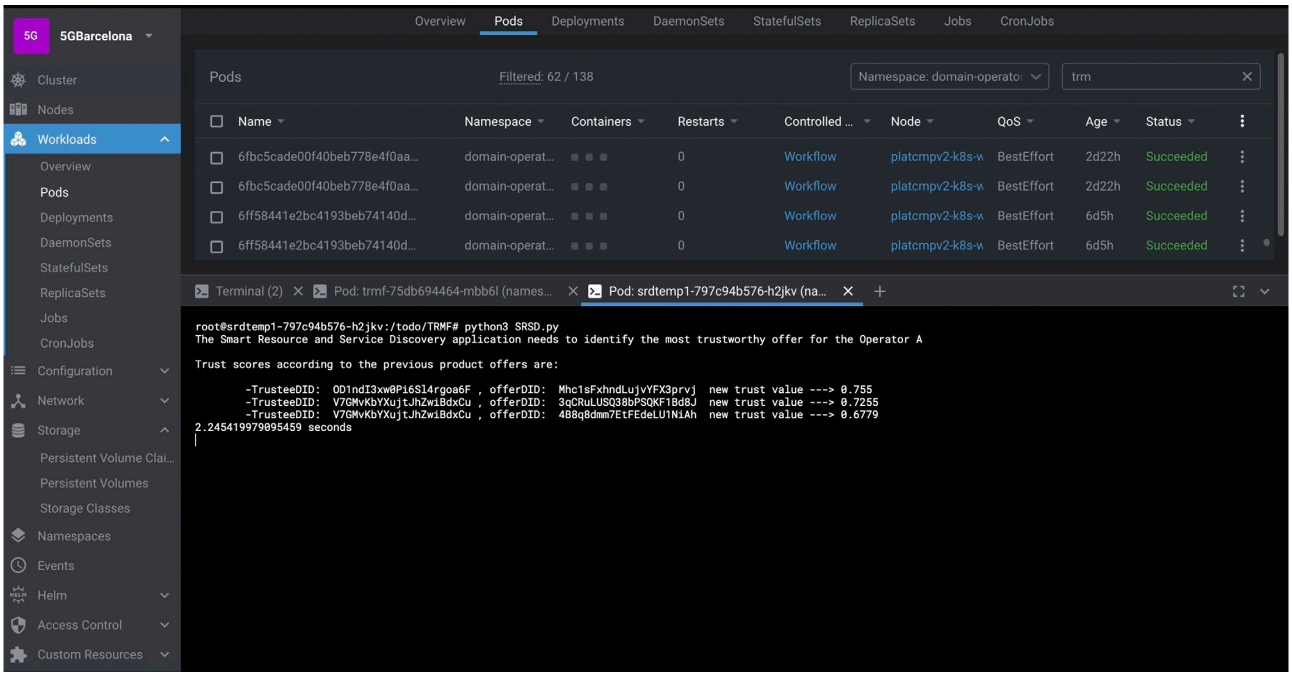
I want a NS resource in Barcelona

Filter: type string...

Name	Description	Category	Location	Stakeholder	Price Type	Trust Score	
OTA demo eucnc core	OTA demo eucnc core offer	Network Service	Barcelona i2CAT Area, Spain	Operator-a	oneTime	0.7423	SHOW
CDN Network Service (CDN+SAS)	A Network Service for the CDN application. It contains 3 VNFs	Network Service	Barcelona i2CAT Area, Spain	Operator-a		0.7416	SHOW
Offer Optimized Service	Offer Optimized Service - IBM	Network Service	Barcelona i2CAT Area, Spain	Operator-a	usage	0.735	SHOW

Figure 4-22: Intent request for a Network Service Offer send to SRSD's endpoint through Operator-A Portal.

As a sub-step of the intent-based discovery process, the SRSD module forwards a set of product offers to the 5G-TRMF so as to classify them based on reputation scores. In this vein, the 5G-TRMF carries out a thorough analysis of POs and their providers to figure out how trustworthy each available candidate is. As can be seen in Figure 4-23, the 5G-TRMF sends all offers back to the SRSD ranking from highest to lowest level of trust and reputation. Note that, D4.4 explains in detail the whole lifecycle of the 5G-TRMF as well as internal steps to compute the final trust scores.



	Name	Namespace	Containers	Restarts	Controlled ...	Node	QoS	Age	Status
☐	6fbc5cade00f40beb778e4f0aa...	domain-operat...	■ ■ ■	0	Workflow	platcmpv2-k8s-w	BestEffort	2d22h	Succeeded
☐	6fbc5cade00f40beb778e4f0aa...	domain-operat...	■ ■ ■	0	Workflow	platcmpv2-k8s-w	BestEffort	2d22h	Succeeded
☐	6ff58441e2bc4193beb74140d...	domain-operat...	■ ■ ■	0	Workflow	platcmpv2-k8s-w	BestEffort	6d5h	Succeeded
☐	6ff58441e2bc4193beb74140d...	domain-operat...	■ ■ ■	0	Workflow	platcmpv2-k8s-w	BestEffort	6d5h	Succeeded

```
root@srdtemp1-797c94b576-h2jkv:/todo/TRMF# python3 SRSD.py
The Smart Resource and Service Discovery application needs to identify the most trustworthy offer for the Operator A

Trust scores according to the previous product offers are:

        -TrusteeDID:  OD1ndI3xw0Pi6Sl4rgoa6F , offerDID:  Mhc1sFxhndLujvYFX3prvj  new trust value ---> 0.755
        -TrusteeDID:  V7GMvKbYXujtJhZwiBdxCu , offerDID:  3qCRuLUSQ38bPSQKF1Bd8J  new trust value ---> 0.7255
        -TrusteeDID:  V7GMvKbYXujtJhZwiBdxCu , offerDID:  4B8q8dmm7EtFEdeLU1NiAh  new trust value ---> 0.6779
2.245419979095459 seconds
|
```

Figure 4-23: Trust-based product offer ranking

4.6.2.2. Pipelines creation for a transaction

The Datalake DSS was tested by integrating it with the ISBP component as illustrated in Figure 4-4. The following sequence of operations occurs.

1. The ISBP component registers with the Datalake Server by invoking the *createService* API. For this, it provides the stakeholder ID and credentials, as well as a reference to the container that implements the ISBP functionality.
2. The Datalake Server processes the request and, given the request is valid, instantiates the service and returns the new service ID as well as the URLs for its basic communication services provided by the Datalake, including the input and output topics URLs as well as a port for accessing the Datalake DSS.
3. The ISBP component registers with the DSS to receive the streamed metrics data related to a specified product ID over a specified Kafka.
4. From now on, everything is set up and the following processing steps are added to the data processing pipeline for the aggregated metrics data that arrives on the Datalake Ingestion service.
    - The specified metrics data is forwarded to the DSS.
    - The DSS filters the data based on a configured productID and forwards the appropriate data to the Kafka topic of the ISBP.

