5GZORRO

Grant Agreement 871533

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

D3.3: Final design of the 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:	Project Acronym:	Project title:
871533	**5GZORRO**	**Zero-touch security and trust for ubiquitous computing and connectivity in 5G networks.**

Lead Beneficiary:	Document version:
IBM	**V4.1**

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

Deliverable title:
D3.3: Final design of the evolved 5G Service layer solutions

Start date of the project:	Contractual delivery date:	Actual delivery date:
01/11/2019 (duration 30 months) (extended to 36 months)	**30.04.2022**	**16.05.2022**

Editor(s)
Katherine Barabash (IBM)

List of Contributors

Participant	Short Name	Contributor
IBM Israel Science and Technology	IBM	Katherine Barabash, Kalman Meth, David Breitgand
Nextworks	NXW	Pietro G. Giardina, Michael De Angelis, Giacomo Bernini
Fundaciò i2CAT	I2CAT	Adriana Fernández-Fernández, Carlos Herranz, Daniel Bautista, S. Siddiqui
Telefonica Investigación Investigacióny Desarrollo	TID	D. R. López, C. Rodríguez Cerro
Ubiwhere	UW	Filipa Martins, Francisco Santos
Fondazione Bruno Kessler	FBK	Rasoul Behravesh
Universidad de Murcia	UMU	G. Martínez Pérez, M. Gil Pérez, J.M. Jorquera Valero, P. M. Sánchez Sánchez
Altice Labs	ALB	Bruno Santos
Intracom	ICOM	Dimitris Laskaratos, Marinela Mertiri
Atos Spain	ATOS	Guillermo Gomez, Fernando Bravo
Malta Communications Authority	MCA	Jean-Marie Mifsud, Antoine Sciberras, Andrew Caruana, Charlo Baldacchino

List of Reviewers

Participant	Short Name	Contributor
Fundaciò i2CAT	I2CAT	A. Fernández-Fernández
Fondazione Bruno Kessler	FBK	Rasoul Behravesh
Nextworks	NXW	G. Bernini
Fundaciò i2CAT	I2CAT	M. S. Shuaib

Change History

Version	Date	Partners	Description/Comments
0.0	05-04-2022	IBM, NXW	Table of Contents (and editing assignments)
0.1	19-04-2022	IBM, i2CAT, NXW	Updated table of contents and assignments
0.1-nxw	19-04-2022	NXW	Initial contents of DLT (2.1) and Governance Services (3.1)
0.2	21-04-2022	IBM	More initial text and guidelines for further contributions
0.3	27-04-2022	ATOS IBM NXW ALB	Update Smart Contract and eLicencingl data model (4.2 and 4.6); Update Data Lake APIs (2.2) Legal prose module (3.1.2); Resource data models (4.4.1 and 4.4.3) Identity and permission service details (3.2 and 4.1)

		I2CAT	Trustworthy Marketplace Services (3.3)
0.4	01-05-2022	I2CAT ICOM UW ALB	Updates to 3.3.1, 3.3.2; 3.44; 4.3; 4.4 Updates to 3.4.3 Update 3.4.3, 3.3.3, and 3.1.3 Updates to 3.1.3, 3.2.5, and 3.4.2
1.0-rev Ready for tech review	03-05-2022	IBM NXW UMU	Delete section 3.4.1 (contents are redundant); update 2.2 Updates to 3.2.2 Updated to 3.3.3; revision of 4.6
1.0.1-rev	08-05-2022	I2CAT ALB MCA IBM	Provide tech review comments Update 2.1 and 3.1 Revise contents related to the regulator considerations (e.g., spectoken) Process tech review comments and new contributions
2.0.2-rev	10-05-2022	I2CAT NXW UW IBM	Address review comments Address review comments Address review comments Process tech review outcomes and update references/formatting across the document
3.0.3-rev	15-05-2022	FBK NXW IBM	Provide review comments Provide review comments Address review comments
4.0.4-rev	16-05-2022	I2CAT ALB ICOM IBM	Address review comments in 3.2.2 Address review comments in 3.2.1 Address review comments in 3.3.2 Incorporate all the updates
4.0	16-05-2022	I2CAT	Final submission.
4.1	06-03-2023	NXW	Typos fixed

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 deliverable describes the design of the evolved 5G Service layer solutions, in its final form as being realised in the 5GZORRO platform. The document builds upon the structure and the contents of D3.1 where the preliminary architecture and design choices were presented. It is intended to be a standalone version of 5GZORRO Service layer solutions design, aligned with the final version of 5GZORRO architecture reported in WP2. In addition to WP3 components design, the document includes descriptions of major technology enablers used to design and realise the 5GZORRO Service layer solutions, showing why these concepts (the DLT and the AIOPs) were chosen and how they have been applied in 5GZORRO context, both for realizing 5G Service layer solutions of WP3 (*Identity and Permissions, Governance, Marketplace, Cross-domain Analytics & Intelligence*) and in a broader context of *Zero-touch Service Management and Orchestration* developed in WP4.

This final version of 5G Service layer solutions design is informed by the implementation activites performed across WP3 and WP4 (organized as several goal-oriented software development teams) as well as by the integration and validation activities ongoing as part of WP5 (*Validation through Use Cases*).

To summarize, this deliverable complements the final version of 5GZORRO architecture, providing deeper level of detail for components and services contributed by WP3 along with their internal design and interworkings that will be further detailed in the last deliverable of WP3, D3.2, with low level descriptions of the software prototypes of evolved 5G Service layer solutions.

1 Introduction

This document describes the design of 5G Service layer solutions provided by the 5GZORRO platform to enable zero-trust secure and intelligent automation of 5G networks and services.

This document is a revision of the initial version of the 5G Service layer solutions design delivered in D3.1 and is created to be standalone and self contained and to reflect the evolved 5GZORRO Platform architecture delivered in D2.4. We include the summary of the differences between the D3.1 and the D3.3 structure and contents in Section 1.1 below.

As a baseline for presenting the architectural decisions and describing the components, the document contains description of the underlying technology enablers, namely the Distributed Ledger Technologies (DLT) and the Operational Data Lake for 5G AIOps. The report discusses why these concepts are needed and what they are used for and describes the domain-specific systems architecture of DLT and the 5G Operational Data Lake (also referred to as the Data Lake or datalake in the rest of the document) applied and realised in 5GZORRO.

In addition, the document provides the next level of detail over the whole 5GZORRO architecture described in deliverable D2.4 [1], focusing on each one of the 5G Service layer solution components, its internal design, interfaces, information models, addressing both the per-domain and the cross-domain operational aspects.

Referring to the overall 5GZORRO architecture described in deliverable D2.4 [1] and shown here in Figure 1-1, this document covers:

1. The decsriptions of why and how 5GZORRO WP3 applies the DLT and the Dala Lake as enabling techlologies to create the 5GZORRO Marketplace DLT Platform, the 5GZORRO Identity and Trust DLT Platform, and the 5GZORRO Operational Data Lake Platform.
2. The descriptions of the 5G Service layer solutions WP3 contributes to the 5GZORRO Logical Subsytems. Mainly, these solutions belong to the Marketplace and Business, the Security and Trust, and the Analytics and Intelligence Logical Subsytems and provide supporting functions for the Zero-touch Service Management and Orchestration Logical Subsytem. The descriptions here are design level and cover the purpose, the functional decomposition, and the interfaces of each solution or service.
3. The information models used to communicate across the 5G Service layer solutions in WP3, including messages exchanged over the 5GZORRO communication fabrics, the operational data stored and processed in the 5GZORRO Operational Data Lake, and the objects describing the Marketplace resources.

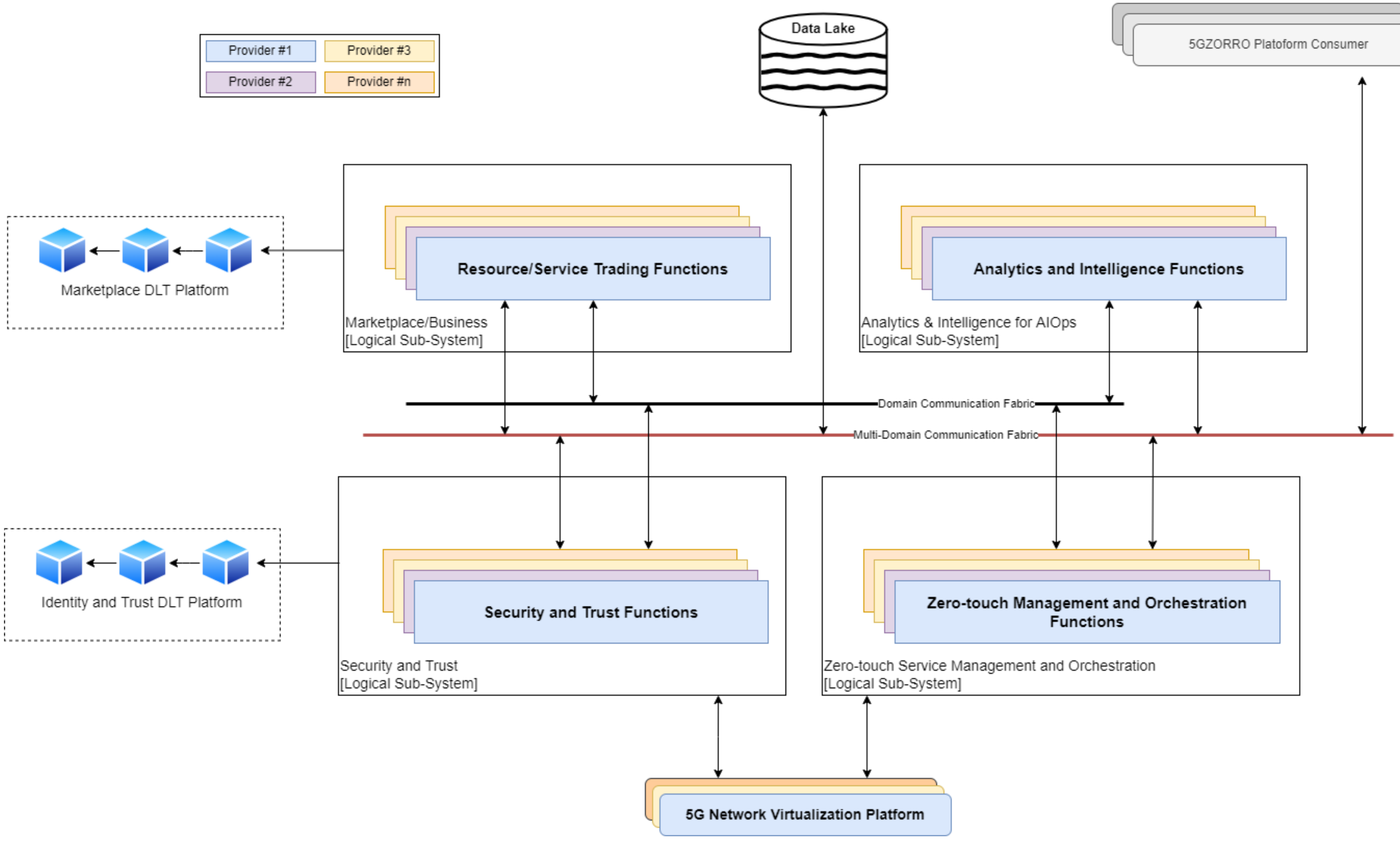
Provider #1
Provider #3
Provider #2
Provider #n
Data Lake
5GZORRO Platoform Consumer
Marketplace DLT Platform
Resource/Service Trading Functions
Marketplace/Business [Logical Sub-System]
Analytics and Intelligence Functions
Analytics & Intelligence for AIOps [Logical Sub-System]
Domain Communication Fabric
Multi-Domain Communication Fabric
Identity and Trust DLT Platform
Security and Trust Functions
Security and Trust [Logical Sub-System]
Zero-touch Management and Orchestration Functions
Zero-touch Service Management and Orchestration [Logical Sub-System]
5G Network Virtualization Platform

Figure 1-1: 5GZORRO High Level reference architecture

1.1 Summary of changes over D3.1

This document is a revision of the initial version of the 5G Service layer solutions design delivered in D3.1 and is created to be standalone and self contained. Compared to D3.1, this revision has been:

1. **Restructured**. The first reason for updating the document structure was to separate the description of the underlying enabling technologies (the DLT and the Data Lake) and their use in 5GZORRO from the 5GZORRO Platform Services implemented to realize the delivered 5G Service layer solutions. The second reason for some of the minor restructure was to align the presentation with the updated version of 5GZORRO architecture delivered in D2.4. The third reason was to separate all the Information Model specifications into a separate section.
2. **Updated**. First, the text was updated to exclude the discussion about components and implementations that were considered for use initially but were not included in the final design, so that it its current form the document only referes to components, implementations, and specification relevant to the final design of the 5GZORRO Platform. Additional source of update is descriptions of new and updated components, APIs and Information Models that were added or revised during the second phase of the project, informed by implementation and integration activities and by the evolution of the 5GZORRO platform architecture (D2.4). Examples of such major updates are: descriptions of the 5GZOROO User Portals, the Governance Portal and the Marketplace Portal; descriptions of all the 5GZORRO Resource Models and, especially, the modelling of the radio spectrum resource; the Monitoring Data Agregator component, etc. Most of the text describing the technology enablers, the DLT and the Data Lake, was revised and updated to reflect the eventual usage by the 5GZORRO platform as well as the final choices made with respect to concrete technlologies and projects leveraged by 5GZORRO.

1.2 Document outline

The document is structured as follows:

- Section 2 describes the two major technology enablers adopted for realizing the 5GZORRO amibition and presents their domain-specific realizations in the form of 5GZORRO DLT Platform and the 5GZORRO Operational Data Lake. This section inherits some text from D3.1 but in general is mainly revised.
- Section 3 is the main technical section and contains a subsection for each specific 5G Service layer solution, describing its purpose and role, its component-level design, as well as its internal and external interfaces. The section describes the Governance Services, the Trustworthy Marketplace Services, as well as the Analytics and Intelligence services. This section is restructured to follow the presentation of D2.4 and revised to describe the updated design of 5GZORRO services.
- Section 4 contains the information models used for communications among the 5G Servce layer solutions and across the whole 5GZORRO platform. In previous revision, the information models were spread throughout the document; here, we devote a separate section to them, and add/update the models reflect the final design.
- Section 5 concludes and summarises the document.
- Section 6 lists the references.
- Section 7 lists the abbreviations used that appear in the document.

2 Technology Enablers for the Evolved 5G Service Layer Solutions

This section describes how two main enabling technologies, the DLT and the Operational Data Lake, were applied in 5GZORRO for implementing the advanced capabilities that underly the 5GZORRO platform and infuse security, trust, and data-driven intelligence across multiple 5G domains and across all four of the 5GZORRO logical subsystems shown in Figure 1-1 – the Marketplace and Business, the Security and Trust, the Analytics and Intelligence, and the Zero-touch Service Management and Orchestration.

2.1 DLT Platform

Based on a thorough review of the suitability of DLTs for the 5GZORRO platform, two categories of DLT requirements were identified, one to support marketplace resource and service trading and the other to support cross-domain identity and permissions. Lengthy investigation led to the conclusion that whilst a single DLT COULD be used to satisfy both use cases, the decentralised identity requirement represented an opportunity to leverage a DLT developed specifically for this use case.

Several strong candidates in R3 Corda [30], Hyperledger Fabric [32] and Quorum [31] were reviewed for supporting the 5GZORRO marketplace trading of resources/services, the associated agreement lifecycles and SLA enforcement. It is likely that all three DLTs shortlisted for review would be capable of servicing the requirements imposed by the 5GZORRO marketplace. However, R3 Corda was favoured as the exemplary implementation on account of the distinguishing features described below.

Hyperledger Indy [17] has been built from the ground up to support the decentralized identity use case and as such has been selected to realise a governance DLT to manage Decentralized Identifiers (DIDs) and the issuance/verification of Verifiable Claims to support identity and permission requirements. It was decided that this was preferable to the alternative of utilising a single DLT to satisfy both 5GZORRO use cases, which would require significant effort to design and implement features already offered out-of-the-box by Hyperledger Indy.

2.1.1 DLT for 5GZORRO Markeplace

Each of the afore-mentioned DLTs provide features to satisfy enterprise requirements to varying degrees. Analysis covered both technical and business-level considerations around privacy, performance, and project/community health.

R3 Corda takes a novel approach in that ledger state ("facts") is only shared amongst network participants on a need-to-know basis. To this end, it is ideally placed to offer the privacy required to support agreements between stakeholders in the 5GZORRO marketplace.

The adoption of the UTXO model (unspent transaction output[1]) means that Corda achieves high transaction throughput and transactions only need to be executed by the interested parties rather than all ledger participants. Quorum achieves much lower transaction rates and whilst Hyperledger Fabric is performant, its requirement for channels and side-channels to achieve the same level of privacy make it less suitable. In contrast to Hyperledger Fabric and Quorum, Corda is byzantine fault tolerant out-of-the-box.

Corda has a concept of Flows that enables business logic to be coded at the DLT level for orchestrating complex business processes involving multi-stage ledger state changes and the gathering of signatures from

[1] R2 Corda, What is a UTXO?, available online at: https://www.r3.com/blog/what-is-a-uxto/

participants and the notary pool. Checkpointing of these flows means that a flow can – for example – recover when a participant node is unavailable, whereby a flow state will be checkpointed, persisted, and retried. This DLT-native orchestration is not a feature of either Quorum or Hyperledger Fabric.

Corda has a strong focus on legally enforceable smart contracts, where legal prose is associated with a given contract state. This clearly is a key facet to ensure 5GZORRO can trust in the enforceability of business agreements made through the 5GZORRO platform.

2.1.1.1 *Relevant Entities and Modules*

The Corda DLT infrastructure is realised through the utilisation and implementation of both application and network elements offered by the open-source platform.

Implementation of the contract and business logic is achieved through contract states, contracts, and flows. These are the elements that will enable the autonomous lifecycle of marketplace agreements and SLAs and ensure that ledger state transitions evolve only in accordance with the rules defined in smart contracts. Figure 2-1 illustrates how each of the DLT elements outlined below contribute to a ledger update, from RPC call, through to contract execution, verification and finally vault update (commitment of the transaction).

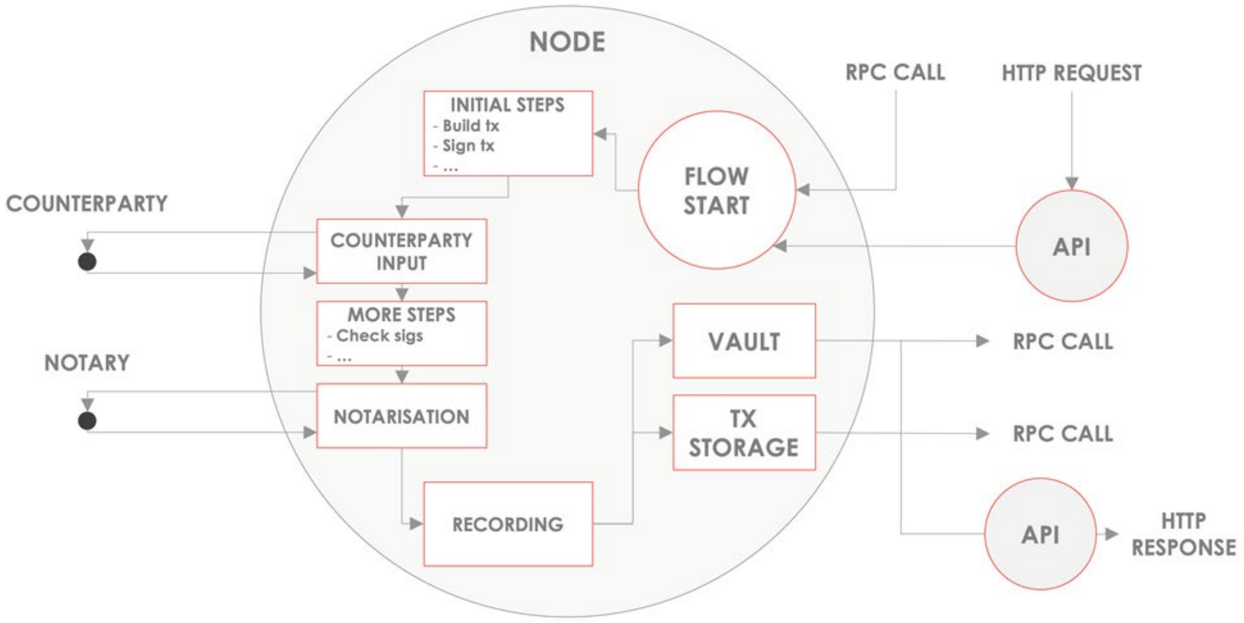

Figure 2-1: Corda Node

Contract States & Vault: In Corda, ledger states are only distributed on a need-to-know basis, i.e., a transaction involving two participants would be executed by them and on notarisation (verification by the notary) committed to each participant node's vault; the vault is a Postgres database where all state is stored by a given node. Contract states are classes that encapsulate on-ledger facts and as such will be used to represent several 5GZORRO artifacts such as products and SLAs.

Contracts: Contracts are deterministic classes that govern the state transitions and parties required to sign transactions before they can be committed to the ledger. Contracts are contract state companions and will be used to define the governance rules over valid state transitions of the ledger, e.g., what parties are permitted to update a 5GZORRO agreement or record an SLA violation.