4.6.2.3. Aggregation of monitoring data and storage on the Data Lake

The aggregation and flow of storing monitoring data in the Data Lake starts with the NSSO sending a configuration to the MDA, as shown in Figure 4-24. To aggregate metrics, this config needs some required fields for each metric, such as *step*, *step_aggregation* and *aggregation_method.*

```
POST 'http://172.28.3.15:30040/settings'

{
    "transaction_id": "ab51f3e1-7b61-4f9d-85a4-9e9f366b593b",
    "instance_id": "05cce067-4818-4afc-b0f8-0e4a1babf753",
    "product_id": "10",
    "tenant_id": "domain-operator-a",
    "context_ids": [
        {
            "resource_id": "10",
            "network_slice_id": "05cce067-4818-4afc-b0f8-0e4a1babf753",
            "parent_id": "10"
        }
    ],
    "topic": "operator-a-in-0",
    "monitoring_endpoint": "http://172.28.3.15:30036/osm/api/v1/query",
    "data_source_type": "OSM",
    "metrics": [
        {
            "metric_name": "osm_requests",
            "metric_type": "numerical",
            "step": "1m",
            "aggregation_method": "AVG",
            "step_aggregation": "2m"
        }
    ]
}
```

Figure 4-24: Creation of monitoring configuration

Whenever the MDA receives a new configuration, an automatic process runs to read metric values from OSM, Figure 4-25, or other source, and this process relies on Step, which is a metric reading interval, saved into a database temporarily.

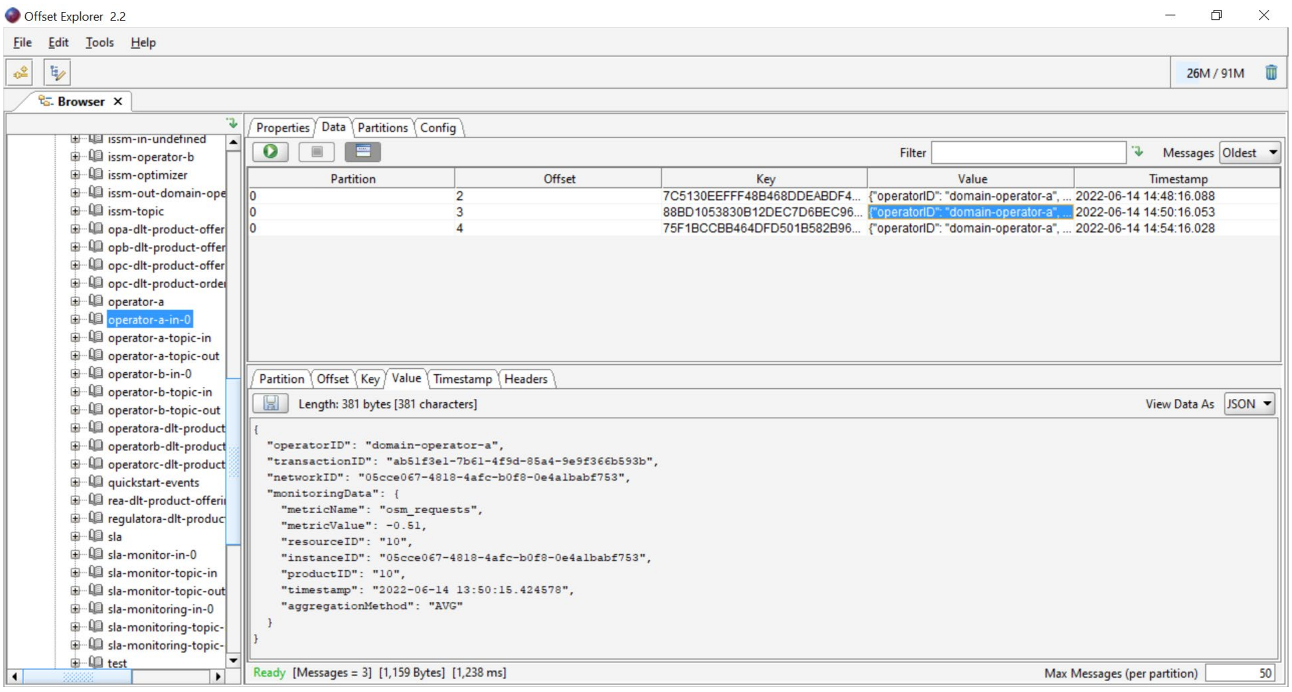
```
GET 'http://172.28.3.15:30036/osm/api/v1/query?query=osm_requests&time=2022-06-14T15:22:33.564731Z'

{
    "status": "success",
    "data": {
        "resultType": "matrix",
        "result": [
            {
                "metric": {
                    "__name__": "osm_requests",
                    "job": "mon_exporter",
                    "instance": "mon:8000",
                    "ns_id": "05cce067-4818-4afc-b0f8-0e4a1babf753",
                    "ns_name": "test0_instance",
                    "project_id": "0b0d7475-319f-48ce-a216-92b85f7800a8",
                    "vdu_name": "test-0-mgmtVM-0",
                    "vnf_member_index": "0"
                },
                "value": [
                    1636046553.564731,
                    "0.43"
                ]
            }
        ]
    }
}
```

Figure 4-25: Requesting of configured metric and its value

The next automatic process triggered is the application of aggregations, by using the *step_aggregation* key. Here, it will be determined the interval to read data from the corresponding metric. The read values are then aggregated according to the *aggregation_method* key. This aggregation method must include a valid mathematical operation, and to do so, one of these methods needs to be specified. These can be: *sum*, *average*, *count*, *max*, *min* and *stddev* (standard deviation).

The aggregation process then ends with sending the aggregated data to the Data Lake, Figure 4-26, according to the Kafka topic also received in the configuration.


Offset Explorer 2.2
File Edit Tools Help
26M / 91M
Browser
Properties Data Partitions Config
Filter
Messages Oldest
Partition Offset Key Value Timestamp Headers
View Data As JSON
Max Messages (per partition) 50

issm-in-undefined
issm-operator-b
issm-optimizer
issm-out-domain-ope
issm-topic
opa-dlt-product-offer
opb-dlt-product-offer
opc-dlt-product-offer
opc-dlt-product-order
operator-a
operator-a-in-0
operator-a-topic-in
operator-a-topic-out
operator-b-in-0
operator-b-topic-in
operator-b-topic-out
operatora-dlt-product
operatorb-dlt-product
operatorc-dlt-product
quickstart-events
rea-dlt-product-offeri
regulatora-dlt-produc
sla
sla-monitor-in-0
sla-monitor-topic-in
sla-monitor-topic-out
sla-monitoring-in-0
sla-monitoring-topic-
sla-monitoring-topic-
test

Partition	Offset	Key	Value	Timestamp
0	2	7C5130EEFFF48B468DDEABDF4...	{"operatorID": "domain-operator-a", ...	2022-06-14 14:48:16.088
0	3	88BD1053830B12DEC7D6BEC96...	{"operatorID": "domain-operator-a", ...	2022-06-14 14:50:16.053
0	4	75F1BCCBB464DFD501B582B96...	{"operatorID": "domain-operator-a", ...	2022-06-14 14:54:16.028

Length: 381 bytes [381 characters]

```
{
  "operatorID": "domain-operator-a",
  "transactionID": "ab51f3e1-7b61-4f9d-85a4-9e9f366b593b",
  "networkID": "05cce067-4818-4afc-b0f8-0e4a1babf753",
  "monitoringData": {
    "metricName": "osm_requests",
    "metricValue": -0.51,
    "resourceID": "10",
    "instanceID": "05cce067-4818-4afc-b0f8-0e4a1babf753",
    "productID": "10",
    "timestamp": "2022-06-14 13:50:15.424578",
    "aggregationMethod": "AVG"
  }
}
```

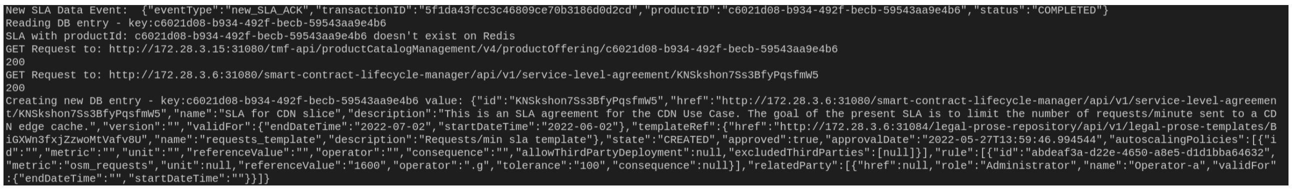
Ready [Messages = 3] [1,159 Bytes] [1,238 ms]

Figure 4-26: Storage of data on the Data Lake

With this data now present in the Data Lake, components such as the SLA Monitoring and ISBP can now determine if any irregularity from the received information on, i.e., SLAs, has occurred.

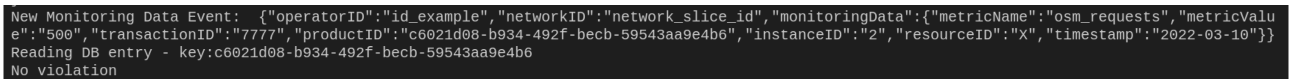
4.6.2.4. *SLA monitoring*

In Figure 4-27 we can see an SLA event being received by SLA Monitoring and the trigger of the preparation phase described previously in Section 4.3.1. Figure 4-28 and Figure 4-29 show an example of two messages containing monitoring data received by SLA Monitoring from the Datalake and acquired by the MDA. The first one shows the inexistence of a violation and the second one shows the violation/breach notification produced. All messages are formatted in JSON.



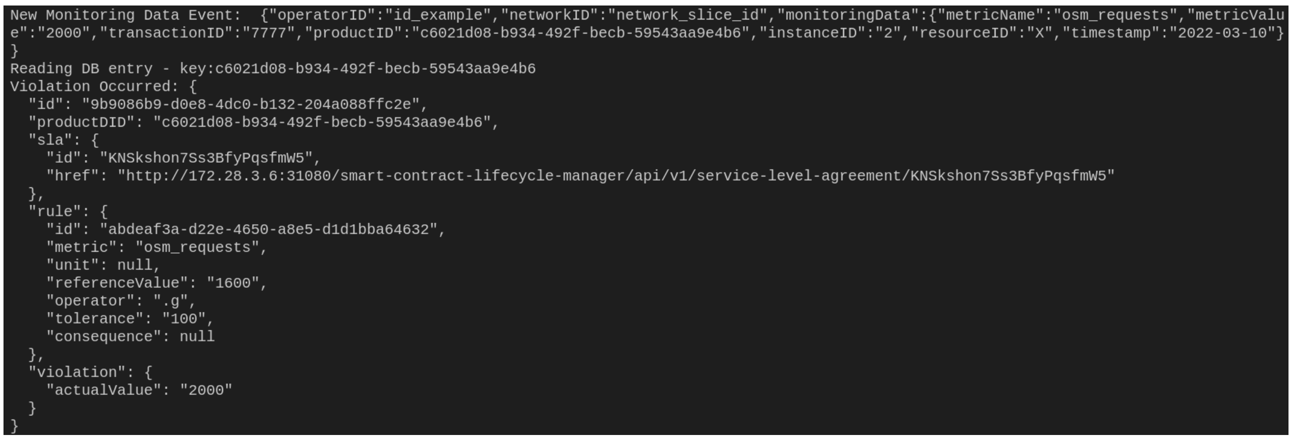
```
New SLA Data Event:  {"eventType":"new_SLA_ACK","transactionID":"5f1da43fcc3c46809ce70b3186d0d2cd","productID":"c6021d08-b934-492f-becb-59543aa9e4b6","status":"COMPLETED"}
Reading DB entry - key:c6021d08-b934-492f-becb-59543aa9e4b6
SLA with productId: c6021d08-b934-492f-becb-59543aa9e4b6 doesn't exist on Redis
GET Request to: http://172.28.3.15:31080/tmf-api/productCatalogManagement/v4/productOffering/c6021d08-b934-492f-becb-59543aa9e4b6
200
GET Request to: http://172.28.3.6:31080/smart-contract-lifecycle-manager/api/v1/service-level-agreement/KNSkshon7Ss3BfyPqsfmW5
200
Creating new DB entry - key:c6021d08-b934-492f-becb-59543aa9e4b6 value: {"id":"KNSkshon7Ss3BfyPqsfmW5","href":"http://172.28.3.6:31080/smart-contract-lifecycle-manager/api/v1/service-level-agreemen
t/KNSkshon7Ss3BfyPqsfmW5","name":"SLA for CDN slice","description":"This is an SLA agreement for the CDN Use Case. The goal of the present SLA is to limit the number of requests/minute sent to a CD
N edge cache.","version":"","validFor":{"endDateTime":"2022-07-02","startDateTime":"2022-06-02"},"templateRef":{"href":"http://172.28.3.6:31084/legal-prose-repository/api/v1/legal-prose-templates/B
iGXWn3fxjZzwoMtVafv8U","name":"requests_template","description":"Requests/min sla template"},"state":"CREATED","approved":true,"approvalDate":"2022-05-27T13:59:46.994544","autoscalingPolicies":[{"i
d":"","metric":"","unit":"","referenceValue":"","operator":"","consequence":"","allowThirdPartyDeployment":null,"excludedThirdParties":[null]}],"rule":[{"id":"abdeaf3a-d22e-4650-a8e5-d1d1bba64632",
"metric":"osm_requests","unit":null,"referenceValue":"1600","operator":".g","tolerance":"100","consequence":null}],"relatedParty":[{"href":null,"role":"Administrator","name":"Operator-a","validFor"
:{"endDateTime":"","startDateTime":""}}]}
```