Flows: Flows are used to define multi-step processes that orchestrate the composition and agreement (signing) of transactions. For example, flows will be developed to support marketplace operations such as registering product offers and negotiating agreements. Each defined flow will programmatically define the input/output states of a transaction, mediate the negotiation of agreements, collection of stakeholder signatures and notarisation prior to committing the transaction to the ledger. Flows can also comprise of sub-flows, acting as a means of initiating subsequent actions such as billing events on termination of an agreement.

CorDapp & RPC: CorDapps are distributed applications that run on each corda node. They encapsulate the flows, the contracts, and the contract states. Once a CorDapp is deployed to a node, the owner can invoke it's flows and query vault state over RPC. It is via this RPC interface that the 5GZORRO service layer will interact with the DLT.

Tokens: R3 Corda implements tokens within the contract logic. Corda permits the creation of tokens and the control of its lifecycle with three commands: issue, move, and redeem. A token in R3 Corda is seen as a combination of contract state, linear state, ownable state and fungibility. On the last aspect, Corda currently supports fungible, non-fungible, and evolvable tokens, being the latter a special case of the non-fungible tokens (NFT) that allows maintainers (i.e., issuers of the token) to update a subset of the token attributes after meeting some specific criteria defined in the contract flows.

There are also certain Corda network components that will need to be deployed to realise the 5GZORRO DLT infrastructure.

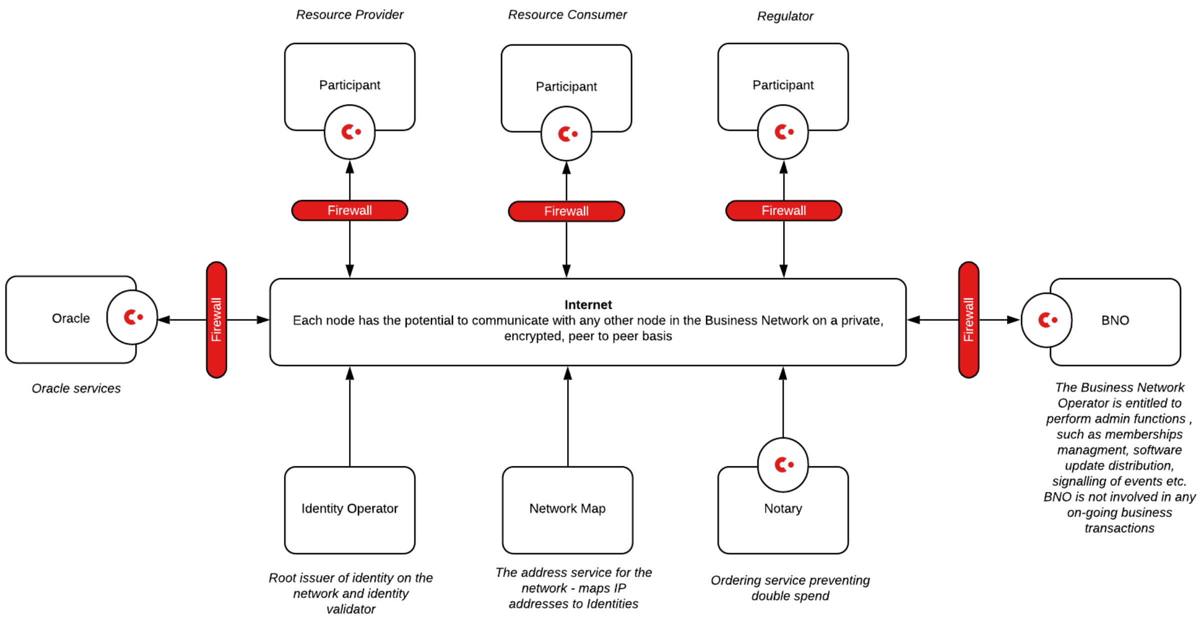
Resource Provider
Resource Consumer
Regulator
Participant
Participant
Participant
Firewall
Firewall
Firewall
Oracle
Oracle services
Firewall
Internet
Each node has the potential to communicate with any other node in the Business Network on a private, encrypted, peer to peer basis
Firewall
BNO
The Business Network Operator is entitled to perform admin functions , such as memberships managment, software update distribution, signalling of events etc. BNO is not involved in any on-going business transactions
Identity Operator
Root issuer of identity on the network and identity validator
Network Map
The address service for the network - maps IP addresses to Identities
Notary
Ordering service preventing double spend

Figure 2-2: High-level Corda Network Architecture

Network Map (Cordite): A network map server is required for nodes to register and discover one another on a Corda network. As part of the DLT infrastructure a Cordite Network Map server will be deployed and operated by the DLT operator. Cordite is an open-source implementation of Corda's Network Map and Doorman protocols that govern access to the network. This service facilitates the administration of stakeholder node participation on the Corda network and addition/removal of notaries.

Participant Nodes: Each 5GZORRO stakeholder (provider, consumer, regulator etc.) will deploy a Corda node and the 5GZORRO CorDapps to execute (via RPC client). It will be configured to use the 5GZORRO network map to discover other network participants and notary services.

Notary: To ensure that transactions are valid and no double-spends have occurred, the DLT operator will deploy a Notary pool of one or more nodes. The notary will sign all transactions prior to being committed to the ledger.

Figure 2-2 gives a high-level overview of the architecture of a Corda network, the key nodes and services that are deployed and their role. Each stakeholder will deploy nodes with the 5GZORRO CorDapp(s) that encapsulate the flows to achieve the necessary ledger updates for the marketplace.

2.1.2 DLT for distributed identities

As introduced in deliverable D2.4 [1], 5GZORRO has adopted the W3C Decentralised Identifiers (DIDs) [13] and associated Verifiable Credentials (VCs) [14] as key enablers to implement trusted interactions across domains. The W3C standardisation work is evolving with the support of two major reference implementations initiatives covering different functionalities: Linux Foundation Hyperledger Identity "stack" and Decentralised Identity Foundation.

The Linux Foundation Hyperledger Identity "stack" was created from original Sovrin Foundation [15] code base and is comprised of three main projects presented in Figure 2-3:

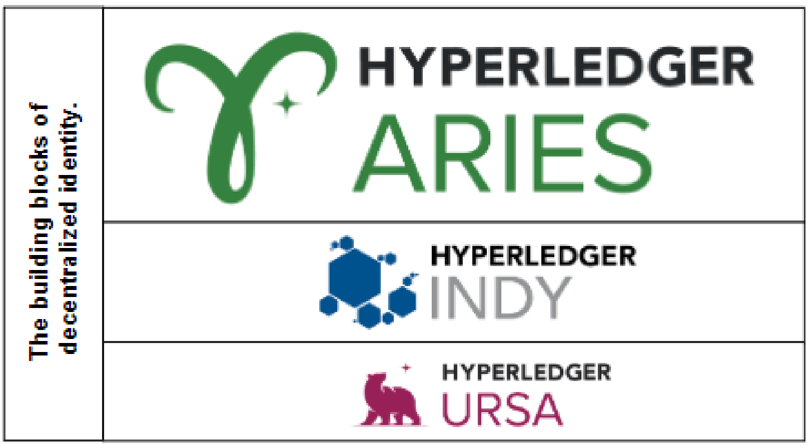

Figure 2-3: Linux Foundation Hyperledger Identity "stack": URSA, INDY and ARIES

- The **Hyperledger URSA** [16] project provides cryptographic primitives. Ursa packages the primitives in a way that can be consumed by Indy, Aries and any other software that needs a solid, vetted cryptographic base.
- The **Hyperledger Indy Node** [17] project 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.
- The **Hyperledger ARIES** [18] complements Hyperledger Indy Node by providing a toolkit to create DID Agents that manage the creation, transmission, storage and verification of DID verifiable digital credentials that are compliant with W3C Verifiable Credentials. This project is jointly working with Decentralized Identity Foundation (DIF) (see below) to develop a secure and standard communication based on DIDs — DIDComm – to enable DID Agents interoperability independently of the DLT technology used.

The Decentralised Identity Foundation (DIF) [19] shown in Figure 2-4 is another initiative around decentralised identities aiming to enable interoperability between any DID independently of the DLT used. The following DIF projects are considered:

- **Universal Resolver** [20]: an identifier resolver that works with any decentralised identifier system. It is a server that utilises a collection of DID Drivers to provide a standard means of lookup and resolution for DIDs across implementations and decentralised systems and that returns the DID Document Object (DDO) that encapsulates DPKI (Decentralised Public Key Infrastructure) metadata associated with a DID.
- **Universal Registrar** [21]: an identifier registrar that works with any decentralised identifier system that utilises a collection of DID Drivers to provide a standard means to create, update and deactivate DIDs and DID Documents.
- **DIF Identity Hubs** [22]: a replicated mesh of encrypted personal datastores, composed of cloud and edge instances (like mobile phones, PCs or smart speakers), that facilitate identity data storage and identity interactions. Current reference implementation is in Node.js.
- **DID Communication** [23]: provides 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.

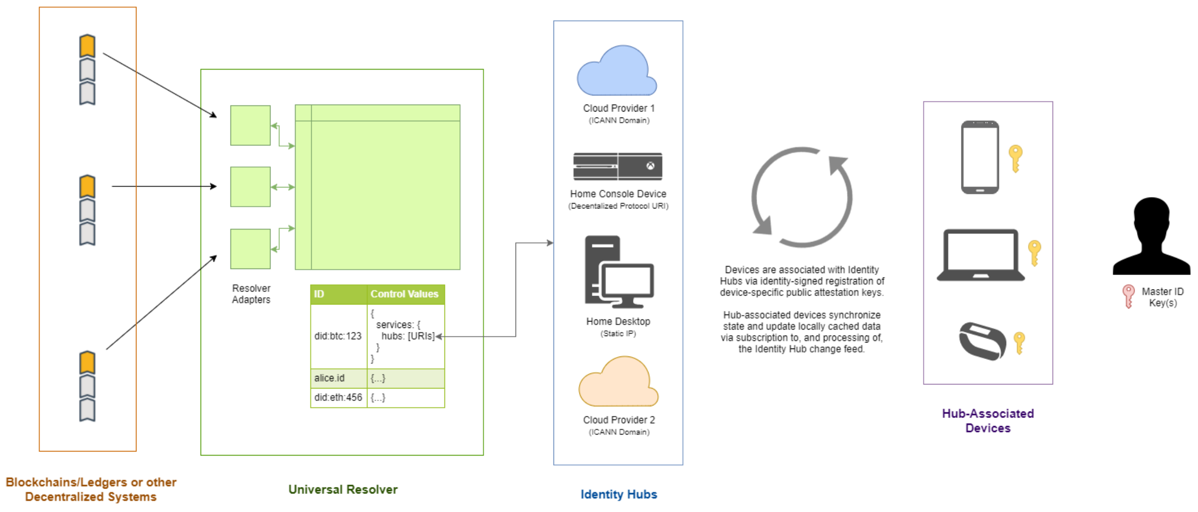
Resolver Adapters
ID
Control Values
did:btc:123
{ services: { hubs: [URIs] } }
alice.id
{...}
did:eth:456
{...}
Cloud Provider 1 (ICANN Domain)
Home Console Device (Decentalized Protocol URI)
Home Desktop (Static IP)
Cloud Provider 2 (ICANN Domain)
Devices are associated with Identity Hubs via identity-signed registration of device-specific public attestation keys.
Hub-associated devices synchronize state and update locally cached data via subscription to, and processing of, the Identity Hub change feed.
Master ID Key(s)
Hub-Associated Devices
Blockchains/Ledgers or other Decentralized Systems
Universal Resolver
Identity Hubs

Figure 2-4: Decentralised Identity Foundation (DIF) projects scope

Additional alternatives were considered at initial project phases, as reported in D3.1 and it was determined to choose Hyperledger Indy as the most widely used distributed ledger to incorporate the registration of W3C DIDs and Verifiable Credentials.

2.1.2.1 *Relevant Entities and Modules*

The most relevant W3C DID related open-source projects, briefly introduced above, were experimented with, taking into account the functional requirements described in D2.4 [1]. According to this experimental evaluation the following components were selected (see Figure 2-5 and Figure 2-6).

The Hyperledger Indy DLT was selected to provide decentralised verifiable DID Registry features including creation and verification of identifiers, verifiable credential schemas, revocation registries and the issuer of public keys. As experimented Hyperledger Indy DLT should provide all required features and no development is expected by 5GZORRO.

The Hyperledger ARIES Cloud Agent Python (ACA-Py) [24] was selected as the main baseline for 5GZORRO Identity and Permission manager (Id&P) features development.

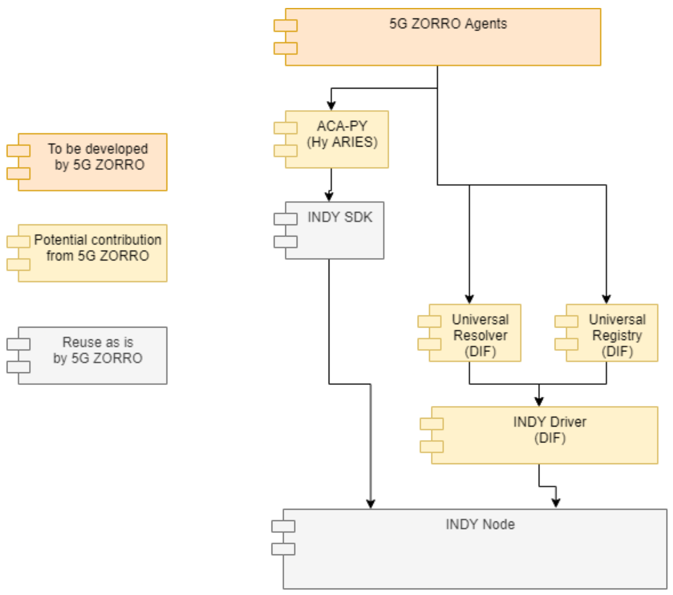

Figure 2-5: Existing DLT Modules to be Leveraged by 5GZORRO Identity and Permissions Manager

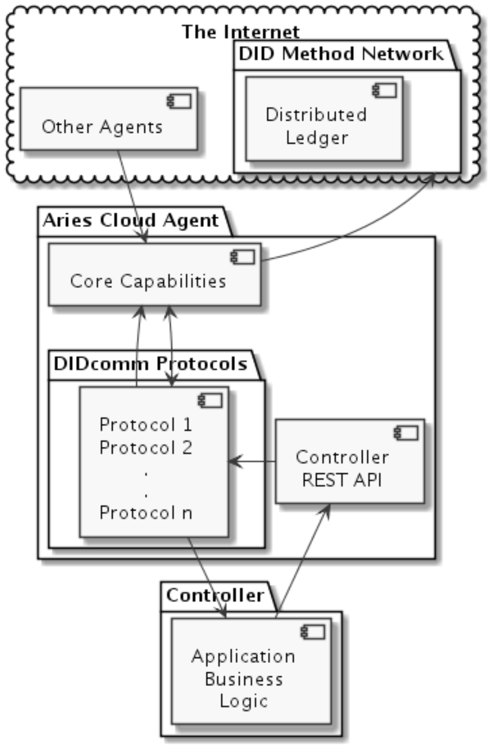

Figure 2-6: Hyperledger ARIES Cloud Agent Python Architecture

According to ACA-Py architecture (see Figure 2-6), the development of 5GZORRO Id&P business logic is implemented as an ACA-Py controller. Such controller registers a webhook with the agent, and the event notifications are HTTP callbacks, and the agent exposes a REST API to the controller for all the administrative messages it is configured to handle. Each of the DIDcomm protocols supported by the agent adds a set of administrative messages for the controller to use in responding to events. The ARIES Cloud Agent includes an OpenAPI (aka Swagger) user interface for a developer to explore the API for a specific agent. The Indy SDK is embedded in the ARIES Cloud Agent and implements the default secure storage. In the current ARIES Cloud Agent implementation, the Indy SDK provides an interface to an Indy-based public ledger for verifiable credential protocols. In future, ledger implementations (including those other than Indy) might be moved into the DIDcomm protocol modules to be included as needed within a configured ARIES Cloud Agent instance based on the DIDcomm protocols used by the agent. 5GZORRO may contribute to ARIES Cloud Agent Python (ACA-Py) development in case new features are required.

2.2 AIOPs and Operational Data Lake Platform

Since 2017, when Gartner has coined the term AIOps to stand for Artificial Intelligence for IT Operations and have given it an official definition [3], AIOps has seen a surge in adoption in various large-scale operational environments such as Cloud environments, distributed storage systems and services, etc. Multiple AIOps products and services have been created for various use-cases but according to Gartner Market Report, all AIOps platforms follow generic principles presented in Figure 2-7 and incorporate three main aspects:

1. Data ingestion and handling (Observe)
2. Machine Learning (ML) analytics (Engage)
3. Remediation (Act)

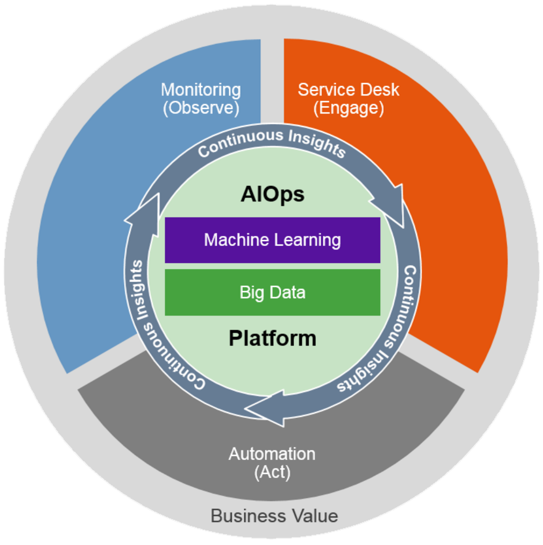

Figure 2-7: AIOps Principles

This view shows that AIOps paradigm is *data-driven* and involves smart and efficient *data collection* (capture, monitoring, telemetry) whereby the data is intelligently governed and stored over time, as well as advanced *data analytics* (statistics, machine learning, artificial intelligence) to provide valuable insights actionable in the context of a particular use case. While most generic AIOps solutions are based on traditional data pipelines with serialized synchronic stages, in 5GZORRO we apply AIOps approach in a context of 5G, adapting data-centric techniques for smart and automated 5G network management in complex multi-player environments.

First, the multi-operator aspect of 5GZORRO use case raises issues around muti-party data collection and sharing. We envision that beyond 5G operational data will come from multitude of devices and services deployed in different geographical, operational, and ownership domains and that not always these players will be willing to share the data openly. On the other hand, providers that offer services composed from lower-level services, to satisfy SLAs contracted with their customers, will need data from underlying devices and services to be incorporated into their automation pipelines. There can be several approaches to resolve this tension. We choose to create Zero-trust environment where each player's data can be stored safely and privately, fully protected, so it can be retrieved and operated on only by parties and for reasons the data owner has agreed to through 5GZORRO multi-party smart contracts.

Second, to facilitate analytics across multiple operators and technology domains, data received from different sources must be enriched, contextualized, and sometimes translated into a unified format. For this, 5GZORRO introduces metadata service component to keep the metadata for cross-referencing and correlation.

Third, multivariate time series measurements, such as network flows, compute, memory, spectrum, and other resource usage, coming from multiple geographies and devices, will inherently loose some of the original temporal context due to lack of clock synchronization between devices, due to communication/processing delays, or due to variances in timestamp encoding etc. On the other hand, many learning algorithms rely on having data prepared in form of fixed-sized time windows. In such cases, there is a need to align and correlate these asynchronous time-series streams coming from multiple subsystems of multiple providers' systems. For this, 5GZORRO data pipeline must normalize the timestamps of all the data sources and aggregate the measurements into elementary time windows, e.g., per minute. These aggregated data blocks are annotated with their time window identity as part of their metadata.

As shown in Figure 2-8, 5GZORRO AIOps architecture allows asynchronous and parallel data processing by decoupled data actors, data providers and data consumers, that act upon logically centralized *5GZORRO Operational Data* repository. Data residing in the repository is described by the Metadata and governed by the Policy components. Data provider agents create new data sets while data consumer agents access existing data sets, process the data, and generate new data sets and/or events, e.g., triggering alerts or invoking orchestration workflows. This architecture is flexible and supports data ingestion from multiple sources, e.g., system's own sensors, external sensors, and vertical feedback sources, as well as dynamic creation and composition of data processing stages, e.g., aggregation, filtering, formatting, and data analytics stages, e.g., contextualizing, model creation, model training, etc.