Figure 4-27: New Sla Event



```
New Monitoring Data Event:  {"operatorID":"id_example","networkID":"network_slice_id","monitoringData":{"metricName":"osm_requests","metricValu
e":"500","transactionID":"7777","productID":"c6021d08-b934-492f-becb-59543aa9e4b6","instanceID":"2","resourceID":"X","timestamp":"2022-03-10"}}
Reading DB entry - key:c6021d08-b934-492f-becb-59543aa9e4b6
No violation
```

Figure 4-28: Monitoring data without violation

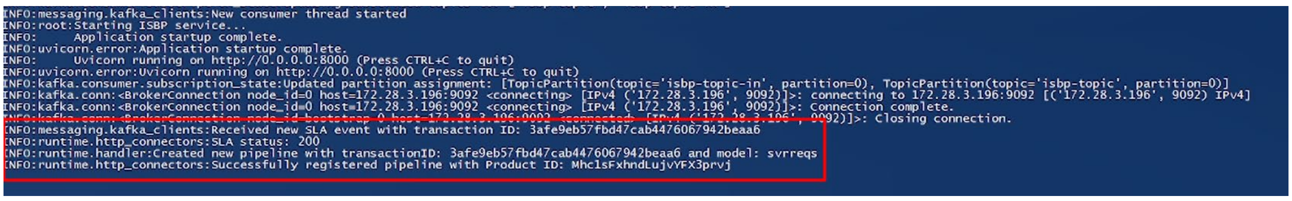


```
New Monitoring Data Event:  {"operatorID":"id_example","networkID":"network_slice_id","monitoringData":{"metricName":"osm_requests","metricValu
e":"2000","transactionID":"7777","productID":"c6021d08-b934-492f-becb-59543aa9e4b6","instanceID":"2","resourceID":"X","timestamp":"2022-03-10"}
}
Reading DB entry - key:c6021d08-b934-492f-becb-59543aa9e4b6
Violation Occurred: {
  "id": "9b9086b9-d0e8-4dc0-b132-204a088ffc2e",
  "productDID": "c6021d08-b934-492f-becb-59543aa9e4b6",
  "sla": {
    "id": "KNSkshon7Ss3BfyPqsfmW5",
    "href": "http://172.28.3.6:31080/smart-contract-lifecycle-manager/api/v1/service-level-agreement/KNSkshon7Ss3BfyPqsfmW5"
  },
  "rule": {
    "id": "abdeaf3a-d22e-4650-a8e5-d1d1bba64632",
    "metric": "osm_requests",
    "unit": null,
    "referenceValue": "1600",
    "operator": ".g",
    "tolerance": "100",
    "consequence": null
  },
  "violation": {
    "actualValue": "2000"
  }
}
```

Figure 4-29: Monitoring data with violation

4.6.2.5. *SLA breach prediction*

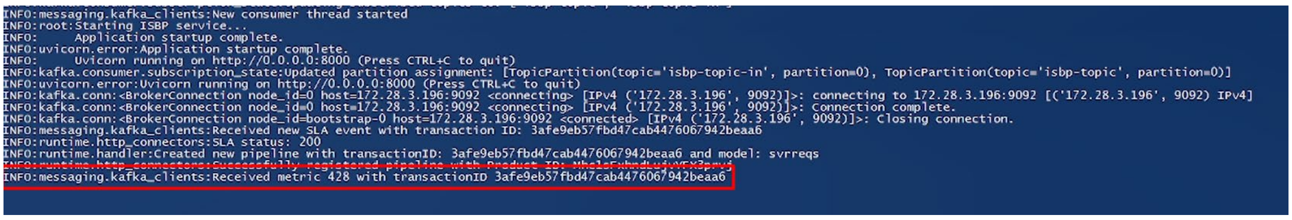
Figure 4-30 shows the initial configuration received by ISBP through a message queue. The configuration contains information that includes the product ID that was instantiated as well as the unique instantiation ID that will be used to correlate the incoming monitoring data later. Additionally, the SLA ID is used to retrieve the SLA details (monitored metric, threshold, etc.) from the platform repository.



```
INFO:messaging.kafka_clients:New consumer thread started
INFO:root:Starting ISBP service...
INFO:     Application startup complete.
INFO:uvicorn.error:Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:uvicorn.error:Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:kafka.consumer.subscription_state:Updated partition assignment: [TopicPartition(topic='isbp-topic-in', partition=0), TopicPartition(topic='isbp-topic', partition=0)]
INFO:kafka.conn:<BrokerConnection node_id=0 host=172.28.3.196:9092 <connecting> [IPv4 ('172.28.3.196', 9092)]>: connecting to 172.28.3.196:9092 [('172.28.3.196', 9092) IPv4]
INFO:kafka.conn:<BrokerConnection node_id=0 host=172.28.3.196:9092 <connecting> [IPv4 ('172.28.3.196', 9092)]>: Connection complete.
INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=172.28.3.196:9092 <connected> [IPv4 ('172.28.3.196', 9092)]>: Closing connection.
INFO:messaging.kafka_clients:Received new SLA event with transaction ID: 3afe9eb57fbd47cab4476067942beaa6
INFO:runtime.http_connectors:SLA status: 200
INFO:runtime.handler:Created new pipeline with transactionID: 3afe9eb57fbd47cab4476067942beaa6 and model: svrreqs
INFO:runtime.http_connectors:Successfully registered pipeline with Product ID: MhclsFxhndLujvYFX3prvj
```