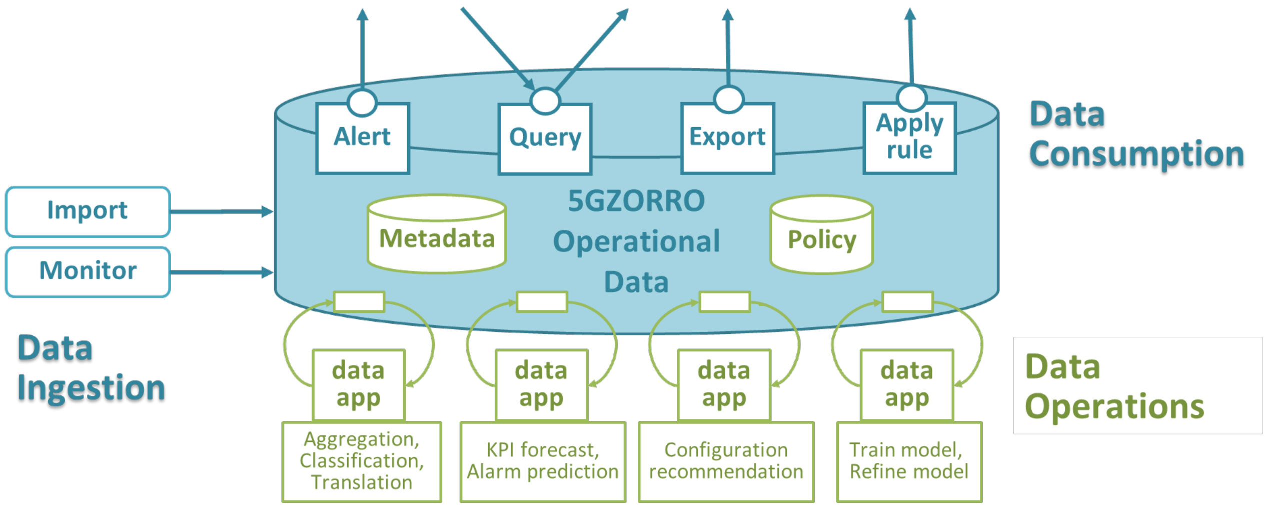

Figure 2-8: 5GZORRO AIOps

5GZORRO Operational Data repository is created roughly following the *Data Lake* concept used to include both the Data Store and the surrounding tools to perform data operations. Many players in the Cloud arena provide Data Lake services, including AWS Lake-Formation [38] , GCP Cloud Storage as Data Lake [39], Azure Data Lake Storage [40], Oracle's Data Lake Solution Patterns for Big Data Use Cases [41], Cloudera Data Platform [42], Zaloni Arena Data Governance Platform [43], Teradata Data Lake Solutions [44], Red Hat Data Services [46] and others. Many of these systems are proprietary and are created for generic use cases involving processing large amounts of data. Open Data Hub (ODH) [46], supported by Red Hat, is an open-source project for running large scale, distributed AI workloads on OpenShift Container Platform. ODH combines multiple existing open-source tools for data storage, distributed AI and ML workflows, messaging, and a Notebook development environment. *5GZORRO Operational Data Lake* is modelled upon the components of ODH, inheriting most of the services, and extending as needed to support 5GZORRO Analytics and Intelligence Services.

2.2.1 5GZORRO Operational Data Lake

As shown in Figure 2-9, 5GZORRO Operational Data Lake is composed of multiple building blocks, described below:

1. **Data Storage** for storing the data; this component should be capable to handle multiple data types, sizes, and can be composed of different subsystems for different data types or to support distributed deployements; 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; must support distributed deployments.
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 such as Apache Hadoop [47] or Apache Spark [48] 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 data sets, and operators that manage data processing workflows.
7. **Runtime** system to host all the components; we choose to use cloud native runtime enriched with advanced capabilities for workflow management, eventing, and serverless execution.

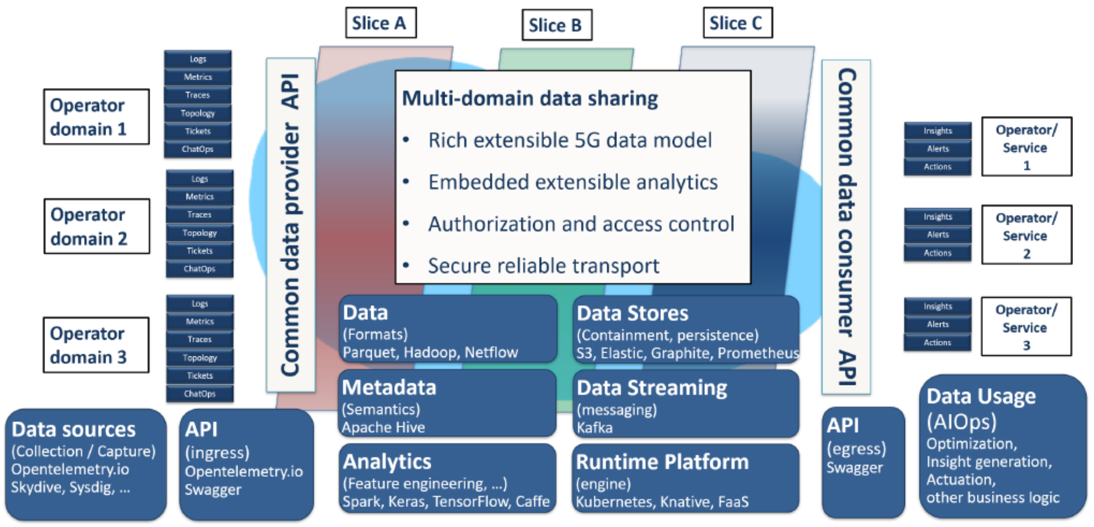
Slice A
Slice B
Slice C
Operator domain 1
Operator domain 2
Operator domain 3
Logs
Metrics
Traces
Topology
Tickets
ChatOps
Common data provider API
Multi-domain data sharing
Rich extensible 5G data model
Embedded extensible analytics
Authorization and access control
Secure reliable transport
Common data 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 2-9: 5GZORRO Operational Data Lake

The following specific choices were made to implement the envisaged 5GZORRO analytics workflows:

For runtime, we employ Kubernetes (k8s) [49], an open-source system for automating deployment, scaling, and management of containerized applications and Argo Workflows [50], an open-source container-native workflow engine for orchestrating parallel jobs on k8s.

For Data Storage, we use Ceph [51], an open-source software storage platform which implements object storage on a single distributed computer cluster. Ceph delivers object, block, and file storage in one unified system.

For messaging and streaming, we use Apache Kafka [52], a distributed streaming platform for publishing and subscribing records as well as storing and processing streams of records.

For data transformation and analytics, we use Apache Hadoop [47], a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models and Apache Spark™ [48], 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. For example, in 5GZORRO all messages to Data Lake must be signed to ensure authenticity of data and to enable authorised access and corrupted data detection. This is accomplished by adding a metadata field that includes an encrypted hash of data content, using the private key of the sender for the encryption and the public key of the claimed sender for verification. In production systems, more sophisticated metadata services will be needed.

For the API service, we expose the URLs of the Data Lake components and provide additional API server endpoint to serve the Data Lake specific APIs described later in this section.

2.2.2 Data Lake Services and Workflows

We refer to the data processing or analytic functionality as a *service*.

Using the 5GZORRO Operational Data Lake to implement 5GZORRO Analytics & Intelligence Services involves defining several basic services, composing them into *workflows*, and deploying these workflows using the Data Lake API. In what follows, we also refer to workflows as *pipelines* and to services composing the workflow as stages to avoid confusion about higher level 5GZORRO services realized on top of the Data Lake, e.g., the Intelligent SLA Monitoring Service. Examples of typical basic Data Lake services:

1. gather data, e.g., collect monitoring data of a computer cluster
2. register the data in a catalogue
3. pre-process the data, e.g., train an anomaly detection model to recognize abnormal behaviour,
4. perform some analysis on the data, e.g., to detect anomalies during production or to generate useful actionable insights
5. perform some action based on the result of the analysis, e.g., invoke orchestration workflow or deploy model at run-time

Some Data Lake services exist out of the box, e.g., data ingestion, while some require custom code to be developed. Out of the box services increase Data Lake usefulness by providing common services (or interfaces) required to ease pipeline composition and deployment.

The information transferred between stages may be the actual data upon which to operate, or it may simply be a notification that the previous processing stage has completed with an indication of where to find the data in the Data Store.

Data Lake workflows are essentially event-driven pipelines implemented with Argo [50]. In cases where event driven invocation is not sufficient, general request-response functions are provided to manage workflows and their stages. This is supported using the underlying k8s mechanisms created to run long-running modules that expose REST interfaces. Generic k8s native service orchestration framework is also used to allow easy connectivity and coordination between services and workflows deployed in a Data Lake. The design rationale is to use K8s to orchestrate services execution. The principles for composing the Data Lake services (stages) into more complex high-level services with workflows are:

- Each service is defined as a workflow using Argo Workflow Custom Resource (CR).

- Each service has a channel (Kafka topic) from which it receives input and a channel (another Kafka topic) to which it sends output. The input/output might be the actual data, or it may include a pointer to the location of the relevant data to be processed.
- The data to be used from the input (Kafka) channel will be specified as input parameters to the Argo workflow. The output of the Argo workflow is sent to the output (Kafka) channel.
- Workflows can communicate with each other via Kafka bus or Argo Events or a combination thereof.
- The output produced by a single component could be consumed by multiple consumers. For example, the output of a data aggregator service may be forwarded both to the data store, to a service that checks for anomalies, and to a service that transormes it for further analytics.
- Output of a service can constitute an event that will trigger another service or workflow. For example, if a service needs to trigger some functionality on the client, it can encapsulate the request in a message delivered via the output channel (Kafka topic) of a service.

The implementation and deployment of the stages of the pipeline are coordinated so that the output from one stage often serves as the input for another stage as shown in Figure 2-10.

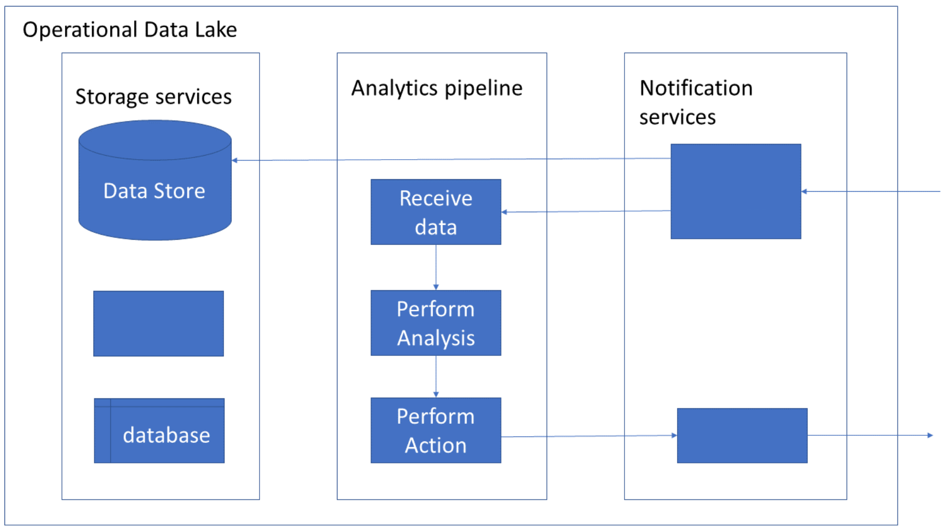

Figure 2-10: Typical Data Lake analytics pipeline setup

2.2.3 Data Lake APIs

The Data Lake inherits the APIs from the various tools that are used in the Data Lake.

- Kubernetes API [53]
- S3 Data Store API [54]
- Apache Kafka API [55]
- Argo Workflows API [56]

For these operations, the Data Lake exposes the URLs of the underlying services (Kubernetes, Kafka, Data Store, etc.) to allow users to derive the most benefit from the Data Lake environment.

An Operator is a user of the Data Lake. In 5GZORRO, both Resource (Infrastructure or Spectrum) Providers and Resource Consumers (e.g., Service Providers) are examples of Operators.

In addition, the Data Lake provides its own APIs to be used by other 5GZORRO components, e.g., methods for registering Data Lake users, for creating and deploying Data Lake workflows (Argo pipelines), for connecting input and output Kafka topics to trigger Data Lake workflows using Function as a Service (FaaS) as shown in the tables below:

Operation name: **registerOperator**			
Description		This API Registers the Operator, verifies that the Operator is allowed to use Data Lake services, defines entities needed to manage Data Lake resources used by the Operator, such as a namespace to be used to identify data stored by the Operator, Kafka topics to be used by the Operator, etc.	
Input Parameters		**Type**	**Description**
	operatorId	string	Unique identifier for Operator that is connecting to the services of the Data Lake.
	authToken	string	Authorization token to authenticate the user.
Output Parameters		**Type**	**Description**
	nameSpace	string	Identifier used to distinguish the resources designated for use by this Operator.
	availableResources	json	A list of resources (e.g., existing pipelines, selected tools, topics) that the Data Lake provides for all registered Operators. Includes list of URLs to allow Operator to access directly Kubernetes, Data Store, Kafka, etc, that are supported on the Data Lake. Includes Data Store bucket into which Operator's data is stored.

Operation name: **unregisterOperator**			
Description		Clean up the resources allocated for the operator. Includes: delete pipelines registered to the operator; delete storage allocated for the operator; remove its namespace, etc.	
Input Parameters		**Type**	**Description**
	operatorId	string	Unique identifier for user.
	authToken	string	Authorization token to authenticate the user.
Output Parameters		**Type**	**Description**
	N/A		

Operation name: **createPipeline**			
Description		Load specified pipeline in the Data Lake runtime environment and enable it to run as a Function as a Service.	
Input Parameters		**Type**	**Description**
	operatorId	string	Unique identifier for user.
	authToken	string	Authorization token to authenticate the user.
	pipelineDefinition	json	An Argo workflow template in json format.
Output Parameters		**Type**	**Description**
	pipelineId	string	Unique identifier of pipeline for user.
	inputTopic	string	A Kafka topic to be used for input to the workflow.
	outputTopic	string	A Kafka topic to be used for output from the workflow to send notifications and/or information to the Operator.

Operation name: **getPipeline**		
Description	Return the information related to specified pipeline.	
Input Parameters	**Type**	**Description**

	operatorId	string	Unique identifier for user.
	authToken	string	Authorization token to authenticate the user.
	pipelineId	json	Identifier of pipeline previously provided by createPipeline.
Output Parameters		**Type**	**Description**
	inputTopic	string	A Kafka topic used for input to the workflow.
	outputTopic	string	A Kafka topic used for output from the workflow to send notifications and/or information to the Operator.
	pipelineDefinition	json	An Argo workflow template in json format of the requested pipeline.

Operation name: **listPipelines**			
Description		Return list of pipelines that have been defined by this Operator and their properties.	
Input Parameters		**Type**	**Description**
	operatorId	string	Unique identifier for user.
	authToken	string	Authorization token to authenticate the user.
Output Parameters		**Type**	**Description**
	pipelines	json	A list of pipelines and their properties in json format.

Operation name: **deletePipeline**			
Description		Unload specified pipeline. Free up the allocated resources, including the Kafka topics that were allocated.	
Input Parameters		**Type**	**Description**
	operatorId	string	Unique identifier for user.
	authToken	string	Authorization token to authenticate the user.
	pipelineId	json	Identifier of pipeline previously provided by createPipeline.
Output Parameters		**Type**	**Description**
	N/A		

Operation name: **createService**			
Description		Load specified service in the Data Lake runtime environment and enable it to run as a REST server.	
Input Parameters		**Type**	**Description**
	operatorId	string	Unique identifier for user.
	authToken	string	Authorization token to authenticate the user.
	containerDefinition	json	A Kubernetes container specification in json format.
Output Parameters		**Type**	**Description**
	serviceId	string	Unique identifier of service for user.
	inputTopic	string	A Kafka topic to be used for input to the service.
	outputTopic	string	A Kafka topic to be used for output from the service to send notifications and/or information to the Operator.
	ports	[]string	Ports that need to be exposed for the service

Operation name: **getService**			
Description		Return the information related to specified pipeline.	
Input Parameters		**Type**	**Description**

	operatorId	string	Unique identifier for user.
	authToken	string	Authorization token to authenticate the user.
	serviceId	json	Identifier of service previously provided by createService.
Output Parameters		**Type**	**Description**
	inputTopic	string	A Kafka topic used for input to the service.
	outputTopic	string	A Kafka topic used for output from the service to send notifications and/or information to the Operator.
	containerDefinition	json	A Kubernetes container template in json format of the requested service.

Operation name: **listServices**			
Description		Return list of services that have been defined by this Operator and their properties.	
Input Parameters		**Type**	**Description**
	operatorId	string	Unique identifier for user.
	authToken	string	Authorization token to authenticate the user.
Output Parameters		**Type**	**Description**
	services	json	A list of services and their properties in json format.

Operation name: **deleteService**			
Description		Unload specified pipeline. Free up the allocated resources, including the Kafka topics that were allocated.	
Input Parameters		**Type**	**Description**
	operatorId	string	Unique identifier for user.
	authToken	string	Authorization token to authenticate the user.
	serviceId	json	Identifier of service previously provided by createService.
Output Parameters		**Type**	**Description**
	N/A		

As stated above, several default services and workflows are provided by the Data Lake out of the box. Some workflows are internal. For example, upon calling *registerOperator()*, an internal workflow is executed and a list of *availableResources* that this newly registered Data Lake user may exploit is returned. Some workflows are exposed by the Data Lake to its users, e.g., is the *resourceMetricsIngestPipeline* workflow that provides a Kafka topic on which the metrics data may be ingested to the datalake and then consumed by other services. Data sent to this workflow is stored in the object storage and is indexed for easy lookup by the *datalake catalog* service. This allows identifying metrics data stored based on various identifiers (producrID, resourceID, etc). Later, the data is made available by a *stream data* service that has been implemented to retrieve and stream the requested metrics data stored in the Data Lake.

2.2.4 Data Stored in the Data Lake

The 5GZORRO Operational Data Lake is built as a platform for handling all the operational data arising as part of all the 5GZORRO platform use cases, those realized during the project lifetime and those that might appear at later exploitation stages. The need to be generic (in scope of the 5G technology domain) required the Data Lake platform to be agnostic of the different data types and formats that will be stored and processed in it and allow various data, both structured and unstructured and provide general services to store and handle any type of data.

Specific 5GZORRO services that rely on the Data Lake, have identified specific data typed and formats for various purposes, e.g., for AI-ML tasks in predicting SLA breaches or for network optimization. A comprehensive description of the application scopes of this data to AI/ML services in 5G Networks can be found in [35], with direct contributions from 5GZORRO Consortium.

The data types and formats descriptions are included in deliverables describing these specific 5GZORRO Services, e.g., in Section 3 of this document and in D4.4 [3]. In addition, detailed information models related to the Data Lake platform can be found in Section 4.4 and detailed information models related to 5GZORRO operational data in Section 4.4 of this document.

2.2.5 Data Lake example

This section is inherited from D3.1 and provides a generic example of how one can use the 5GZORRO Operational Data Lake platform, its APIs and its out of the box services, to create higher level data driven analytics services.

The example is based on a simplified version of the SLA violation detection use case and consists of the following basic stages:

1. provide monitoring data
2. index the data for easy lookup
3. perform analytics to detect SLA violation
4. upon detection, invoke specified action, e.g., generate an alert

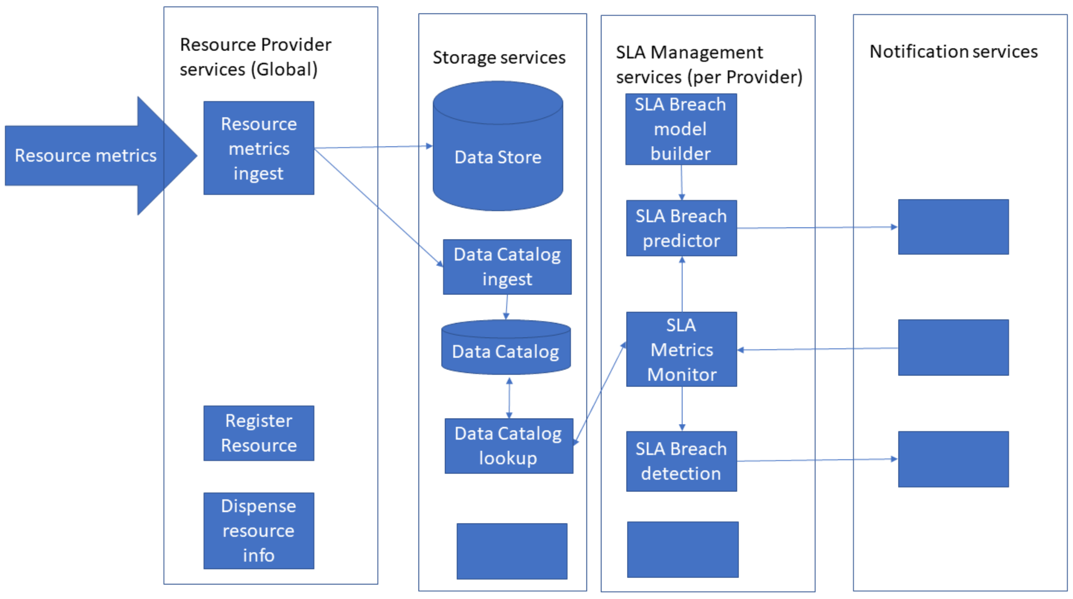

Figure 2-11: Resource Monitoring and SLA breach detection/prediction

Even this simple example involves two asynchrionious event-driven pipelines, one to collect, aggregate, and feed the data and the other is to consume and analyse it. Data Lake provides all the required functionality to define and deploy these pipelines and to arrange the required interactions between them as depicted in Figure 2-11. All the incoming resource metrics must be processed to identify their specific metadata and to correlate them to a specific SLAs affected by this resource. The SLA monitoring workflow then can retrieve all the metrics relevant to the monitored SLA, apply its analytics, e.g., anomaly detection, to detect SLA violations, and act upon the detected violation, e.g., by generating an alert or by invoking additional workflow, in the Data Lake or out of it, e.g., the orchestration workflow in Intelligent Slice and Service Manager (ISSM). This is accomplished with a combination of data-driven events (using Kafka topics) and other specialized (REST) interfaces, as needed.

The portion of the pipeline of detecting anomalies during production is depicted in Figure 2-12. Monitoring data is provided regularly and is aggregated. When the anomaly detector analytics module detects an anomaly, it invokes some function to react to the anomaly.

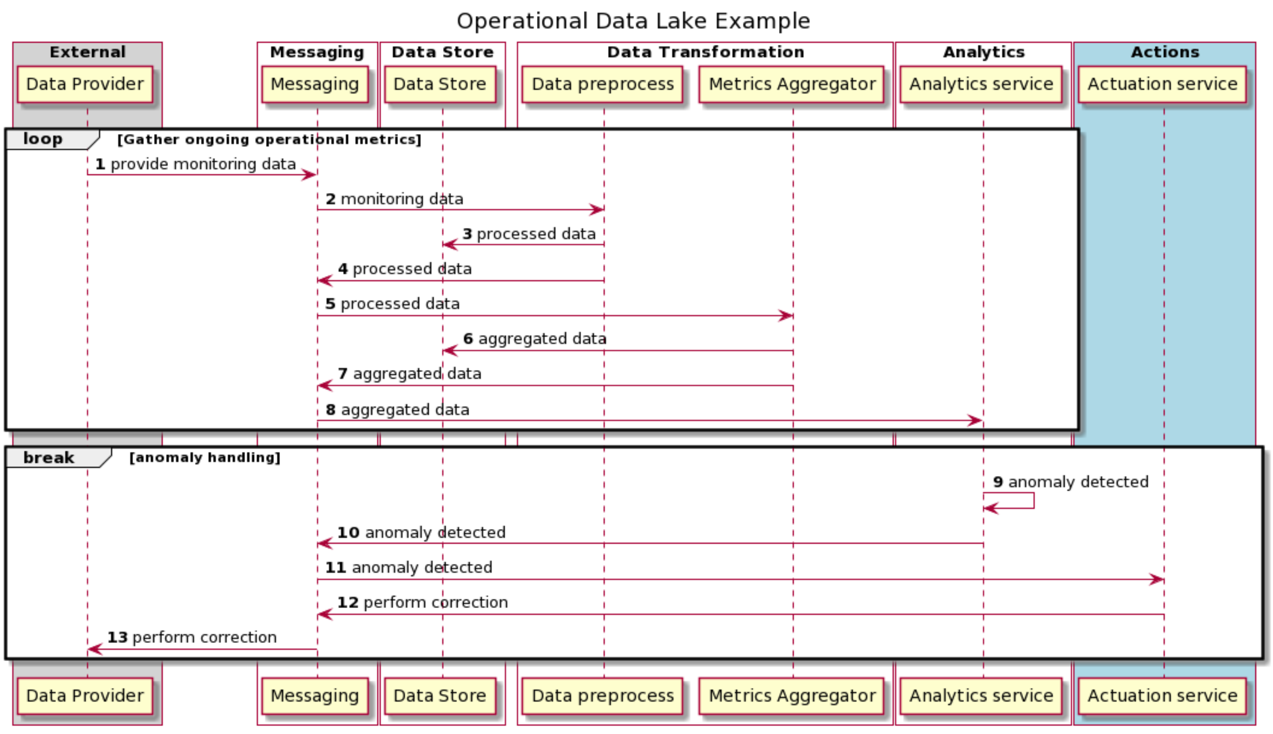

Figure 2-12: Operational Data Lake example for SLA Monitoring

3 Evolved 5G Service Layer Solutions

This section is devoted to describing the design of the 5G Service Layer Solutions that compose three of the four 5GZORRO Logical Subsytems of the 5GZORRO Platform architecture presented in Figure 1-1 – the Security and Trust Logical Subsytem, the Marketplace/Business Logical Subsytem, and the Analytics and Intelligence Logical Subsytem. Referring to 5GZORRO Software Platform described in D2.4 [1], these three Logical Subsystems correspond to three main software platform components, the Governance Platfform, the Marketplace Platform, and the Cross-domain Analytics & Intelligence for AIOps Platform, as emphasised in Figure 3-1 below:

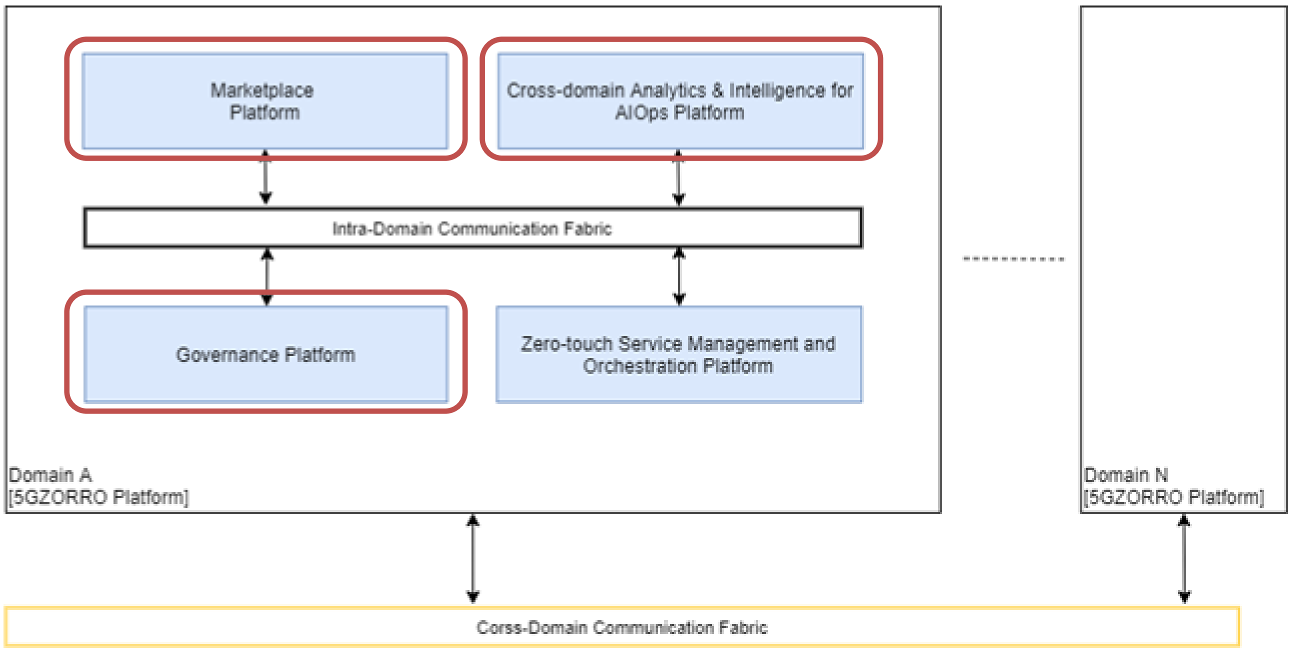
Marketplace Platform
Cross-domain Analytics & Intelligence for AIOps Platform
Intra-Domain Communication Fabric
Governance Platform
Zero-touch Service Management and Orchestration Platform
Domain A [5GZORRO Platform]
Domain N [5GZORRO Platform]
Corss-Domain Communication Fabric

Figure 3-1: Evolved 5G Service Layer Solutions as part of the 5GZORRO Software Platform

Services that compose these software platforms are described in three subsections that follow: Section 3.1 for the Governance Platform Services, Section 3.2 for the Marketplace Platform Services, and Section 3.3 for the Cross-domain Analytics & Intelligence Platform Services. All the involved information elements are then described in the next section, Section 4.

3.1 Governance Platform Services

As described in D2.4 [1], 5GZORRO Governance Platform (see Figure 3-1) is responsible for defining, validating, and operating the identities, the certificates, and the permissions for all 5GZORRO stakeholders as well as for all the the entities defined and traded in 5GZORRO Marketplace. The Governance Platform features the decentralized management of global (cross-domain) identifiers (e.g., stakeholder identifiers and 5GZORRO resource identifiers) according to Self-sovereign Identity principles and by leveraging DLT technologies. It supports creation, verification, and revocation of certificates as well as authentication and authorisation of identities across all 5GZORRO domains.

5GZORRO Governance platform is realised as a set of software modules that provide core governance services, either to 5GZORRO stakeholders or to other 5GZORRO platforms. These services are presented in Figure 3-2 and covered in the following subsections: the DLT Governance Manager in Section 3.1.1, the

Identity and Permissions Manager in Section 3.1.2, the Legal Prose Manager in Section 3.1.3, and the Governance Portal in Section 3.1.4.

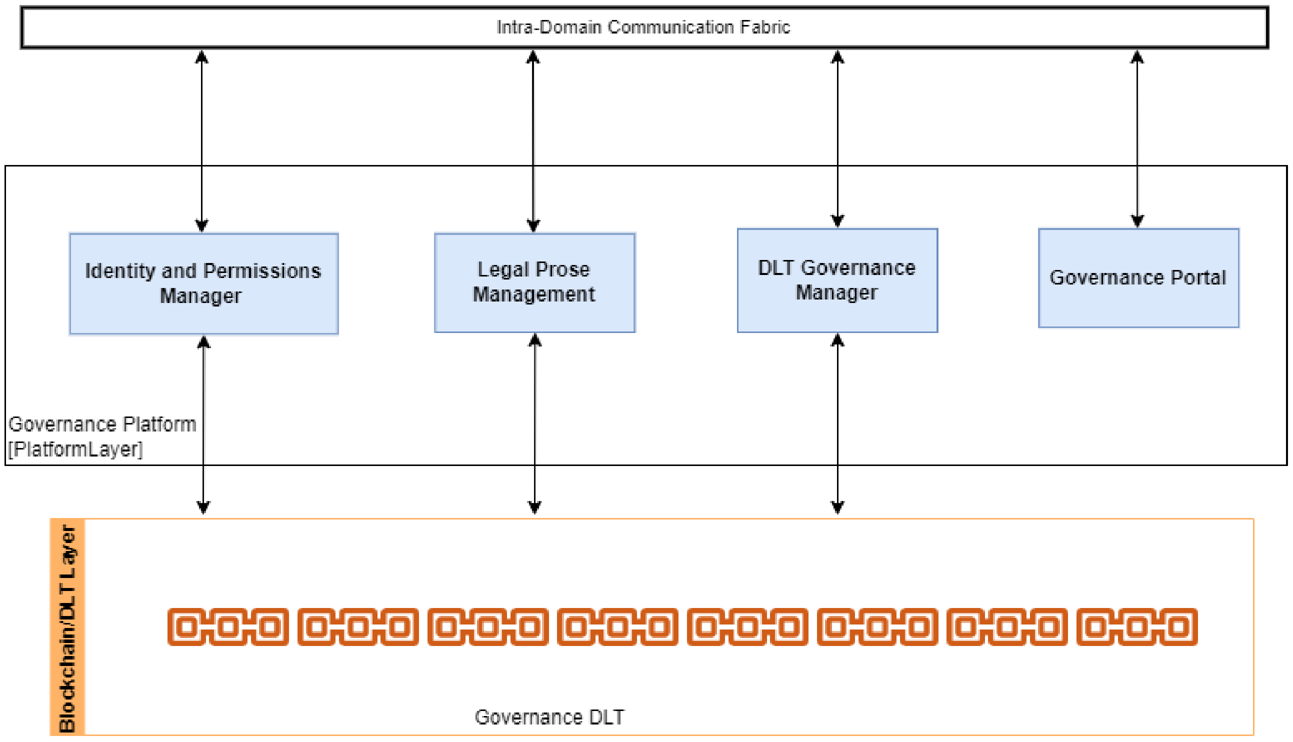
Intra-Domain Communication Fabric
Identity and Permissions Manager
Legal Prose Management
DLT Governance Manager
Governance Portal
Governance Platform [PlatformLayer]
Blockchain/DLT Layer
Governance DLT

Figure 3-2: Governance Platform Services

3.1.1 DLT Governance Manager

Central to the 5GZORRO Governance Platform, the Governance Manager module is responsible for facilitating and coordinating marketplace governance in a decentralised manner. Administrators of the platform (Marketplace Governance Board) are responsible for partaking in reviewing and voting on proposals subject to governance. This decentralized process will be underpinned by the Governance DLT in order to ensure full transparency and auditability. The voting permission and requirement is dictated by the presence of an admin stakeholder in the Governance Board DID document and associated Verifiable Credential.

Since governance is not the highest priority in terms of 5GZORRO functionalities, a simplified governance framework will be implemented for a defined set of scenarios and a simplified model e.g., all admins approve.

3.1.1.1 *5GZORRO Specific Enhancements*

By leveraging DLT to underpin the marketplace governance we see the decentralized management of the platform with greater transparency and auditability. Table 3-1 lists all the processes and all the entities subject to 5GZORRO governance.

Governance proposals such as a new Stakeholder registration or new legal prose template will be identified by a generated DID and the lifecycle events (votes and approve/reject) pertaining to the DID captured through the issuance of verifiable claims.

Various legal requirements apply across Europe in relation to 'fit and proper' persons/entities owning spectrum. It is standard practice for due diligence to take place to qualify persons/entities applying for spectrum rights. In the case of 5GZORRO and to facilitate the real-time approach, such qualification will take place during the on-boarding process (see Section 3.1.2.4.1 where this workflow is initiated). Furthermore, approval of a membership request from a party wishing to offer spectrum will be at the discretion of the regulator.

Table 3-1: 5GZORRO Entities Subject to Governance

Entity	Governance Processes	Description
Stakeholder	Registration	Ability for each stakeholder to apply, stating their intended role and capabilities
Service Level Agreement Breach	Dispute	Ability for a provider to initiate a dispute process; supplying appropriate evidence to support the claim. This should result in a consortium decision that is either upheld or rejected.
Legal Prose / License Template	Propose New / Propose Archive	Each legal prose template should be subject to a governance process before being approved for use in the marketplace. By doing so we can be sure that the prose and model of contracts has been subject to scrutiny and marketplace stakeholders can have confidence in their utilisation

Where it is not practical for a regulator to approve spectrum resource trading in real-time, the 5GZORRO platform will also afford oversight of all Spectrum trades through the Marketplace Portal. As such, if on inspection the regulator deems a trade improper e.g., the consumer should not legitimately be able to hold the spectrum, then the regulator will also have the capability to terminate the agreement.

3.1.1.2 *Governance Manager Design Details*

The Governance Manager plays a key role in the attainment of the following KPIs:

- *Provide mechanisms for zero touch trust automation in multi-domain scenarios on top of a 5G service management framework (KPI target: to cover up to 4 different stakeholders as part of the automated trust establishment process and to enable its automatic renegotiation when a stakeholder is joining or leaving the trust link).*
- *The approval mechanism to be a resource provider in 5GZORRO **MUST** be handled according to 5GZORRO decentralized governance model (distributed consensus). A governance model that considers at least 2 admin stakeholders and 1 regulator stakeholder with the power to veto issuance of spectrum rights should be demonstrated)*

In order to deliver the afore-mentioned capabilities the internal architecture of the Governance Manager module is shown below, comprising the following main entities presented in Figure 3-3:

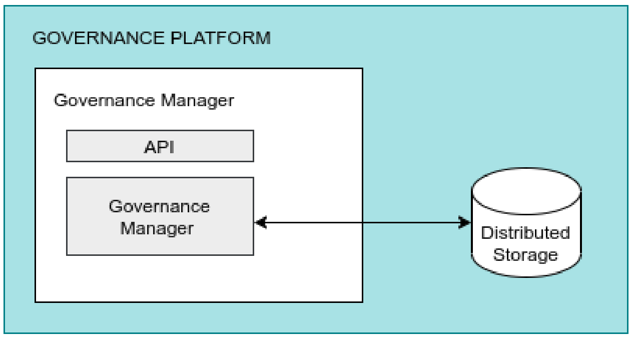

Figure 3-3: Governance Manager Module Architecture

- **API**: a set of APIs implemented to expose the module capabilities regarding the proposal of a particular action subject to governance, and interfaces to support the subsequent voting and issuance of a decision
- **Governance Manager**: Functional entity that implements the logic of the Governance Manager, interacting with the Governance DID Agent for the purposes of issuing the necessary DIDs and VCs that capture the definition and status of governance decisions in a verifiable manner. Distributed storage is leveraged so that admin stakeholders can keep a synchronized view of membership and proposal status and support query capabilities.

3.1.1.3 *Governance Manager Workflow*

The process of proposal, voting and decision is largely the same for each governance process considered for 5GZORRO and the steps are illustrated in Figure 3-4 below.

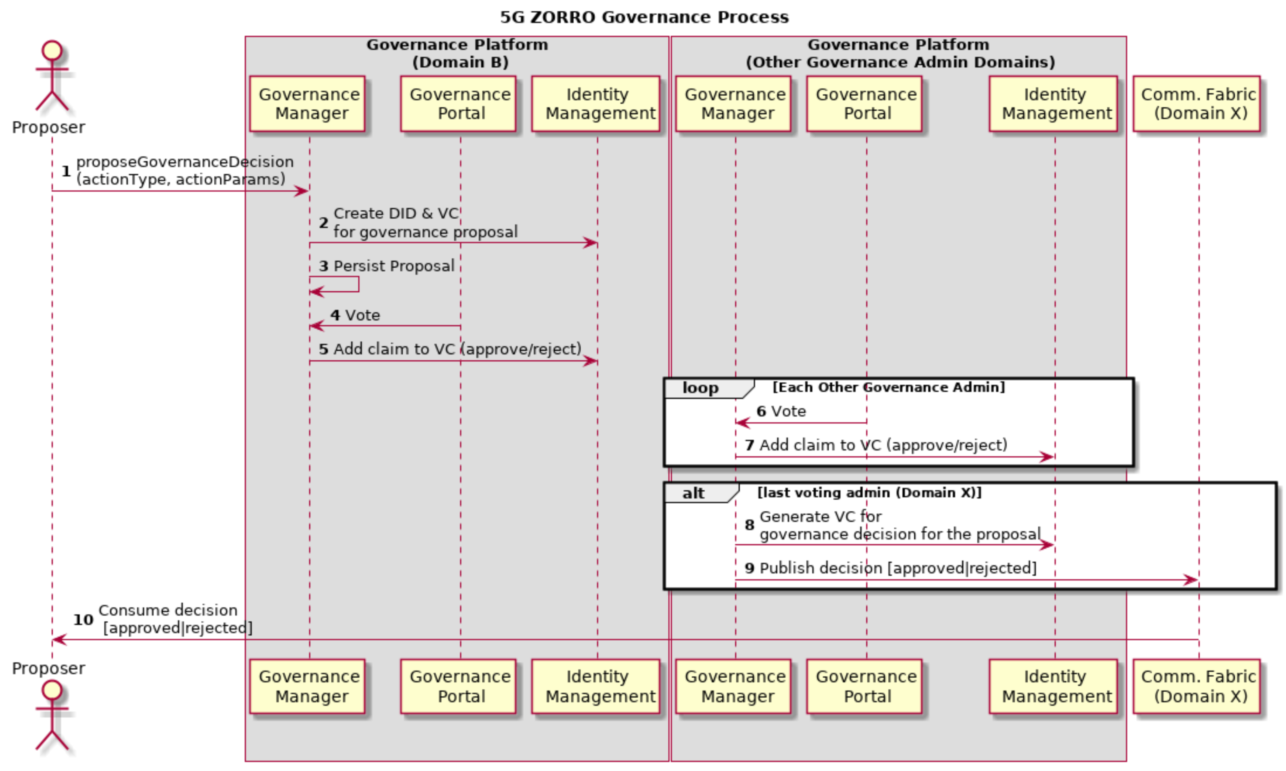
5G ZORRO Governance Process
Governance Platform (Domain B)
Governance Platform (Other Governance Admin Domains)
Proposer
Governance Manager
Governance Portal
Identity Management
Governance Manager
Governance Portal
Identity Management
Comm. Fabric (Domain X)
1 proposeGovernanceDecision (actionType, actionParams)
2 Create DID & VC for governance proposal
3 Persist Proposal
4 Vote
5 Add claim to VC (approve/reject)
loop [Each Other Governance Admin]
6 Vote
7 Add claim to VC (approve/reject)
alt [last voting admin (Domain X)]
8 Generate VC for governance decision for the proposal
9 Publish decision [approved|rejected]
10 Consume decision [approved|rejected]

Figure 3-4: Governance Workflow

Step 1: a proposal is made by an entity to the governance manager deployed in the domain of a stakeholder on the Governance board. The discovery of this service is achieved through accessing the DID doc of the Governance Board DID and resolving an endpoint identified in there.

Step 2-3: a DID is created for global identification of the proposal and the proposal stored in distributed storage with a status of PROPOSED.

Step 4-5: the stakeholder reviews the evidence associated with the proposal through the governance portal and votes (approve/reject) issuing a VC for the proposal DID to capture their decision.

Step 6-7: Likewise, each other member of the governance board will review and submit their vote via the Governance Portal, again issuing a VC to reflect their decision.

Step 8: the final voting admin will also generate a VC to reflect the outcome based on the votes cast and in-line with the adopted governance model.

Step 9-10: The decision/outcome is then published for the requesting entity to process accordingly.