Figure 4-30: SLA event received by ISBP

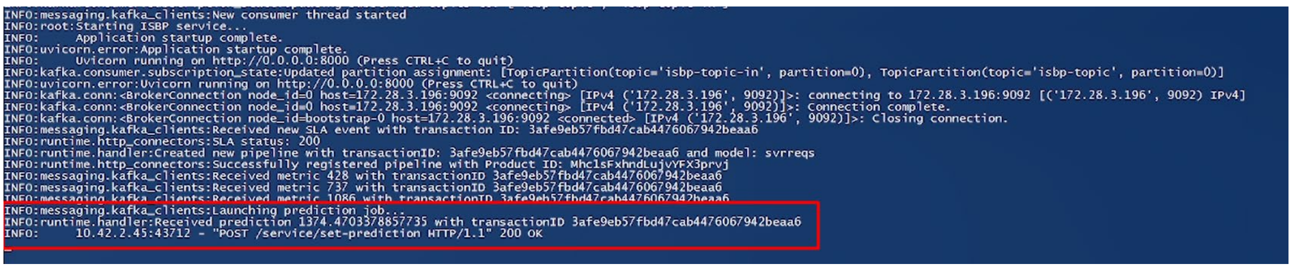
Figure 4-31 shows the reception of the monitoring data with the same transaction ID as in the configuration explained in Figure 4-30. After a number (in this case 3) of such metrics are received, a prediction job is expected to be launched.



```
INFO:messaging.kafka_clients:New consumer thread started
INFO:root:Starting ISBP service...
INFO:     Application startup complete.
INFO:uvicorn.error:Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:kafka.consumer.subscription_state:Updated partition assignment: [TopicPartition(topic='isbp-topic-in', partition=0), TopicPartition(topic='isbp-topic', partition=0)]
INFO:uvicorn.error:Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:kafka.conn:<BrokerConnection node_id=0 host=172.28.3.196:9092 <connecting> [IPv4 ('172.28.3.196', 9092)]>: connecting to 172.28.3.196:9092 [('172.28.3.196', 9092) IPv4]
INFO:kafka.conn:<BrokerConnection node_id=0 host=172.28.3.196:9092 <connecting> [IPv4 ('172.28.3.196', 9092)]>: Connection complete.
INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=172.28.3.196:9092 <connected> [IPv4 ('172.28.3.196', 9092)]>: Closing connection.
INFO:messaging.kafka_clients:Received new SLA event with transaction ID: 3afe9eb57fbd47cab4476067942beaa6
INFO:runtime.http_connectors:SLA status: 200
INFO:runtime.handler:Created new pipeline with transactionID: 3afe9eb57fbd47cab4476067942beaa6 and model: svrreqs
INFO:runtime.http_connectors:Successfully registered pipeline with Product ID: MhclsFxhndLujvYFX3prvj
INFO:messaging.kafka_clients:Received metric 428 with transactionID 3afe9eb57fbd47cab4476067942beaa6
```

Figure 4-31: Receiving metric with the source specified

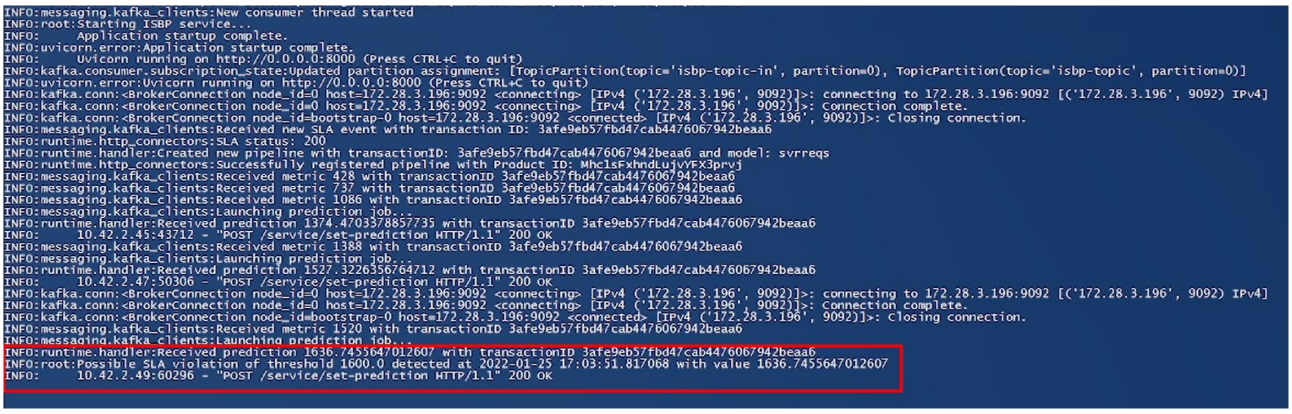
Indeed, a ML model predicts the next value in the sequence as shown in Figure 4-32. It should be noted that the input to the ML model(s) is a sliding window of the values, meaning that, for every prediction job, the first value of the previous input is discarded and the newest metric received, is added to the series.



```
INFO:messaging.kafka_clients:New consumer thread started
INFO:root:Starting ISBP service...
INFO:     Application startup complete.
INFO:uvicorn.error:Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:kafka.consumer.subscription_state:Updated partition assignment: [TopicPartition(topic='isbp-topic-in', partition=0), TopicPartition(topic='isbp-topic', partition=0)]
INFO:uvicorn.error:Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:kafka.conn:<BrokerConnection node_id=0 host=172.28.3.196:9092 <connecting> [IPv4 ('172.28.3.196', 9092)]>: connecting to 172.28.3.196:9092 [('172.28.3.196', 9092) IPv4]
INFO:kafka.conn:<BrokerConnection node_id=0 host=172.28.3.196:9092 <connecting> [IPv4 ('172.28.3.196', 9092)]>: Connection complete.
INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=172.28.3.196:9092 <connected> [IPv4 ('172.28.3.196', 9092)]>: Closing connection.
INFO:messaging.kafka_clients:Received new SLA event with transaction ID: 3afe9eb57fbd47cab4476067942beaa6
INFO:runtime.http_connectors:SLA status: 200
INFO:runtime.handler:Created new pipeline with transactionID: 3afe9eb57fbd47cab4476067942beaa6 and model: svrreqs
INFO:runtime.http_connectors:Successfully registered pipeline with Product ID: MhclsFxhndLujvYFX3prvj
INFO:messaging.kafka_clients:Received metric 428 with transactionID 3afe9eb57fbd47cab4476067942beaa6
INFO:messaging.kafka_clients:Received metric 737 with transactionID 3afe9eb57fbd47cab4476067942beaa6
INFO:messaging.kafka_clients:Received metric 1086 with transactionID 3afe9eb57fbd47cab4476067942beaa6
INFO:messaging.kafka_clients:Launching prediction job...
INFO:runtime.handler:Received prediction 1374.4703378857735 with transactionID 3afe9eb57fbd47cab4476067942beaa6
INFO:     10.42.2.45:43712 - "POST /service/set-prediction HTTP/1.1" 200 OK
```

Figure 4-32: First prediction generated, specifying the service it refers to.