3.1.1.4 *Governance Manager APIs*

Operation name	**checkMembershipStatus**	
Description	API endpoint to allow a platform user to check the membership status of a particular stakeholder	
Input Parameters		
Name	**Type**	**Description**
stakeholderDID	String	DID that uniquely identify the legal entity to be checked
Output Parameters		
MembershipStatus	MembershipStatus	Status information regarding the progress of a membership request or more generally the current status of the DID owning stakeholder

Operation name	**getMembers**	
Description	API endpoint to retrieve a list of active Marketplace members	
Input Parameters		
Name	**Type**	**Description**
None		
Output Parameters		
List<Member>	Member	A list of members and their current status

Operation name	**proposeGovernanceDecision**	
Description	API endpoint to propose something to be decided according to the governance model. Essentially the creation of a DID to track/identify the proposal and associated status	
Input Parameters		
Name	**Type**	**Description**
actionType *actionType*	ActionTypeEnum	Enum that captures the intended type of proposal OnboardStakeholder|SlaDispute|NewLegalProseTemplate|ArchiveLegalProseTemplate
actionParams	ActionParams	Parameters needed to settle a particular action type
Output Parameters		
proposalIdentifier	String	DID that uniquely identify the proposal

Operation name	**getProposals**	
Description	API endpoint to retrieve all submitted governance proposals	
Input Parameters		
Name	**Type**	**Description**
statusFilter	Optional< ProposalStatusEnum>	Optional filter to filter proposals of a particular status
actionTypeFilter	Optional<ActionTypeEnum>	Optional filter to filter proposals of a particular type

Output Parameters		
proposalDetails	GovernanceProposalStatus	Object containing all pertinent proposal information such as affected stakeholders and status

Operation name	**getGovernanceDecision**	
Description	API endpoint to retrieve a governance decision previously proposed and its current status	
Input Parameters		
Name	**Type**	**Description**
proposalIdentifier	String	A DID that uniquely identifies a proposal
Output Parameters		
proposalDetails	GovernanceProposal	Object containing all pertinent proposal information such as affected stakeholders and status

Operation name	**voteGovernanceDecision**	
Description	API endpoint to vote on a governance decision proposed in proposeGovernanceDecision()	
Input Parameters		
Name	**Type**	**Description**
proposalIdentifier	String	A DID that uniquely identifies a proposal
accept	Boolean	Boolean to indicate an accept/reject decision
Output Parameters		
None		

3.1.2 Identity and Permissions Manager

The goal of Identity Management and Permissions Management is to supply the mechanisms required for generating unique identifiers in the 5GZORRO ecosystem, recognising communicating endpoints, identifying and authenticating entities, services and organizations, and authorising consumer requests to access permissioned services and resources.

Identity Management can identify providers, consumers, services, resources, organizations, etc., using DIDs associated with a Verifiable Credential, enabling authentication. In the case of Permissions Management, this allows setting up a secure layer that regulates the access to resources, services, and delimited areas using a set of policies and rules. By means of policies and rules, each domain can determine the amount of information exposed, the duration for which that information is shared, what kind of information is shared, limiting resource capabilities, and so on. Therefore, each domain must define its policies and rules based on its criteria such as improving security, usability, availability, and cost-efficiency. In the end, Permissions Management attempts to prevent unauthorised access to services, resources, and data, making access control enforcement as granular as possible.

3.1.2.1 *Identity and Permissions Deployable Components*

The DID Agent component is the core deployable component of 5GZORRO Identity and Permissions Manager and it follows the main principles of DIF Cloud Agent design [19]. Each DID Agent holds a DLT Wallet according to the DLT technology used, including secured storage to handle private keys. At this stage, there are envisaged three main types of DID Agent components (see Figure 3-5):

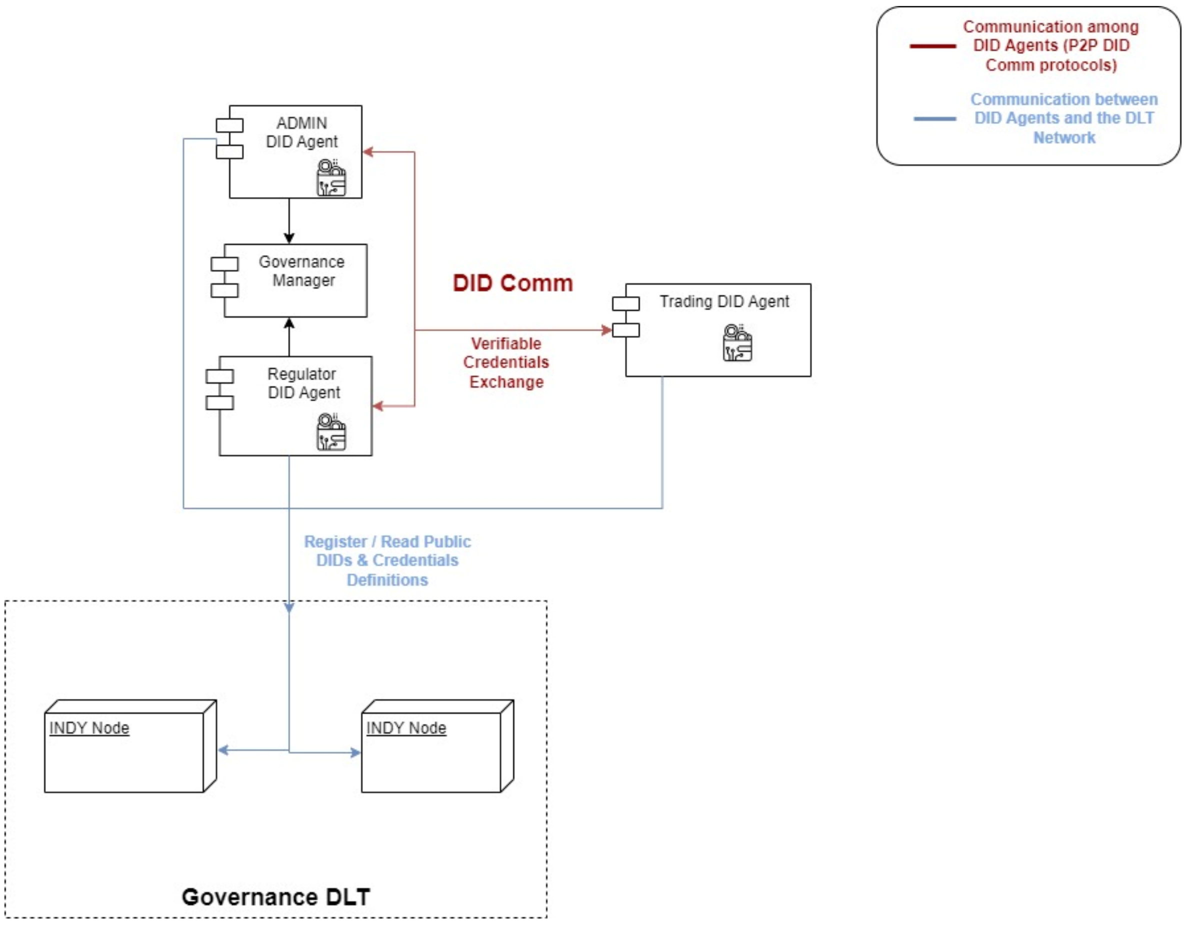
Communication among DID Agents (P2P DID Comm protocols)
Communication between DID Agents and the DLT Network
ADMIN DID Agent
Governance Manager
Regulator DID Agent
DID Comm
Verifiable Credentials Exchange
Trading DID Agent
Register / Read Public DIDs & Credentials Definitions
INDY Node
INDY Node
Governance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

Figure 3-5: Identity and Permissions Manager Deployable Components and their interfaces

- **Admin (or Governance) DID Agent**: it provides specific business logic to be used to issue non-regulated Offers VCs on Providers request and to manage Governance Verifiable Credentials including credentials about 5GZORRO stakeholders Marketplace membership. Admin DID Agents have permissions to write in the Governance DLT.
- **Regulator DID Agent**: it extends the Admin DID Agent to provide additional business logic required to issue Verifiable Credentials for 5G regulated resources notably 5G Spectrum resources. As an extension of Admin DID Agents, Regulator DID Agents also have permissions to write in the Governance DLT.
- **Trading DID Agent**: it provides specific business logic to request & manage Verifiable Credentials 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 i.e., Admin DID Agents. This DID Agent may play two roles:
    1. **Provider Trading Agent**: it plays the role of the Verifiable Credential Offer Provider
    2. **Consumer Trading Agent**: it plays the role of the Verifiable Credential Offer Consumer

DID Agents communicate among each other on top of P2P DID Comm protocols [23] to exchange Verifiable Credentials or its presentations.

The communication between DID Agents and the DLT should be done via the DIF Universal Registrar and Resolver in order to be as agnostic as possible of the DLT technology used. This interface is mainly used to register and read 5GZORRO Public DIDs (i.e., Stakeholder DIDs and the Governance Board DID) and associated DID Documents. For 5GZORRO implementation, the Hyperledger INDY Network Node has been selected.

The Admin DID Agent as well as the Regulator DID Agent should require an interface with the Governance Manager component in order to apply the Governance Model adopted for 5GZORRO Marketplace.

3.1.2.2 *Identity and Permissions Deployment Scenarios*

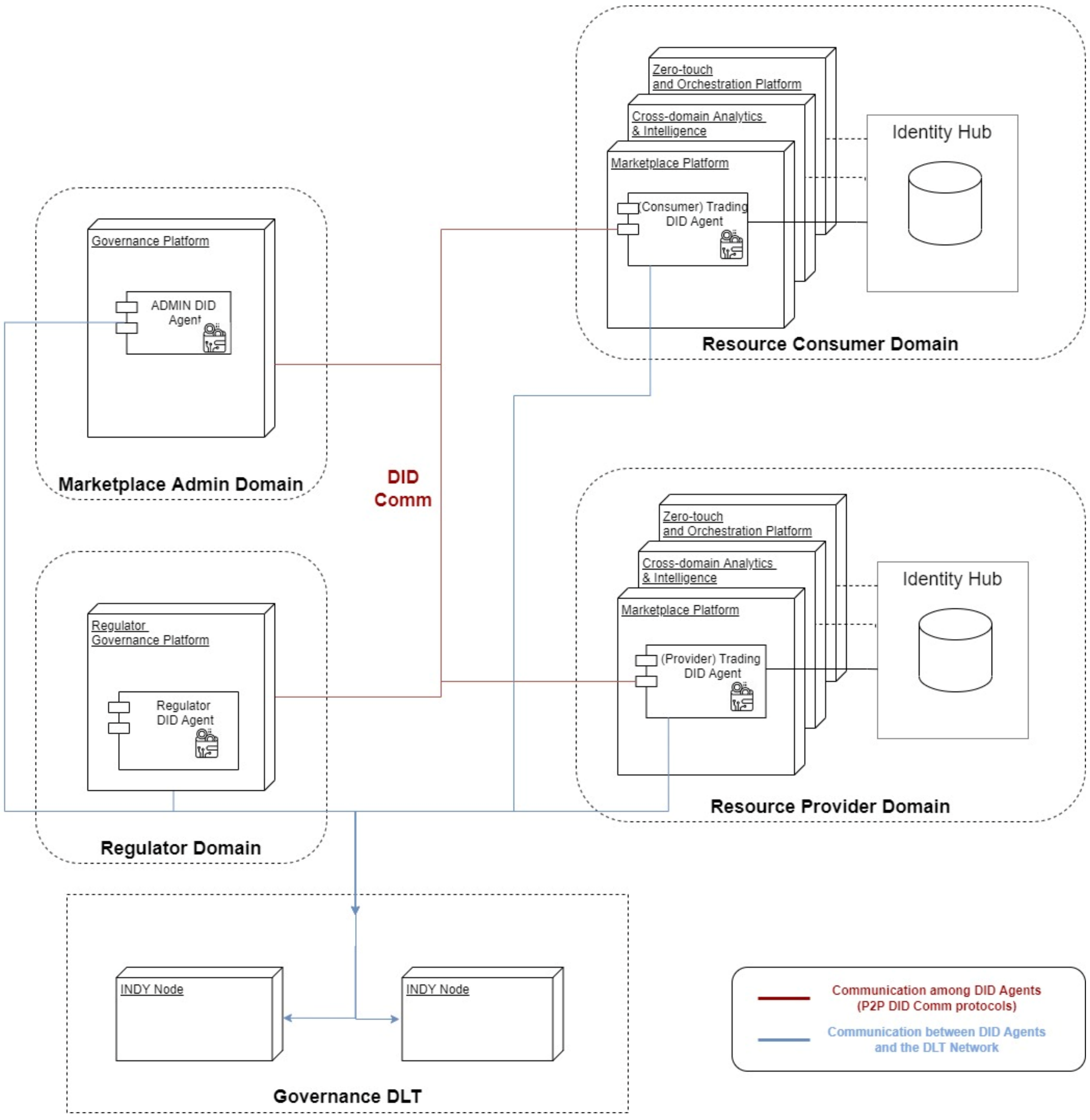
Zero-touch and Orchestration Platform
Cross-domain Analytics & Intelligence
Marketplace Platform
(Consumer) Trading DID Agent
Identity Hub
Resource Consumer Domain
Governance Platform
ADMIN DID Agent
Marketplace Admin Domain
DID Comm
Regulator Governance Platform
Regulator DID Agent
Regulator Domain
Zero-touch and Orchestration Platform
Cross-domain Analytics & Intelligence
Marketplace Platform
(Provider) Trading DID Agent
Identity Hub
Resource Provider Domain
INDY Node
INDY Node
Governance DLT
Communication among DID Agents (P2P DID Comm protocols)
Communication between DID Agents and the DLT Network

Figure 3-6: Identity and Permission Agents Deployment Scenario

Identity and Permissions Manager DID Agents will be deployed in different 5GZORRO platforms according to their types and the scenarios to be supported. Figure 3-6 depicts a possible deployment topology to support the offering and trading of a Regulated Resource in 5GZORRO Marketplace, notably a Spectrum type of Resource, where:

- The Admin DID Agent is deployed in the Governance platform and will be used to issue non-regulated Offers VCs on Providers' request. This Agent also manages any Governance Verifiable Credentials including the ones related with Spectrum Offer SLA violations.

- The Regulator DID Agent is an extension of the Admin DID Agent regulated resources. Thus, it is also deployed in the Governance platform operated by the Regulator to issue the Spectrum VC that will be transferred to the Resource Provider with the rights to offer the Spectrum resource in the Marketplace.
- The (Provider) Trading DID Agent is operated by the Spectrum provider and will be the holder of the Spectrum VC issued by the regulator. The (Provider) Trading DID Agent can also be deployed in the Cross-domain Analytics & Intelligence platform and in the Zero-touch and Orchestration platform as soon as an Identity Hub [22] is deployed. But for simplification purposes, each domain should only have one Trading DID Agent installed. In this case, any 5GZORRO component can interface with the Trading DID Agent by using its API (see Section 3.1.2.5).
- The (Consumer) Trading DID Agent is operated by the Spectrum consumer and will confirm the Provider has the rights over the offered Spectrum by verifying the Spectrum VC issued by the regulator. As previously mentioned, the (consumer) Trading DID Agent can also be deployed in the other two 5GZORRO platforms (Cross-domain Analytics & Intelligence platform and in the Zero-touch and Orchestration platform).

Other 5GZORRO scenarios, including the ones not involving regulated resources, can be supported in similar deployment environments.

3.1.2.3 *Identity and Permissions Manager Design Details*

The Identity and Permissions manager component is developed on top of Hyperledger ARIES Cloud Agent Python (ACA-Py) Controller OpenAPI as an ARIES Cloud Agent Controller containing the Business Logic required by 5GZORRO DID Agents. The common Business Logic of 5GZORRO DID Agents is provided by the DID Agent Core module that is reused by each 5GZORRO DID Agent as depicted in the Figure 3-7.

The ACA-Py framework is already designed to be DLT agnostic by deploying a pluggable Wallet for each DID DLT Network. However, this is not scalable, and it won't be possible to support all DID DLT networks in every Agent. To partially solve this problem, it was envisaged to use the DIF universal DID Resolver to read public DIDs. This integration could be done at ACA-Py level or at 5GZORRO Core Agent Controller level. However, the mechanism to write data to the ledger must be implemented within the Aries agent to ensure full control over the private keys involved in the transactions. Thus, at this stage, the usage of DIF Universal Registrar is not considered. A possible solution is to apply the protocol-on-the-fly [25] concept as introduced by reTHINK project [26].

At this point, the DID Comm protocols currently supported by ACA-Py should be able to support the exchange of 5GZORRO Credentials among the different DID Agents, including:

- The **connection protocol** (ARIES RFC 0160) [27] enables two agents to establish a connection through a series of messages—an invitation, a connection request, and a connection response.
- The **issue credential protocol** (ARIES RFC 0453) [28] allows an agent to issue a credential to another agent.
- The **present proof protocol** (ARIES RFC 0454) [29] enables an agent to request and receive a proof from another agent.

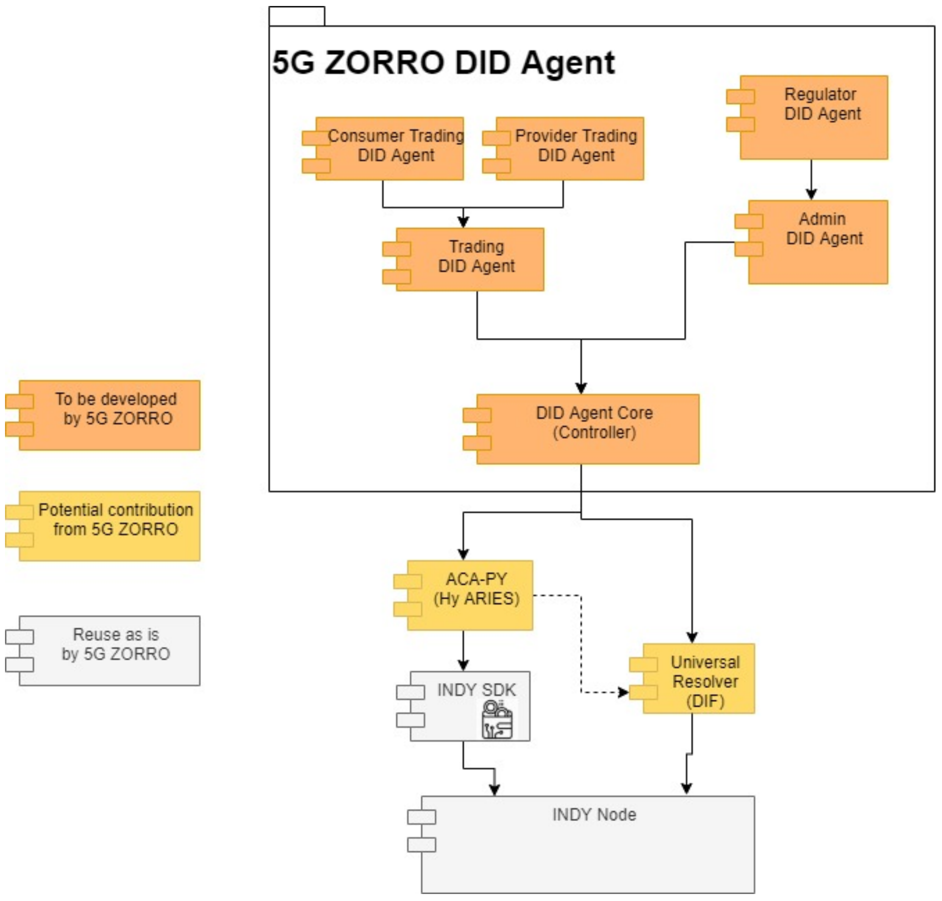

Figure 3-7: DID Agent Design

Nevertheless, if needed, ACA-Py pluggable protocols design should allow the usage of any potential 5GZORRO specific protocol.

The APIs implemented by 5GZORRO Agents are depicted in the Figure 3-8 below and they are described in Section 3.1.2.5.

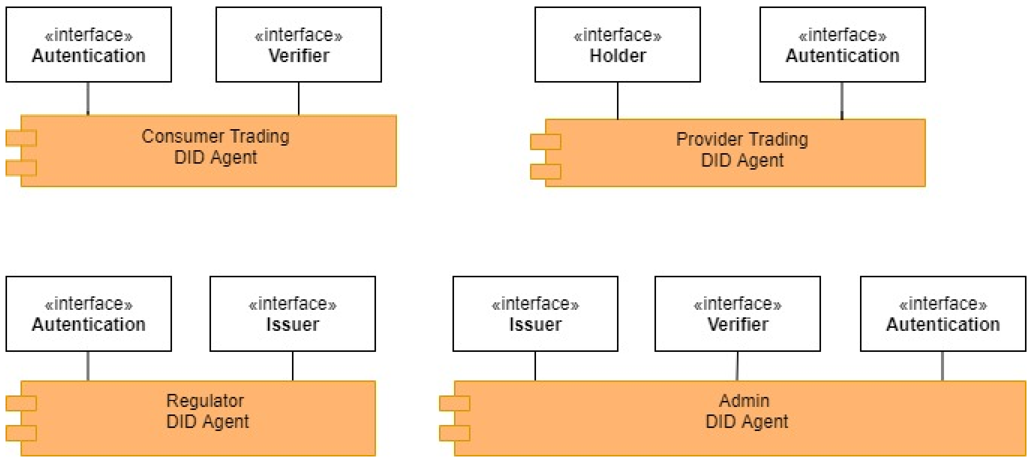
«interface» Autentication
«interface» Verifier
Consumer Trading DID Agent
«interface» Holder
«interface» Autentication
Provider Trading DID Agent
«interface» Autentication
«interface» Issuer
Regulator DID Agent
«interface» Issuer
«interface» Verifier
«interface» Autentication
Admin DID Agent

Figure 3-8: APIs implemented by 5GZORRO Agents

The Core DID Agent provides the following major features:

- Client to interact with ACA-Py Controller REST API

- Core Bootstrap Logic
- Business Logic to support the StakeholderVC Holder features
- Business Logic to support the GovernanceVC Verifier features

The Trading DID Agent provides the following major features:

- Bootstrap Logic for non-Administrators:
    - Public DID is endorsed by an Admin Agent i.e., no write permission on the ledger
    - DID Comm Connection Setup with all Admin Agents
- Business Logic to support the AgreementVC Holder features

The Provider Trading DID Agent provides the following major features:

- Business Logic to support the LicenseVC Holder features

The Consumer Trading DID Agent provides the following major features:

- Business Logic to support the LicenseVC Verifier features
- Business Logic to support the StakeholderVC Verifier features

The Admin DID Agent provides the following major features:

- Bootstrap Logic for Administrators:
    - Create a trustee endorser DID on the ledger that has full write permission on the ledger
    - Business logic to handle the required claims to register as a member of the Marketplace Governance Board
- Business Logic to support all required VC Issuer features

3.1.2.4 *Specific and Relevant Workflows*

3.1.2.4.1 Agent Bootstrap

The diagram in Figure 3-9 below describes the main steps to be performed when a Trading Agent is executed for the first time. The execution command should include at least (step 1):

- Registration of seed in Governance DLT
- An attributed DID of the Marketplace Governance Board
- a list of all platform service endpoints provided by the Governance Manager that can be used to certify the Stakeholder has all required 5GZORRO platforms successfully deployed

The stakeholder DID is created and stored in the Wallet endorser (step 2) and then the Governance Board DID is resolved to retrieve the list of existing Admin Agents and associated invitation URLs (steps 3-6). The invitation URLs are used to establish a secured connection with some existing Admin Agents (steps 7 and 8) and one of the connections is used to request the issue of a Stakeholder Credential (steps 9 and 10). Only one connection is needed to issue the Stakeholder Credential and the connection with other Admin Agents can be established as soon as the new Stakeholder is approved by the Governance Board. The Stakeholder Credential is issued in case the Marketplace Governance Board approves the stakeholder as a new Member of the Marketplace.