Every SLA specifies the threshold of the metric that if exceeded is a violation of the SLA. This number is acquired from the SLA repository and every prediction is compared against it. Figure 4-33 shows that a prediction of 1636 is produced that exceeds the threshold of 1600 that is specified in the SLA. As a result, ISBP encapsulates the prediction number in a message that is dispatched to ISSM for further action.



```
INFO:messaging.kafka_clients:New consumer thread started
INFO:root:Starting ISBP service...
INFO:     Application startup complete.
INFO:uvicorn.error:Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:kafka.consumer.subscription_state:Updated partition assignment: [TopicPartition(topic='isbp-topic-in', partition=0), TopicPartition(topic='isbp-topic', partition=0)]
INFO:uvicorn.error:Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:kafka.conn:<BrokerConnection node_id=0 host=172.28.3.196:9092 <connecting> [IPv4 ('172.28.3.196', 9092)]>: connecting to 172.28.3.196:9092 [('172.28.3.196', 9092) IPv4]
INFO:kafka.conn:<BrokerConnection node_id=0 host=172.28.3.196:9092 <connecting> [IPv4 ('172.28.3.196', 9092)]>: Connection complete.
INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=172.28.3.196:9092 <connected> [IPv4 ('172.28.3.196', 9092)]>: Closing connection.
INFO:messaging.kafka_clients:Received new SLA event with transaction ID: 3afe9eb57fbd47cab4476067942beaa6
INFO:runtime.http_connectors:SLA status: 200
INFO:runtime.handler:Created new pipeline with transactionID: 3afe9eb57fbd47cab4476067942beaa6 and model: svrreqs
INFO:runtime.http_connectors:Successfully registered pipeline with Product ID: MhclsFxhndLujvYFX3prvj
INFO:messaging.kafka_clients:Received metric 428 with transactionID 3afe9eb57fbd47cab4476067942beaa6
INFO:messaging.kafka_clients:Received metric 737 with transactionID 3afe9eb57fbd47cab4476067942beaa6
INFO:messaging.kafka_clients:Received metric 1086 with transactionID 3afe9eb57fbd47cab4476067942beaa6
INFO:messaging.kafka_clients:Launching prediction job...
INFO:runtime.handler:Received prediction 1374.4703378857735 with transactionID 3afe9eb57fbd47cab4476067942beaa6
INFO:     10.42.2.45:43712 - "POST /service/set-prediction HTTP/1.1" 200 OK
INFO:messaging.kafka_clients:Received metric 1388 with transactionID 3afe9eb57fbd47cab4476067942beaa6
INFO:messaging.kafka_clients:Launching prediction job...
INFO:runtime.handler:Received prediction 1527.3226356764712 with transactionID 3afe9eb57fbd47cab4476067942beaa6
INFO:     10.42.2.47:50306 - "POST /service/set-prediction HTTP/1.1" 200 OK
INFO:kafka.conn:<BrokerConnection node_id=0 host=172.28.3.196:9092 <connecting> [IPv4 ('172.28.3.196', 9092)]>: connecting to 172.28.3.196:9092 [('172.28.3.196', 9092) IPv4]
INFO:kafka.conn:<BrokerConnection node_id=0 host=172.28.3.196:9092 <connecting> [IPv4 ('172.28.3.196', 9092)]>: Connection complete.
INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=172.28.3.196:9092 <connected> [IPv4 ('172.28.3.196', 9092)]>: Closing connection.
INFO:messaging.kafka_clients:Received metric 1520 with transactionID 3afe9eb57fbd47cab4476067942beaa6
INFO:messaging.kafka_clients:Launching prediction job...
INFO:runtime.handler:Received prediction 1636.7455647012607 with transactionID 3afe9eb57fbd47cab4476067942beaa6
INFO:root:Possible SLA violation of threshold 1600.0 detected at 2022-01-25 17:03:51.817068 with value 1636.7455647012607
INFO:     10.42.2.49:60296 - "POST /service/set-prediction HTTP/1.1" 200 OK
```

Figure 4-33: SLA threshold violation predicted

5. Installation Procedures

To facilitate the deployment of the 5GZORRO platform a dedicated activity has been started with aim of harmonizing, standardizing, and aligning the installation and deployment steps for the different platform elements, and for the different types of stakeholders.

To achieve this goal, the proposed approach relies on a series of assumptions regarding the underlying infrastructure and the organization of the entities and stakeholders within the ecosystem. These assumptions include, in particular, the following:

- The basic deployment infrastructure uses k8s as a platform for the deployment of various 5GZORRO components and software modules. More k8s clusters may be engaged to host different parts of the platform; the deployment model of single components is agnostic to the topology of the infrastructure.
- The interaction between the components deployed relies either on the REST APIs exposed by the different components or on the data streams implemented and managed by Kafka that is used as message bus between the different components (e.g., with the Data Lake).
- The deployment of the single components and their dependencies is organized in "profiles" representing the stakeholder roles and the cross-domain platform, where the shared components are represented.

In these regards, the implementation of the installation procedures relies on a set of Helm charts [43] that capture the deployment scripts for the single components, and group them together according to the specific stakeholder profile. Schematically, this approach is represented in Figure 5-1:

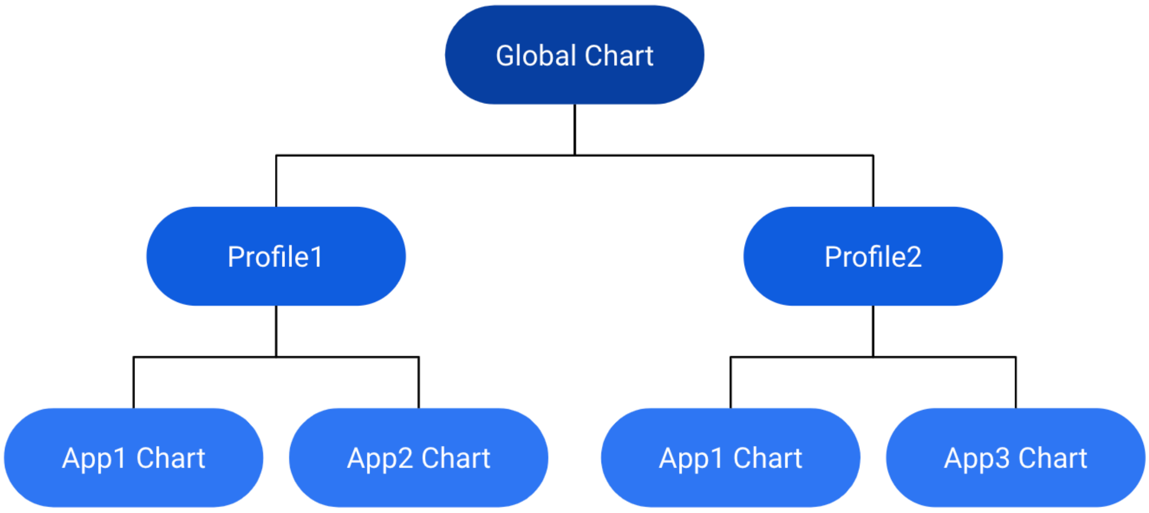


Figure 5-1: Hierarch of deployment Helm charts

In this deployment model the profiles organize single components into logically consistent set of elements representing the stakeholders in a specific platform usage scenario or the cross-domain elements used by the different stakeholders. Correspondingly, we distinguish the following profiles:

- Cross-domain platform profile: Represents the cross-domain analytics and intelligence services as common elements used for related tasks by all the 5GZORRO domains.

- Administrative profile: Represents the components used for the platform governance and for the tasks behind the governance activities. It also includes the same capabilities contained in the Trader Profile, to allow admin stakeholders to also participate in the resource and service trading.
- Regulator profile: The set of components that are used by the platform regulator role. Unlike the admin profile, provider and consumer capabilities are not expected in the regulator profile.
- Trader profile: Represents a generic operator participating to the 5GZORRO ecosystem. This profile uses the components that are necessary to perform the resource provisioning (Provider activities) as well as the resource consumption (Consumer activities).

Please note that this classification is orthogonal to the platform functionalities and is centred around the platform deployment as performed by the different parties. The development of the deployment scripts/charts is a current ongoing work that is planned to be finalized by the end of the project and published in the project repo.

5.1.**Cross-domain Platform Profile**

This profile includes the components of the platform that provide cross-domain analytics and intelligence services and should be deployed as a pre-requisite dependency to all the other profiles. The profile includes, in particular:

- **Data Lake components**. As described in Section 4.1, this set of components includes a variety of the software tools required to persist the data (es., Minio S3 Object Storage, Postgres, MongoDB), Metadata services (Data Catalogue and related components and APIs), Workflow managers for the pipeline deployment and execution on top of the k8s cluster (Argo).
- **Data and event streaming**. To support a variety of scenarios ranging from the monitored data collection to the event-based activation of different processes, Apache Kafka (and related HA SW, such as Zookeeper) is used and made available across the platform. It is important to note that this infrastructural component is used by both the components of the WP3 and those of WP4.
- Components for **Cross-domain Analytics and Intelligence**. This includes components such as ISBP (Section 4.4) and SLA Monitoring (Section 4.3) and SRSD application (Section 4.5).

5.2.**Administrative Profile**

Administrative profile groups together the components related to the platform governance activities, including also the applications required to support the multi-party resource and service trading. This profile includes:

- **Governance DLT:** The underlying VON Network (based on Hyperledger Indy) used as a supporting infrastructure for the Identity and Permission Manager.
- **Marketplace DLT:** The underlying Corda Network used to support the realisation of a decentralised Marketplace, including the domain node and other common artifacts (e.g., the Notary).
- **Governance & Marketplace Portal**: The Web GUI of the operator portal customized to the Admin profile configuration, comprising the governance and marketplace-related functionalities.
- **ID&P:** In case of Admin Profile, the deployment refers to the Hyperledger Aries Cloud Agent (ACA-Py) with the Admin profile parameter configuration.
- **LPR**: An application used by the other module to manage the legal templates.

- **RSOC**: The implementation of the TM Forum Product Catalogue Offering API.
- **SCLCM**: The component used to manage SLA, licensing, and products within the operator profile domain and across the underlying DLT. Depends on the Marketplace DLT.
- **MDA**. The component used by the profile to collect and store within the Data Lake monitored information from resources provisioned at the admin domain.

5.3.Regulator Profile

The Regulator profile comprises the components related to some of the governance activities carried out by regulators of the network. The profile includes:

- **Marketplace DLT:** The Corda DLT node required to participate in the Marketplace workflows in a distributed manner.
- **Governance & Marketplace Portal**: The Web GUI of the operator portal customized to the Admin profile configuration, comprising the governance and marketplace-related functionalities.
- **ID&P:** In case of Regulator Profile, the deployment refers to the Hyperledger Aries Cloud Agent (ACA-Py) with the Regulator profile parameter configuration.
- **LPR**: An application used by the other module to manage the legal templates.
- **RSOC**. The implementation of the TM Forum Product Catalogue Offering API.
- **SCLCM**. The component used to manage agreements, SLA, licensing, and products within the operator profile domain and across the underlying DLT. Depends on the Marketplace DLT.

5.4.Trader Profile

The Trader profile comprises the components required for the management of offers and their agreements, to support the multi-party resource and service trading. This profile includes, in particular:

- **Marketplace DLT:** The Corda DLT node required to participate in the Marketplace workflows in a distributed manner.
- **Marketplace Portal**: The Web GUI of the stakeholder portal comprising the marketplace-related functionalities.
- **ID&P:** In case of Trader Profile, the deployment refers to the Hyperledger Aries Cloud Agent (ACA-Py) with the Trader profile parameter configuration.
- **LPR**: An application used by the other modules to manage the legal templates.
- **RSOC:** The implementation of the TM Forum Product Catalogue Offering API.
- **SCLCM**: The component used to manage SLA, licensing, and products within the operator profile domain and across the underlying DLT. Depends on the Marketplace DLT.
- **MDA**: The component used by the profile to collect and store within the Data Lake monitored information from resources provisioned at the trader domain.

5.5. Components Mapping

The following table summarizes the deployment configurations of the different profiles with respect to the components of WP3.

Table 5-1: WP3 components required per stakeholder role

	Cross-domain	Admin	Regulator	Trader
Governance DLT		X		
Marketplace DLT		X	X	X
Governance Portal		X	X	
Marketplace Portal		X	X	X
ID&P		X	X	X
LPR		X	X	X
RSOC		X	X	X
SCLCM		X	X	X
MDA		X		X
Operational Data Lake	X			
SLA Monitoring	X			
ISBP	X			
SRSD	X			

It is important to note that, besides the aforementioned components, the profiles also include the components for zero-touch automation described in WP4 as reported in [21].

6. Conclusions

This deliverable reports on the software prototypes that have been developed to implement the 5GZORRO Evolved 5G Service layer functionalities, and which contains the Governance Platform, the Marketplace Platform, and the Cross-domain Analytics & Intelligence for AIOps Platform. This document aims at summarizing direct mappings of software modules to functional architectural entities presented in preceding deliverables from WP2 and WP3 design documents.

In summary, the document collects supported functionalities and prototype implementation details of the software produced within WP3, with links to available documentation and code repositories related to the various components released on GitHub. Additionally, validation tests including functional verification reports and usage scenarios, together with installation procedures are provided for each one of the 5GZORRO Evolved 5G Service layer platforms.