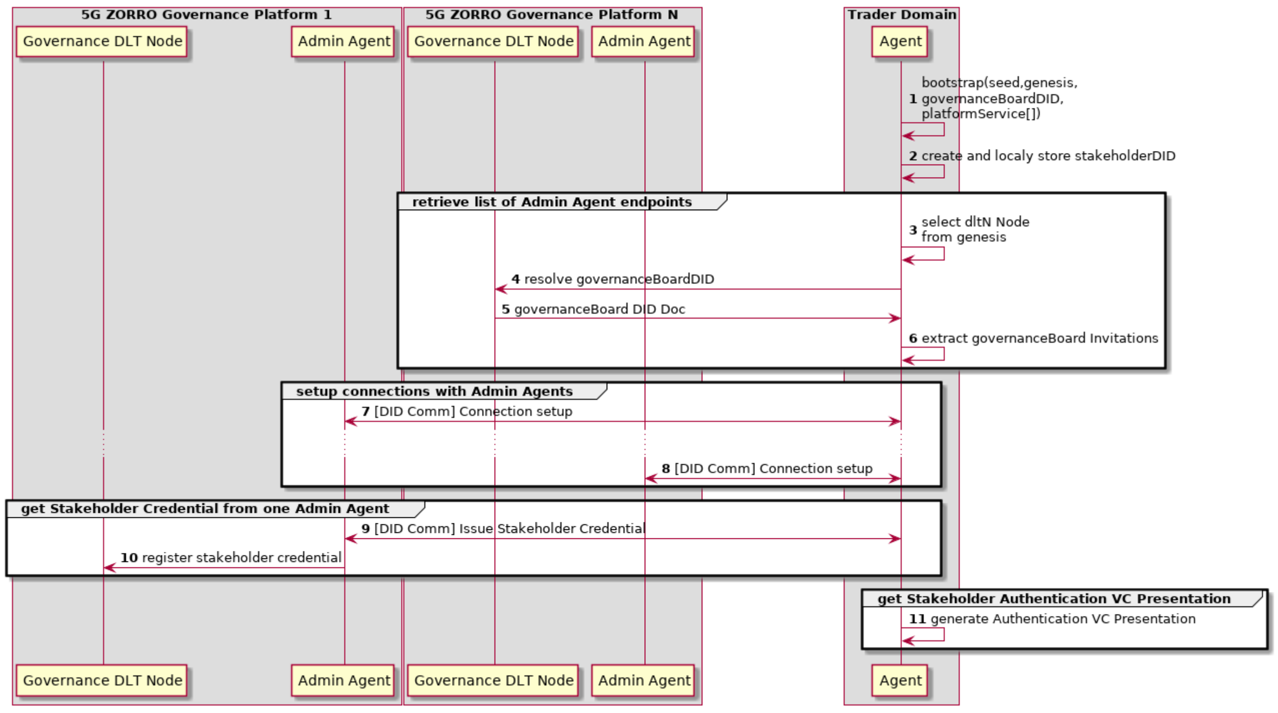
5G ZORRO Governance Platform 1
Governance DLT Node
Admin Agent
5G ZORRO Governance Platform N
Governance DLT Node
Admin Agent
Trader Domain
Agent
1 bootstrap(seed,genesis, governanceBoardDID, platformService[])
2 create and localy store stakeholderDID
retrieve list of Admin Agent endpoints
3 select dltN Node from genesis
4 resolve governanceBoardDID
5 governanceBoard DID Doc
6 extract governanceBoard Invitations
setup connections with Admin Agents
7 [DID Comm] Connection setup
8 [DID Comm] Connection setup
get Stakeholder Credential from one Admin Agent
9 [DID Comm] Issue Stakeholder Credential
10 register stakeholder credential
get Stakeholder Authentication VC Presentation
11 generate Authentication VC Presentation
Governance DLT Node
Admin Agent
Governance DLT Node
Admin Agent
Agent

Figure 3-9: Agent Bootstrap Workflow

3.1.2.4.2 License Creation

The creation of Licenses and associated DIDs is performed by calling the 'createLicense' function. A handler endpoint is provided as an input parameter to receive License processing updates. As soon as the License is requested and locally stored in the Agent Wallet with associated Service endpoints, the execution is returned with the new License DID proposal. Then, the process to issue a License DID credential containing requested claims is executed, where the Holder Agent interacts with one of the available Admin Agents by using the DID Comm Issue Credential protocol. This process, including the registration of the credential definition in the Governance DLT, is performed behind the scenes without requiring any additional interaction with the Holder Client (see how credentials are issued by using the Issuer Agent API in Section 3.1.2.4.3). As soon as the Credential is received by the Holder Agent, a DID Event is dispatched towards the Holder Handler to notify the DID is ready to be used. Optionally, the Holder Client can retrieve the Invitation object that is required by any potential 5GZORRO component interested to request the claim object associated to the new License DID, for example, the Marketplace Catalogue component operated by any interested 5GZORRO Product Consumer.

3.1.2.4.3 Admin Agent Handles Credential Issue Requests

The diagram below (see Figure 3-10) describes the main steps to issue DID Stakeholder Credentials by using the Issuer Agent API from the Identity and Permissions Manager component. The Issuer Agent API is implemented by the Admin Agent and decides and controls the issuance of DID Stakeholder Credentials requested by other Holder Agents (e.g., Trading Provider Agents) from W3C Verifiable Credential Issuer role perspective. The issue of Stakeholder Credential claims as requested by a 5GZORRO Provider is a major example of DID Credential issue. Another relevant example is the issue of Spectrum License Credentials by the Regulator Agent (see 3.1.4.4.2).

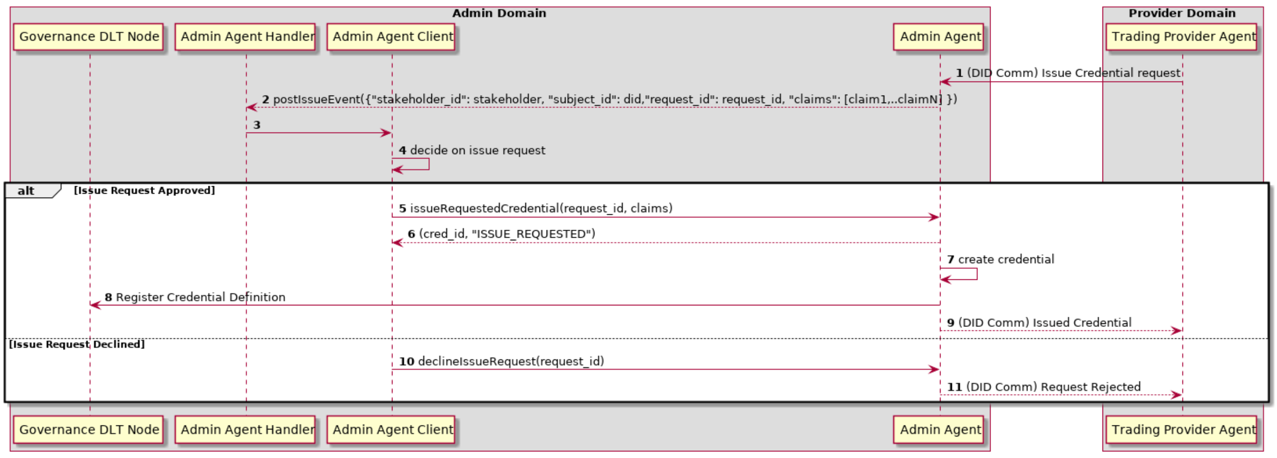
Admin Domain
Provider Domain
Governance DLT Node
Admin Agent Handler
Admin Agent Client
Admin Agent
Trading Provider Agent
1 (DID Comm) Issue Credential request
2 postIssueEvent({"stakeholder_id": stakeholder, "subject_id": did,"request_id": request_id, "claims": [claim1,..claimN] })
3
4 decide on issue request
alt
[Issue Request Approved]
5 issueRequestedCredential(request_id, claims)
6 (cred_id, "ISSUE_REQUESTED")
7 create credential
8 Register Credential Definition
9 (DID Comm) Issued Credential
[Issue Request Declined]
10 declineIssueRequest(request_id)
11 (DID Comm) Request Rejected

Figure 3-10: Credential Issue Workflow

The issue of Credentials is usually initiated from a DID Comm Issue Credential Protocol message triggered by the DID Creation process performed by a Holder Agent (see how DIDs are created by the Holder Agent in Section 3.1.2.4.3) (step 1) and an Issue Event is dispatched towards the Admin Agent Handler (step 2).

In case the request is approved (steps 3-4), the Admin Agent Client will call the 'issueRequestedCredential' function (step 5) to issue the requested credential providing as input the request id extracted from the Issue Event received in step 2. The credential is created with all cryptographic proof methods required to its verification (step 7), its definition is registered in the Governance DLT (step 8) and then transferred to the requester, the Trading Provider Agent (step 9).

If the request is declined, the Trading Provider Agent requester is informed as specified by the DID Comm Issue Credential protocol (steps 10 and 11).

3.1.2.5 *Identity and Permissions Manager APIs*

3.1.2.5.1 Authentication API

Operation name	**operatorKeyPair**	
Description	Enable generation of encoded operator key pairing for secure channel establishment	
Input Parameters		
Name	**Type**	**Description**
shared_secret	str	A component-only secret known between some 5GZORRO components to enable secure authorized requests
Output Parameters		
response	str	Encoded object containing the private key pairing
reason_code	object	If authentication fails, the reason should be provided here
Notes		
In future versions, this API can evolve to be more aligned with SIOP DID spec		

Operation name	**operatorKeyPairVerify**	
Description	Verify operator key pairing issued by another DID Agent to enable secure channel establishment	
Input Parameters		
Name	**Type**	**Description**
shared_secret	str	A component-only secret known between some 5Gzorro components to enable secure requests

did	str	The *did* that is associated in the encoded key pairing
public_key	str	The *public_key* that is associated in the encoded key pairing
timestamp	str	The *timestamp* that is associated in the encoded key pairing
Output Parameters		
response	str	Response detailing if provided key pairing is valid
reason_code	object	If refresh fails, the reason should be provided here

3.1.2.5.2 Holder Agent API

Operation name	**registerStakeholder**	
Description	Request to create a 5GZORRO Stakeholder Credential and all associated Claims	
Input Parameters		
Name	**Type**	**Description**
key	str	A component-only key known between some 5Gzorro components to enable secure authorized requests
governanceBoardDID	str	DID of a stakeholder in the Governance board domain
stakeholderRoles	object	Roles to be associated with the Stakeholder
stakeholderProfile	object	Profile to be associated with the Stakeholder
handler	str	Handler endpoint to process DID status events dispatched by the Agent
Output Parameters		
response	object	Object containing the sent response to the Admin DID Agent
reason_code	object	If refresh fails, the reason should be provided here
Notes		
The request to create a new Stakeholder DID Credential is asynchronous and the Handler endpoint must be provided to receive events about the DID. A request is submitted to an ADMIN Agent to issue associated credentials and may be subject to Governance Board decision according to the adopted Governance Model.		

Operation name	**readStakeholderStatus**	
Description	Request to read Stakeholder status	
Input Parameters		
Name	**Type**	**Description**
state	str	The target state to check for the Stakeholder Credential
Output Parameters		
response	object	Object containing the query response by the DID Agent

Operation name	**readStakeholderDID**	
Description	Request to read Stakeholder by associated DID	
Input Parameters		
Name	**Type**	**Description**
did	str	The target DID
Output Parameters		
response	object	Object containing the query response by the DID Agent

Operation name	**registerLicense**	
Description	Request to create a 5GZORRO Stakeholder License Credential and all associated Claims	
Input Parameters		
Name	**Type**	**Description**
Id_token	str	Token available in an approved Stakeholder DID Credential
stakeholderServices	object	Services to be associated with the License
Output Parameters		

response	object	Object containing the sent response to the Regulator DID Agent
reason_code	object	If refresh fails, the reason should be provided here
Notes		
The request to create a new Stakeholder License Credential is asynchronous and the Handler endpoint must be provided to receive events about the DID. A request is submitted to a REGULATOR Agent to issue associated credentials and may be subject to Governance Board decision according to the adopted Governance Model.		

Operation name	**readLicense**	
Description	Lists all or specific agent's unique License Credentials on the wallet	
Input Parameters		
Name	**Type**	**Description**
Output Parameters		
response	object	Object containing all registered licenses of a Holder DID Agent

Operation name	**createDID**	
Description	Request to create a private 5GZORRO DID and all associated Claims	
Input Parameters		
Name	**Type**	**Description**
token	str	Token available in an approved Stakeholder DID Credential
type	str	5GZORRO Subject type including "Edge", "Cloud", "Spectrum", etc.
handler	str	Handler endpoint to process DID status events dispatched by the Agent
Output Parameters		
did	str	New DID created
state	str	State of the DID creation request

Operation name	**readDIDStatus**	
Description	Request to read DID status	
Input Parameters		
Name	**Type**	**Description**
did	str	The target DID
Output Parameters		
state	str	The DID Status

Operation name	**readDID**	
Description	Lists all or specific agent's unique Decentralized Identifiers (DID) on the wallet	
Input Parameters		
Name	**Type**	**Description**
did	str	(Optional) Decentralized Identifier value to be read
Output Parameters		
posture	str	The DID Status
services	object	Services included in the DID Document
credentials	object	Verified Credentials associated to the DID
Notes		
To fetch a specific DID, use the optional *did* input parameter		

3.1.2.5.3 Holder Agent Handler API

The Holder Agent Handler API is implemented by components that are clients of the Holder Agent API.

Every time the Holder DID status is updated, a serialized DID Event JSON object is sent to the provided Handler URL via POST requests. The full set of properties of the DID Event payload is listed below:

Parameter	Type	Description
did	DID	identifier of the subject which status was changed and reported with this object
state	String	The DID state value
description	String	Any optional textual description of the event

The possible DID state transitions are described in the state machine diagram provided below (see Figure 3-11):

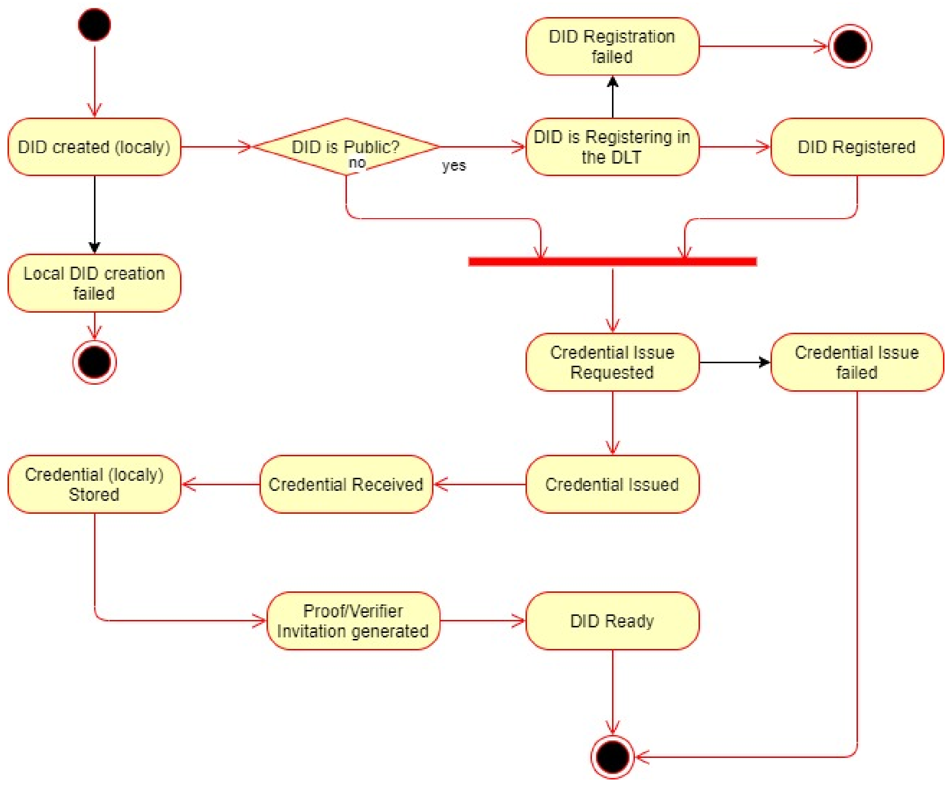

Figure 3-11: DID State Machine

3.1.2.5.4 Issuer Admin Agent API

Note: An Issuer Admin Agent also contains all functionalities specified in 3.2.5.1 & 3.2.5.2

Operation name	**issueRequestedCredential**	
Description	Creates a credential requested by some Holder Agent	
Input Parameters		
Name	**Type**	**Description**
request_id	str	Identifier of the issue request
claims	object	List of claims to be included in the Credential
Output Parameters		
cred_id	str	Credential identifier
state	str	State of the Credential sent
Notes		

The request_id is extracted from the incoming issue request event processed by the Admin Handler.

Operation name	**declineIssueRequest**	
Description	Declines the issue of a credential requested by some Holder Agent	
Input Parameters		
Name	**Type**	**Description**
request_id	str	Identifier of the requested issue which is declined
Output Parameters		
N/A	--	--
Notes		
The *request_id* is extracted from the incoming issue request event processed by the Admin Handler.		

Operation name	**readCredentials**	
Description	List Credentials issued by the agent	
Input Parameters		
Name	**Type**	**Description**
N/A	--	--
Output Parameters		
credential	object	Requested Credentials
Notes		
The specified output fields are among the most important ones.		

Operation name	**readDeclinedIssueRequests**	
Description	Lists declined sent Credentials provided by the agent	
Input Parameters		
Name	**Type**	**Description**
N/A	--	--
Output Parameters		
credential	object	Requested Credentials
Notes		
The specified output fields are among the most important ones.		

Operation name	**readPendingdIssueRequests**	
Description	Lists pending Credentials to be resolved by the agent	
Input Parameters		
Name	**Type**	**Description**
N/A	--	--
Output Parameters		
credential	object	Requested Credentials
Notes		
The specified output fields are among the most important ones.		

Operation name	**revokeStakeholderCredential**
Description	This method is utilised to revoke an active Stakeholder credential
Input Parameters	

Name	Type	Description
stakeholder_did	str	DID that identify the credential associated to the Stakeholder to be revoked
Id_token	str	Token available in an Admin Stakeholder DID Credential
Output Parameters		
state	str	Information on the completion status of the current action.
Notes		
This is an asynchronous function where the result will be returned to the Holder DID Handler in the credential topic.		

3.1.2.5.5 Issuer Admin Agent Handler API

The Issuer Admin Agent Handler API is implemented by components that decide about issue request from Holder Agents and which are clients of the Admin Agent API.

Different types of JSON records are sent to the provided Handler URL via POST requests.

When an event is dispatched, the record `topic` is appended as a path component to the URL, for example:

`https://handler.host.example`

becomes

`https://handler.host.example/topic/issueRequest`

when an issue request record is received from an Agent. A POST request is made to the resulting URL with the body of the request comprised by a serialized JSON object. The full set of properties of the current set of handler events payloads are listed below.

Operation name	**issueEvent**	
Description	Credential issue event	
Input Parameters		
Name	**Type**	**Description**
stakeholder_id	DID	Identifier of the stakeholder requesting the credential issue
subject_id	DID	Identifier of the subject to be associated with requested credential
request_id	String	Identifier of the Issue request that will be used to issue the credential when the “issueRequestedCredential” function is called.
Claims	Object List	List of claims to be included in the requested credential
Output Parameters		
N/A		

Operation name	**revokeEvent**	
Description	Credential revoke event	
Input Parameters		
Name	**Type**	**Description**
credential_id	DID	Identifier of the credential to be revoked
Output Parameters		
N/A		

3.1.2.5.6 Regulator Agent API

Note: A Regulator Agent also contains all functionalities specified in 3.2.5.1 (except License requests) & 3.2.5.2

Operation name	**issueRequestedLicenseCredential**	
Description	Creates a License credential requested by some Holder Agent	
Input Parameters		
Name	**Type**	**Description**
request_id	str	Identifier of the issue request
claims	object	List of claims to be included in the Credential
Output Parameters		
cred_id	str	Credential identifier
state	str	State of the Credential sent
Notes		
The request_id is extracted from the incoming issue request event processed by the Regulator Handler.		

Operation name	**declineLicenseIssueRequest**	
Description	Declines the issue of a License credential requested by some Holder Agent	
Input Parameters		
Name	**Type**	**Description**
request_id	str	Identifier of the requested issue which is declined
Output Parameters		
N/A	--	--
Notes		
The *request_id* is extracted from the incoming issue request event processed by the Admin Handler.		

Operation name	**readLicenseCredentials**	
Description	List License Credentials issued by the agent	
Input Parameters		
Name	**Type**	**Description**
N/A	--	--
Output Parameters		
credential	object	Requested Credentials
Notes		
The specified output fields are among the most important ones.		

Operation name	**readDeclinedLicenseIssueRequests**	
Description	Lists declined sent License Credentials provided by the agent	
Input Parameters		
Name	**Type**	**Description**
N/A	--	--
Output Parameters		
credential	object	Requested Credentials
Notes		
The specified output fields are among the most important ones.		

Operation name	**readPendingdLicenseIssueRequests**	
Description	Lists pending License Credentials to be resolved by the agent	
Input Parameters		
Name	**Type**	**Description**

N/A	--	--
Output Parameters		
credential	object	Requested Credentials
Notes		
The specified output fields are among the most important ones.		

3.1.2.5.7 Verifier Agent Handler API

The Verifier Agent Handler API is implemented by components that request proofs about DID Credential Claims and which are clients of the Verifier Agent API.

Every time the Proof status is updated, a serialized Proof Event JSON object is sent to the provided Handler URL via POST requests. The full set of properties of the Proof Event payload is listed below:

Parameter	Type	Description
proof_id	String	Identifier of the proof request which status was changed and reported with this object
state	String	The Proof state value
description	String	Any optional textual description of the event

The possible Proof state transitions are described in the state machine diagram provided in Figure 3-12 below:

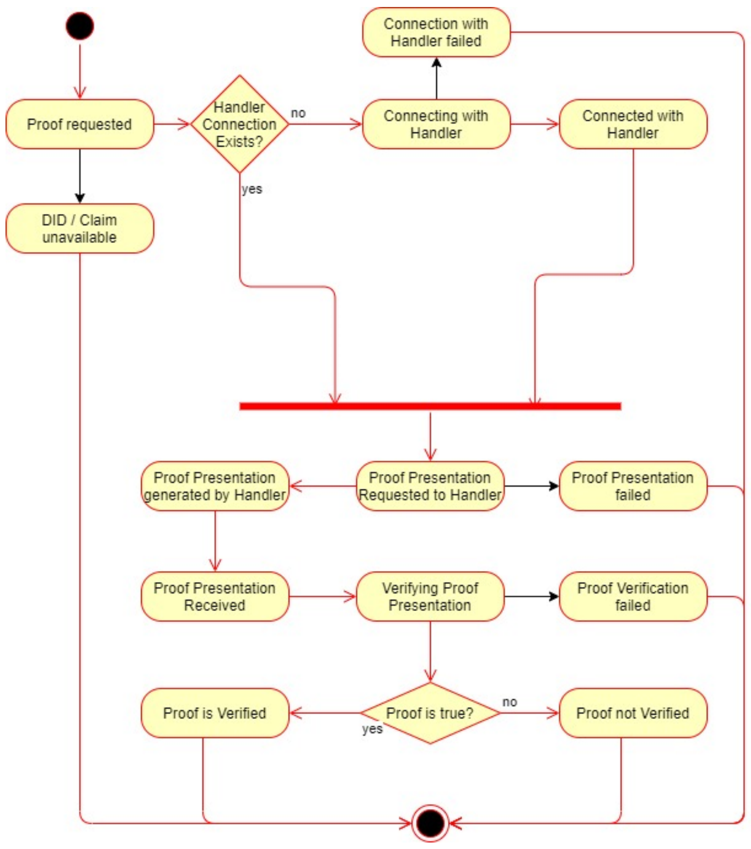
Connection with Handler failed
Proof requested
Handler Connection Exists?
no
Connecting with Handler
Connected with Handler
yes
DID / Claim unavailable
Proof Presentation generated by Handler
Proof Presentation Requested to Handler
Proof Presentation failed
Proof Presentation Received
Verifying Proof Presentation
Proof Verification failed
Proof is Verified
Proof is true?
no
yes
Proof not Verified

Figure 3-12: Proof State Machine

3.1.3 Legal Prose Manager

Legal Prose Management (LPM) is a software module deployed as part of the governance platform by admin stakeholders and provides smart legal contract templating services to the 5GZORRO platform. Stakeholders can propose new templates and updates, which are then stored into the module's internal template repository and can be utilized by Markeplace traders. The name of the module has been changed in Legal Prose Repository, as briefly explained in Section 3.1.3.2.

Templates comprise legal prose and a machine-readable model (Ricardian contract) that can be queried by stakeholder marketplace platforms to build concrete SLAs and Licence Terms. It is crucial that templates also carry general (or specific where required) reference to applicable legal frameworks in order to stand up to scrutiny should a civil legal dispute be raised. It is expected that the drafting of templates will be a collaborative effort between legal and technical departments of a proposing stakeholder, and the verification and approval processes involve the review of both technical and legal correctness.

Marketplace traders will be able to utilise templates provided by the platform, but also propose new ones to meet their needs. These templates can be developed by legal/technical roles and submitted to the 5GZORRO ecosystem. Governance stakeholders would then partake in an approval process through a voting process that results in a DID assignment, giving rise to global cross-domain identifiability, authenticity and verifiable status of templates.

3.1.3.1 *Legal Prose Manager Design Details*

Legal prose management plays a key part in establishing consistency, trust, and automation across the 5GZORRO marketplace, with the following KPIs at the heart of its considered design:

- Ability for untrusted parties to negotiate, set-up and operate a new technical/commercial relationship via a Smart Contract for 3rd-party resource leasing/allocation with associated SLA (*KPI target: Smart Contract for 3 or more untrusted parties*)
- Implement/correlate technical service configurations and SLA monitoring interactions between multiple parties (*KPI target: SLA measurements and validation from at least 3 operators involved in a multi-party service chain*)

Figure 3-13 illustrates the key entities that comprise the module, which also have dependencies on the DLT Governance Management functions through the Identity and Permission Manager (ID&P, see Section 3.1.2) for the release of Template DIDs. Definition and editing of such templates are undertaken through the Governance Portal.

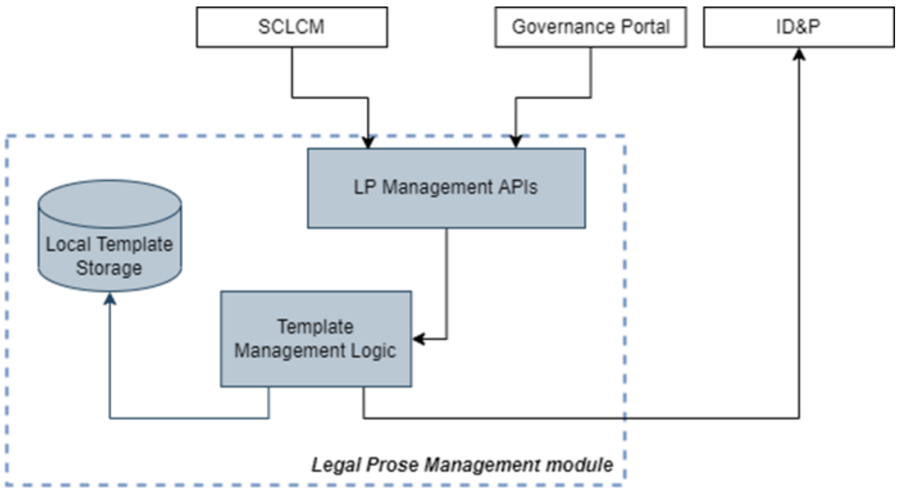

Figure 3-13: Legal Prose Management Module Architecture

- **API** – A set of endpoints for Template manipulation. These endpoints are available in the local domain and are consumed by other modules from 5GZORRO Platform in particular, from the Governance Portal and the Smart Contract Lifecycle Manager (LCM), described in Section 3.1.4 and in Section 3.2.2, respectively.

- **Template Manager** – Functional entity that implements the logic of the module to ensure templates management.
- **Local Template Storage –** Legal template repository.

3.1.3.2 *Legal Prose Template Technology*

Legal Templates have the form of JSON files encompassing both the legal prose (text in natural language) and the data model, used to fill the template with specific values. In the initial Legal Prose Management design, the logic for the manipulation of the template was based on Accord Project [33] a framework that along with the aforementioned elements (legal prose and data model) includes also an executable business logic. The combination of the 3 elements results in a human-machine readable smart agreement. The main purpose of introducing Accord was to provide the templates of an internal logic able to detect potential violations of the agreements e.g., by posting measurements collected by 5GZORRO monitoring service. Subsequent design of 5GZORRO platform moved such detecting logic out of LPM, making the use of Accord technology no longer necessary.

Without the business logic, the module stores the legal templates that are used by other modules in the platform and the name has been changed in **Legal Prose Repository**. The current technology used to build a template is called JSON Template Language [36] and combines parameters with text in natural language, as in the example below.