The prototypes presented in this document corresponds to the final release consolidated in the context of WP3. Required fixes and adaptations on the various modules derived from ongoing and subsequent integration and validation work in WP5 (*Validation through Use Cases*), will be realized as updates to the software code and documentation material included in corresponding GitHub repositories.

7. References

[1] 5GZORRO Consortium, Deliverable D3.1: Design of the evolved 5G Service layer solutions. Available online: https://www.5gzorro.eu/wp-content/uploads/2021/11/5GZORRO_D3.1_v1.9-final.pdf

[2] 5GZORRO Consortium, Deliverable D3.3: Final design of the evolved 5G Service layer solutions.

[3] 5GZORRO Consortium, Deliverable D2.4: Final design of the 5GZORRO Platform for Security & Trust.

[4] 5GZORRO GitHub space. Available online: https://github.com/5GZORRO

[5] Sovrin. Available online: https://sovrin.org/

[6] Digital Government - British Columbia. Available online: https://digital.gov.bc.ca/

[7] VON Network. Available online: https://github.com/bcgov/von-network

[8] Hyperledger Indy. Available online: https://www.hyperledger.org/use/hyperledger-indy

[9] Hyperledger Aries. Available online: https://www.hyperledger.org/use/aries

[10] Hyperledger Aries Cloud Agent Python (ACA-Py). Available online: https://github.com/hyperledger/aries-cloudagent-python

[11] Decentralized Identity Foundation. Available online: https://identity.foundation/

[12] Identity and Permissions Manager. Available online: https://github.com/5GZORRO/identity

[13] Spring Boot @Spring.io. Available online: https://spring.io/projects/spring-boot

[14] OpenAPI Initiative. Available online: https://www.openapis.org/

[15] PostgreSQL: The World's Most Advanced Open-Source Relational Database. Available online: https://www.postgresql.org/

[16] JSON Template Language. Available online: https://github.com/etclabscore/json-template-language

[17] Docker. Available online: https://www.docker.com/

[18] Kubernetes: Production-Grade Container Orchestration. Available online: https://kubernetes.io/

[19] Legal Prose Repository. Available online: https://github.com/5GZORRO/legal-prose-repository

[20] Governance and Marketplace Portal. Available online: https://github.com/5GZORRO/GUI

[21] 5GZORRO Consortium, Deliverable D4.3: Final prototype of Zero Touch Service Mgmt with Security and Trust.

[22] R3 Corda. Available online: https://www.corda.net/

[23] Resource and Service Offer Catalogue. Available online: https://github.com/5GZORRO/resource-and-service-offer-catalog

[24] Smart Contract Lifecycle Manager. Available online: https://github.com/5GZORRO/smart-contract-lifecycle-manager

[25] Open Data Hub. Available online: https://opendatahub.io/

[26] Apache Hadoop. Available online: https://hadoop.apache.org/

[27] Apache Spark. Available online: https://spark.apache.org/

[28] Argo Workflows. Available online: https://argoproj.github.io/workflows

[29] Minio: Multi-Cloud Object Storage. Available online: https://min.io/

[30] Apache Kafka. Available online: https://kafka.apache.org/

[31] Datalake Server API. Available online: https://htmlpreview.github.io/?https://github.com/5GZORRO/datalake/blob/master/datalake_api.html

[32] Datalake Server. Available online: https://github.com/5GZORRO/datalake/tree/master/python-flask-server/swagger_server

[33] Data Ingestion Pipeline. Available online: https://github.com/5GZORRO/datalake/tree/master/pipelines/ingest

[34] Data Lookup Service API. Available online: https://htmlpreview.github.io/?https://github.com/5GZORRO/datalake/blob/master/services/catalog/datalake_catalog_api.html

[35] Data Lookup Service. Available online: https://github.com/5GZORRO/datalake/tree/master/services/catalog

[36] Data Streaming Service API. Available online: https://htmlpreview.github.io/?https://github.com/5GZORRO/datalake/blob/master/services/stream_data/stream_data.html

[37] Data Streaming Service. Available online: https://github.com/5GZORRO/datalake/tree/master/services/stream_data

[38] Monitoring Data Aggregator. Available online: https://github.com/5GZORRO/mda

[39] SLA Monitoring. Available online: https://github.com/5GZORRO/SLA-Monitor

[40] Intelligent SLA Breach Predictor. Available online: https://github.com/5GZORRO/sla-breach-predictor

[41] Smart Resource and Service Discovery. Available online: https://github.com/5GZORRO/Smart-Resource-and-Service-Discovery-application

[42] H2O library. Available online: https://h2o.ai/

[43] Helm Charts: packaging format for Kubernetes. Available online: https://helm.sh/docs/topics/charts/

[44] RITA (NLP) framework. Available online: https://rednoise.org/rita/

[45] spaCy ML backend system. Available online: https://spacy.io/

[46] scikit-learn ML library. Available online: https://scikit-learn.org/stable/

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

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

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

[50] Product Ordering Management API REST Specification, TM Forum Specification, TMF622, Release 19.0.1, November 2019

[51] Geographic Address Management API REST Specification, TM Forum Specification, TMF673, Release Release 17.5.0, January 2018.

[52] Party Management API REST Specification, TM Forum Specification, TMF632, Release 16.0.1, Release 19.0.0, June 2019.

8. Abbreviations

AIOps	Artificial Intelligence for IT operations
CNF	Cloud Native Function
DID	Decentralized Identifier
DIF	Decentralised Identity Foundation
DLT	Distributed Ledger Technology
DSS	Data Streaming Service
EC	European Commission
eLM	e-Licensing Manager
ID&P	Identity & Permissions Manager
ISBP	Intelligent SLA Breach Predictor
ISSM	Intelligent Slice and Service Manager
k8s	Kubernetes
LPR	Legal Prose Repository
MDA	Monitoring Data Aggregator
ML	Machine Learning
NBI	Northbound Interface
NFT	Non-Fungible Token
NFV	Networks Function Virtualization
NLP	Natural Language Processing
NSMM	Network Service Mesh Manager
NSSO	Network Slice and Service Orchestrator
ODH	Open Data Hub
RAN	Radio Access Network
RSOC	Resource and Service Offer Catalogue
SBI	Southbound interface
SCLCM	Smart Contract Lifecycle Manager
SLA	Service Level Agreement
sRM	Spectrum Resource Manager
SRSD	Smart Resource and Service Discovery
VC	Verifiable Credential
VNF	Virtual Network Function
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WP	Work Package
xNF	Any Network Function
xRM	Any Resource Manager

<END OF DOCUMENT>