Example data model

```
{
  "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

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

3.1.3.3 *Specific and Relevant Workflows*

In Figure 3-14 is reported the workflow describing the steps executed to storage a Template in the LPR and its usage during the product offer creation process.

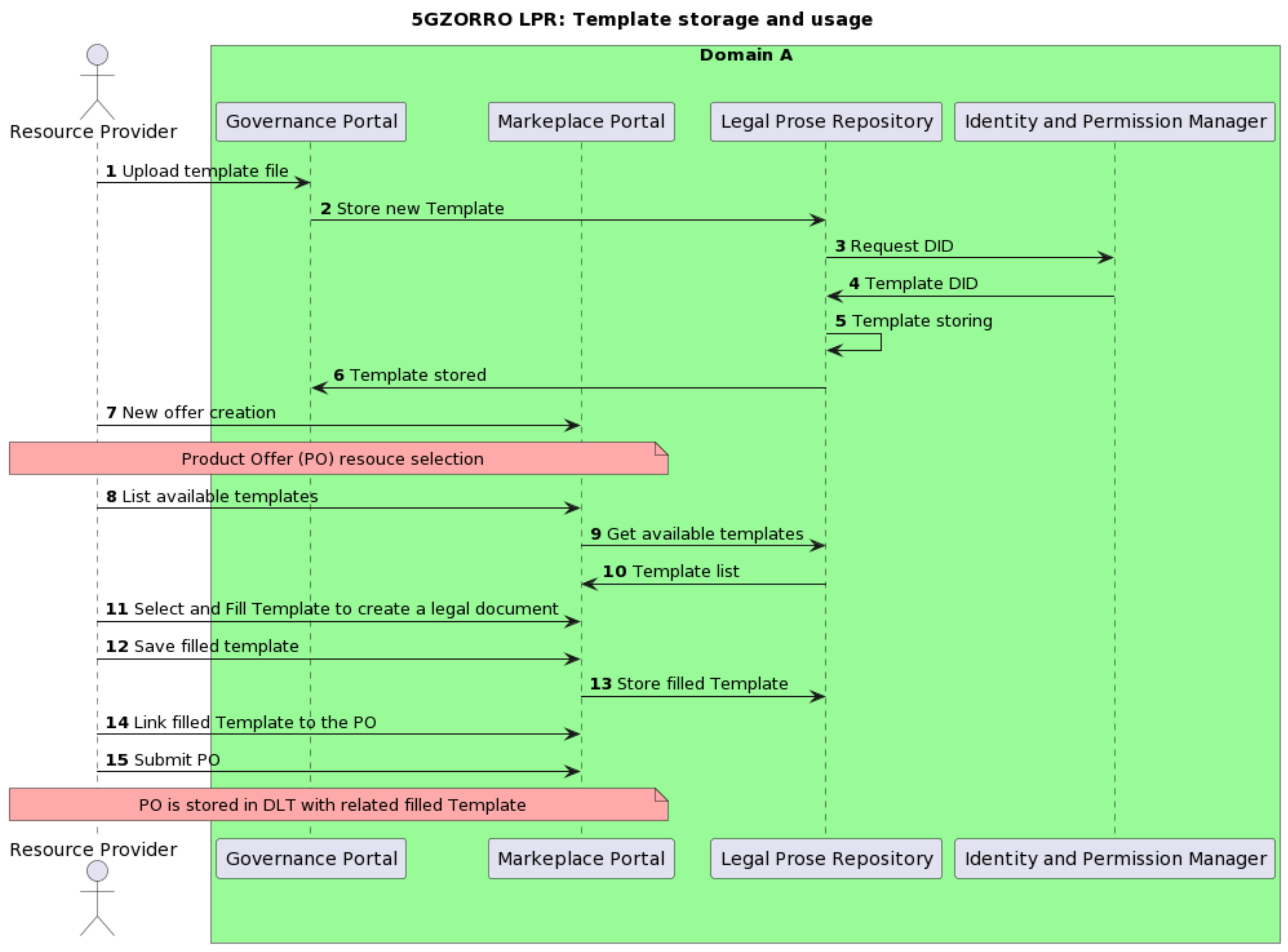
5GZORRO LPR: Template storage and usage
Domain A
Resource Provider
Governance Portal
Markeplace Portal
Legal Prose Repository
Identity and Permission Manager
1 Upload template file
2 Store new Template
3 Request DID
4 Template DID
5 Template storing
6 Template stored
7 New offer creation
Product Offer (PO) resouce selection
8 List available templates
9 Get available templates
10 Template list
11 Select and Fill Template to create a legal document
12 Save filled template
13 Store filled Template
14 Link filled Template to the PO
15 Submit PO
PO is stored in DLT with related filled Template

Figure 3-14: Template storage and usage

Step 1: A Resource Provider uploads a new Template (one or more) throught the Governance Portal.

Step 2-6: The Resource Provider request triggers the LPR, which in turn requests a new DID for the Template to be stored. After DID is granted, the Template is correctly stored in the LPR internal storage and the Resource provider is notified through the Governance Portal.

Step 7: The Resource Provider creates a new offer and wants to associate it a legal agreement based on the Template previously stored.

Step 8-10: The Resource Provider requests a list of available Templates that is returned by the LPR.

Step 11-13: Resource Provider starts filling the selected templates with specific values in order to create e legal document that will be stored in the LPR.

Step 14-15: At this point, the filled Template is ready to be linked to the Product Offer that, once submbitted will be stored in the Markteplace DLT.

3.1.3.4 *Legal Prose Manager APIs*

Operation name: **getLegalProseTemplates**		
Description	API endpoint to retrieve a filtered list of legal prose templates to base an SLA or Licensing Term on	
Input Parameters	**Type**	**Description**

	categoryFilter	List<TemplateCategory>	Optional comma separated list of TemplateCategory's to filter the response by
	filterText	String	Text to filter results by – applied to name and description of templates.
Output Parameters		**Type**	**Description**
	templates	List<LegalProseTemplate>	A list of legal statements for the requester to select from

Operation name: **getLegalProseTemplate**			
Description		API endpoint to retrieve a single template by identifier	
Input Parameters		**Type**	**Description**
	identifier	String	DID corresponding to a template
Output Parameters		**Type**	**Description**
	template	LegalStatmentTemplate	A legal statement matching the requested DID

Operation name: **proposeNewLegalProseTemplate**			
Description		API endpoint to create a new template.	
Input Parameters		**Type**	**Description**
	proposeTemplateRequest	ProposeTemplateRequest	Name, description and category of the proposed template
	templateFile	MultipartFile	Zip file that encapsulates the parameterised specification of a Ricardian contract
Output Parameters		**Type**	**Description**
	id	String	The ID of the newly created template proposal
Notes			
Templates are treated as immutable entities and are subject to a governance process. A new template would be subject to a governance process before it becomes available for use. As such, it would be in a 'PROPOSED" state until a DID is released for it. The returned identifier will be replaced by the DID when the latter is released.			

Operation name: **removeLegalProseTemplate**			
Description		API endpoint to archive a template definition (soft delete)	
Input Parameters		**Type**	**Description**
	did	String	DID of the template to be archived
Output Parameters		**Type**	**Description**
	N/A		
Notes			
The removed template will be marked as ARCHIEVED			

Operation name: **updateTemplateIdentity**			
Description		API endpoint to update the status of a given template definition and complete its creation with the DID released by the Identity & Permission Manager after the governance process.	
Input Parameters		**Type**	**Description**
	id	UUID	DID of the template

	state	DIDState CSDto	Credentilas released by the Identity & Permission Manager containing also the DID that will be attached to the proposed template
Output Parameters		**Type**	**Description**
Notes			
If the template is in "PROPOSED" state, then it will move to either "ACTIVE" or "REJECTED"			

3.1.4 Governance portal

Being a decentralised system, specific modules of 5GZORRO are governed by a set of stakeholders who have the power to collectively decide on actions previously requested by other stakeholders. These specific actions are subject to manual approval and all actions are registered on-chain, whereby a transaction properly signed by a stakeholder's private key ensures authenticity of such request and/or approval. The governance models dictating the different steps to be validated before approval (business logic) are encoded as smart contracts deployed on-chain. Thus, 5GZORRO solution automates the business processes that should take place after a final decision has been made. The governance 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.

Focused on the interaction with such governance structure and relevant software modules, the Governance Portal dApp (decentralised Application) is a component of 5GOZRRO's web GUI which supports Governance user stories. Table 3-2 describes the supported user stories, their inputs and their expected behaviour, and the software modules they interact with, namely the Identity and Permissions Manager (Section 3.1.2) and the Legal Prose Manager (Section 3.1.3).

Table 3-2: Definition of Governance Portal's User Stories

US id	**User Story** (US)	**SW Module** (Interface)	**API endpoint / RPC**	**Input**	**Outputs**
R1	As a Stakeholder, I must be able to request access to join 5GZORRO	Identity & Permissions Manager	*registerStakeholder()*	*Key, governanceBoardDID, stakeholderRoles, stakeholderProfile, handler*	Proposed Stakeholder Verifiable Credential sent to the Admin DID Agent

R2	As a Stakeholder trying to join the Marketplace, I must be able to watch the status of my onboarding process	Identity & Permissions Manager	*readStakeholderStatus()*
			
			OR
			
			readStakeholderDID()
				State, did	Stakeholder Verifiable Credential & associated state

R3	As a Governance Admin, I must be able to revoke other stakeholders access to the Marketplace	Identity & Permissions Manager	*revokeStakeholderCredential()*	*Stakeholder_did, id_token*	Revocation of the selected Stakeholder Verifiable Credential
R4	As a Governance Admin, I must be able to accept	Identity & Permissions Manager	*issueRequestedCredential()*	*Request_id, claims*	Issuance of the selected Stakeholder

US id	**User Story** (US)	**SW Module** (Interface)	**API endpoint / RPC**	**Input**	**Outputs**
	other stakeholders onboarding credentials				Verifiable Credential
R5	As a Governance Admin, I must be able to refuse other stakeholders onboarding credentials	Identity & Permissions Manager	*declineIssueRequest()*	*request_id*	Declinal of the selected Stakeholder Verifiable Credential
R6	As a Stakeholder, I must be able to request a license to be registered in 5GZORRO	Identity & Permissions Manager	*registerLicense()*	*Id_token, stakeholderServices*	Proposed License Verifiable Credential sent to the Admin DID Agent
R7	As a Stakeholder trying to have a License resolved in 5GZORRO, I must be able to watch the status of my licensing process	Identity & Permissions Manager	*readLicense()*		License Verifiable Credentials & associated states
R8	As a License Regulator, I must be able to accept other stakeholders license credentials	Identity & Permissions Manager	*issueRequestedLicenseCredential()*	*Request_id, claims*	Issuance of the selected License Verifiable Credential
R9	As a License Regulator, I must be able to refuse other stakeholders license credentials	Identity & Permissions Manager	*declineLicenseIssueRequest()*	*request_id*	Declinal of the selected License Verifiable Credential
RL1	As a Stakeholder, I must be able to compose and submit a new Smart Legal Template	Legal Prose Repository	*proposeNewLegalStatementTemplate()*	A regular form with: Name, Description Template (file upload)	*proposedTemplateDID*
RL2	As a Stakeholder, I must be able to retrieve all Smart Legal Templates pertaining to SLAs	Legal Prose Repository	*getLegalStatementTemplates()*	A text area where the Stakeholder can input plain text to be used as the criteria parameter	*List of LegalStatementTemplate (5GZORRO IM)*
RL3	As a Stakeholder, I must be able to get the details of	Legal Prose Repository	*getLegalStatementTemplate()*	Template DID	*LegalStatementTemplate (5GZORRO IM)*

US id	User Story (US)	SW Module (Interface)	API endpoint / RPC	Input	Outputs
	a particular Smart Legal Template				

3.2 Marketplace Platform Services

As described in D2.4 [1], 5GZORRO Marketplace Platform (see Figure 3-1) is responsible for fostering multi-party collaboration for on-demand end-to-end service provisioning in dynamic 5G environments required to support and to simplify simplify the processes for procurement, deployment and management of the 5G flexible networks. In general terms, the Marketplace Platform enables the creation and the acquisition of product offers that represent a variety of exposed telco digital assets (i.e., resources and services). These offers include individual resources such as infrastructure components (like cloud, edge, connectivity, wireless or cellular access) and VNFs/CNFs; as well as composed bundles in the form of services/slices. To achieve its goals, the Marketplace Platform provides the tools for 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/consumption; and the continuous SLA management to ensure the fulfilment of agreed conditions.

5GZORRO Marketplace platform is realised as a set of software modules that provide core trustworthy marketplace services, either to 5GZORRO stakeholders or to other 5GZORRO platforms. These services are presented in Figure 3-15 and covered in the following subsections: the Resource & Service Offer Catalogue Services in Section 3.1.13.2.1, the Smart Contract Lifecycle Services in Section 3.1.23.2.2, and the Marketplace Portal in Section 3.1.43.2.3.

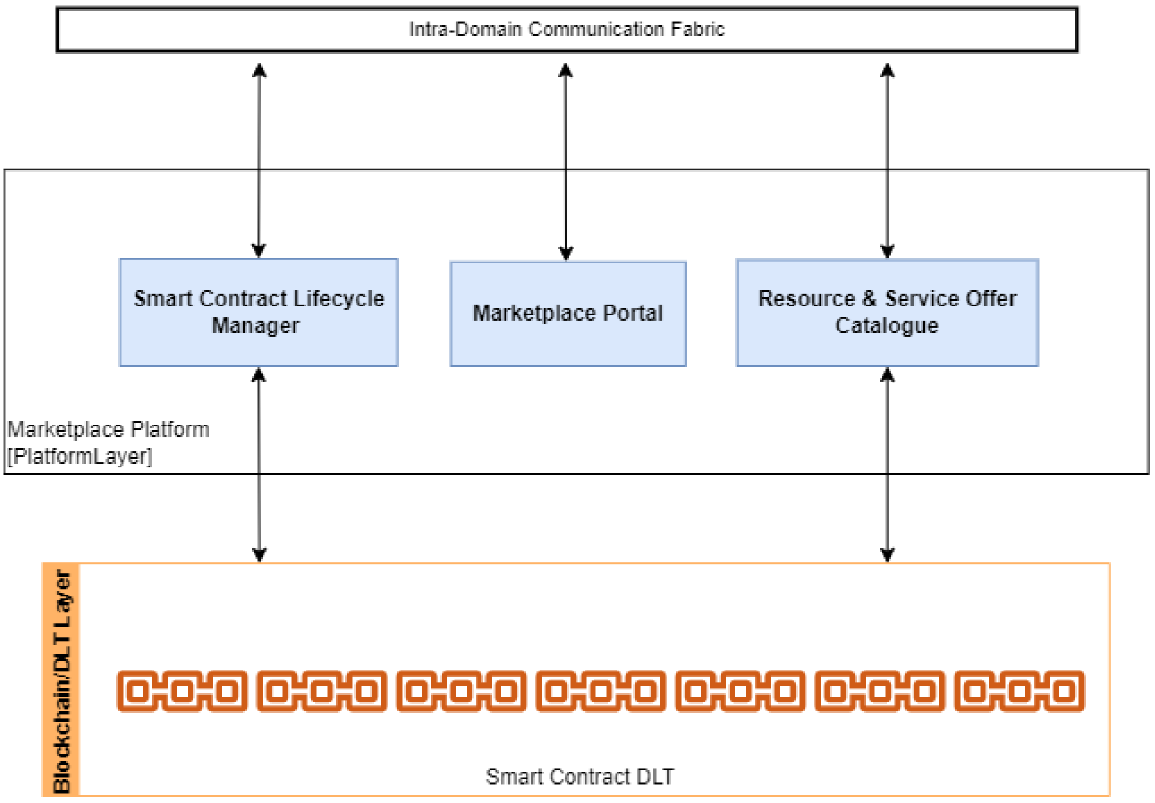
Intra-Domain Communication Fabric
Smart Contract Lifecycle Manager
Marketplace Portal
Resource & Service Offer Catalogue
Marketplace Platform [PlatformLayer]
Blockchain/DLT Layer
Smart Contract DLT

Figure 3-15: Marketplace Platform Services

- ***Marketplace Portal***: Storefront for offer composition, searching and selection to enable a business-compliant and user-friendly offer design and display. Although a portal is provided for facilitating user access to the platform, supported services are exposed for programmable interaction between the Marketplace and other components of the 5GZORRO platform.

- ***Resource and Service Offering Catalogue***: Portfolio of available (resource and service) digital assets and corresponding (product) offers for 5GZORRO parties to offer, discover, request, and consume within the marketplace. This module defines how assets and products are modelled through the use of standard open APIs.
- ***Smart Contracts Lifecycle Manager***: Key driver of how offers, SLAs and commercial agreements are autonomously created and processed through smart contracts. Integration with different DLTs implementation is supported through use of ledger-specific drivers in order to certify transactions ensuring transparency and trust among the participating stakeholders.
- ***Communication Fabric***: Entity that takes care of the interoperation and communication between modules of the Marketplace Platform. Inspired by the ZSM architecture [6], this component facilitates the interaction among Marketplace services by playing both the roles of service consumer and service producer.

3.2.1 Resource and Service Offering Catalogue

As part of the 5GZORRO Marketplace Platform, the Resource and Service Offering Catalogue (RSOC) is the module responsible for collecting the 5G assets that are available to be traded among providers and customers. This decentralized repository enables the processes of registering, browsing, and ordering of product offers on-demand across multiple parties acting as infrastructure providers, spectrum traders, VNF vendors and/or service providers. As described later, these operations are modelled via TM Forum OpenAPIs, which allow the different stakeholders to interact with the 5GZORRO marketplace and consume the exposed capabilities.

In 5GZORRO, the marketplace catalogue is mainly composed of three different object types:

I. The resource catalogue contains the inventory of available resources (e.g., VNF/CNF, RAN elements, spectrum, edge/core resources).

II. The service catalogue contains the inventory of available services (e.g., network services, communication services/slices), which are defined as collections of resources.

III. The product catalogue contains the inventory of available product offers, which add the business terms (pricing, SLA, etc.) associated to the previous two asset categories.

Essentially, 5GZORRO stakeholders acting as offer providers consolidate resources and/or services by abstracting the features and characteristics from their technical specification. To add the legal and business considerations, these technical specifications are then wrapped into commercial product offers, stored in the product catalogue and exposed to the market by means of smart contracts.

3.2.1.1 *5GZORRO Specific enhancements*

In 5GZORRO, the RSOC represents a multi-category repository, addressing the needs in the entire lifecycle of different stakeholders' asset portfolio. This module is responsible for storing and managing the lifecycle of the different offers that are available to be traded in the marketplace, while keeping a consistent and up to date record of available offers.

While resources and services are considered internal representations of provider's assets, the use of product offers as customer facing assets allows the inclusion of business terms such as corresponding pricing models and service level agreements. Considering such information contributes to better represent the nature of inter-party commercial relationships and provides key features for searching and retrieving offers based on such criteria.

Once a resource or service is included as an item into a product offer, it becomes available to be traded. To do so, the Catalogue interacts with the Marketplace DLT via the Smart Contract Lifecycle Manager for the conformation of the corresponding smart contracts and posterior deployment into the ledger.

Additionally, the RSOC enables the automated discovery of available product offers from different domains and service providers. It also contains the required logic to place a product order, which represents the associated commercial agreement between providers and consumers.

3.2.1.2 *Design Details*

The Resource and Service Offering Catalogue plays a fundamental role in the attainment of the following KPI:

- Automatically discover and "inventorize" various types of resources (i.e., compute, storage, network at core, edge, far-edge), spectrum and services capabilities from different domains and service providers (KPI target: distribution of resource updates and discovery in less than 10 mins).

In order to achieve the aforementioned functionalities, the internal architecture of the Resource and Service Offering Catalogue, shown in Figure 3-16, comprises the following main entities:

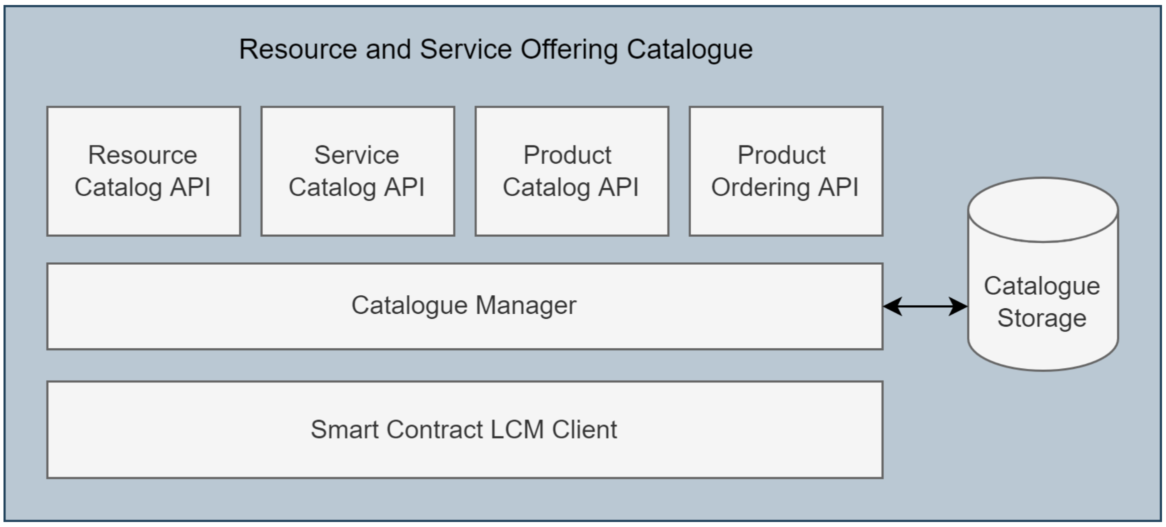

Figure 3-16: Resource and Service Offering Catalogue (RSOC) Architecture

- **Catalogue Northbound Interface (NBI)**: a set of TM Forum APIs is implemented to expose the module capabilities regarding the lifecycle management of assets (resources and services), offers and orders. More details about these APIs are depicted in the following subsections.
- **Catalogue Manager**: Main functional block that implements the logic of the 5GZORRO Catalogue, triggering internal interactions (read/write of database entries), triggering event notifications to registered listeners and reacting to events published in the Communication Fabric regarding the creation, update, and removal of product offers.
- **Catalogue Storage**: Database where the available entities and dependencies associated to Resource, Service and Product models, among others, are stored. The information models adopted are fully compliant with the TM Forum SID [37] and are described in more details in following subsections.
- **Smart Contract LCM Client**: is the entity in charge of interacting with the Smart Contract LCM, triggering API calls to conduct the deployment of smart contracts related to a product offer that becomes available to be traded, product orders, as well as the update and removal of previously deployed smart contracts.

3.2.1.3 *Specific and relevant workflows*

The services offered by the Catalogue contemplate the following supported workflows.

3.2.1.3.1 On-boarding of (Resource and Service) Assets

Resource providers on-board and store in the 5GZORRO Offering Catalogue the technical description of available resource and service assets. The required interaction is shown in Figure 3-17 for the case of a resource asset. In summary, resource providers fetch, via the Marketplace Portal, the assets that are available from the underlying 5G Virtualized platform and exposed through the Any Resource Manager (xRM) module. Then, after selecting a given asset, the onboarding request is trigerred from the Portal and delegated to the RSOC for the creation of the corresponding resource item.

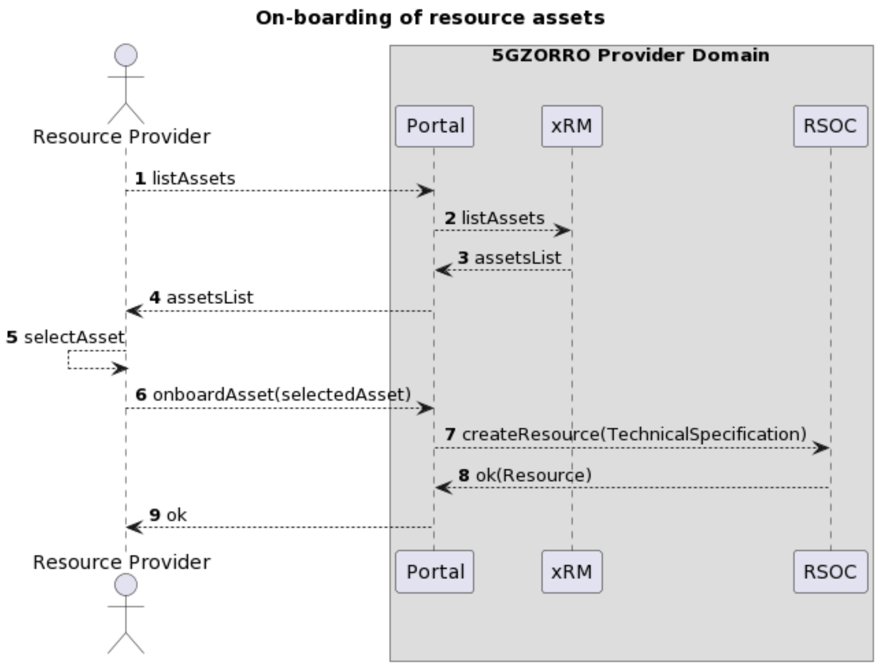

Figure 3-17: On-boarding of resource asset workflow

Given the diverse nature of considered resources, the provided asset specification (see step 7 in Figure 3-17) includes, but is not limited to, resource category (cloud, edge, spectrum, RAN, etc.), geographic location and main characteristics that outline the offered capabilities. Stored asset information also contains a reference to the corresponding management entity, abstratcted by the xRM module, that exposes and controls this resource within the 5GZORRO platform. Note that, in a similar way, assets can be off-boarded once they are no longer available for trading.

In the case of spectrum assets, previous to the on-boading stage into the Marketplace, their registration into the corresponding resource manager (i.e., the Spectrum Resource Manager (sRM)) is subject to compliance with the spectrum rights of the associated resource provider. This information, backed by a spectrum certificate that has gone through the approval of the Regulator, is used to ensure that only valid spectrum assets can be registered at the underling resource manager in order to safeguard from competition distortion ex-ante.

3.2.1.3.2 Composition and Publishing of Product Offers

Based on on-boarded assets, product offerings can be quickly assembled at the Catalogue following the workflow shown in Figure 3-18. Main steps rationale is described next.

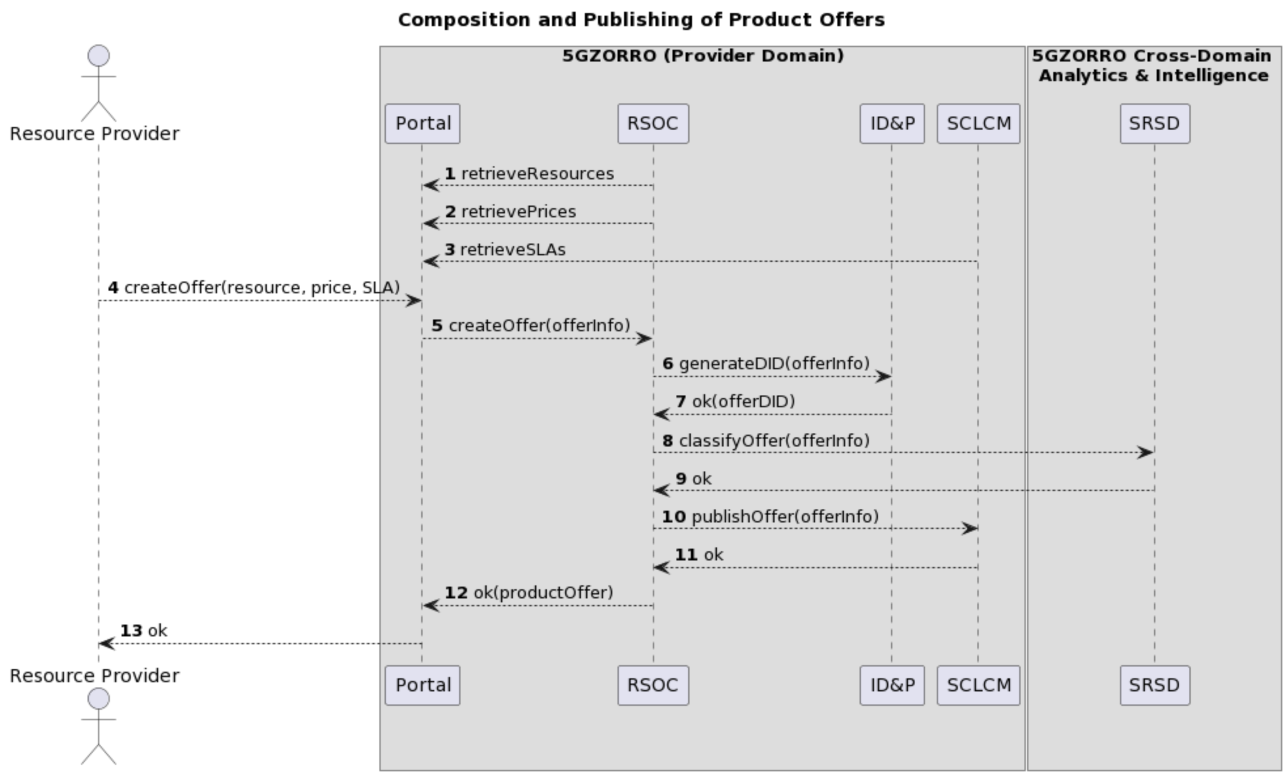
Composition and Publishing of Product Offers
5GZORRO (Provider Domain)
5GZORRO Cross-Domain Analytics & Intelligence
Resource Provider
Portal
RSOC
ID&P
SCLCM
SRSD
1 retrieveResources
2 retrievePrices
3 retrieveSLAs
4 createOffer(resource, price, SLA)
5 createOffer(offerInfo)
6 generateDID(offerInfo)
7 ok(offerDID)
8 classifyOffer(offerInfo)
9 ok
10 publishOffer(offerInfo)
11 ok
12 ok(productOffer)
13 ok

Figure 3-18: Composition and publishing of product offers workflow

Step 1-3: In addition to the description of the corresponding resource to be put into the Marketplace for trading, pricing information and SLA terms are also required for the composition of a product offer. Following a consistent and standardized component definition, involved dependencies such as the price can be created once and reused many times as part of new product offerings. The SLA corresponding to the conceived product is also associated to the offer, which is retrieved from the Smart Contract Lifecycle Manager (SCLCM), where an SLA manager sub-module is in charge of conforming a variety of SLAs based on suitable contract templates.

Step 4-5: Based on selected items (resource, price and SLA), resource providers interact with the Portal to conform a product offer. Note that additional information about the related offer must be also provided (e.g., name, description, category, location, among others).

Step 6-7: Once the product offer is assembled, the Catalogue interacts with the Identity and Permissions (ID&P) module, to retrieve the unique distributed indentifier (DID) that is associated to every offer, in order to globally indentify offers across the distributed 5GZORRO ecosystem.

Step 8-9: Likewise, the newly created product offer is ingested into the Smart Resource and Service Discovery (SRSD) application for the execution of ML-based classification later explained in Section 3.3.

Step 10-11: Similarly, the Catalogue interacts with the SCLM for the deployment and commitment into the ledger of the smart contract related to the new product offering.

3.2.1.3.3 Retrieval of DLT-announced Product Offers

In order to keep a consistent and distributed "knowledge" about available marketplace offers, Catalogue instances subscribe to DLT-events related to offer and order registration/update/removal that are communicated via the (Intra-domain) Communication Fabric. This procedure is depicted in Figure 3-19 for the case of a new offer.

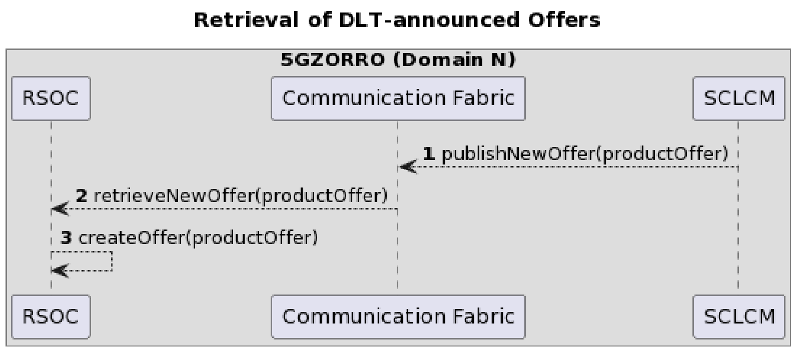
Retrieval of DLT-announced Offers
5GZORRO (Domain N)
RSOC
Communication Fabric
SCLCM
1 publishNewOffer(productOffer)
2 retrieveNewOffer(productOffer)
3 createOffer(productOffer)
RSOC
Communication Fabric
SCLCM

Figure 3-19: Retrieval of DLT-announced product offers workflow

Step 1-2: After such events are advertised (via the Communication Fabric), shared information is processed, and every peer's storage is updated.

Step 3: Considering for instance the case of new offers, data exchanged via the ledger is consumed for offer composition and storage, leaving it available for lookup at each participant instance.

While the updates about product offers are disseminated across all the Marketplace stakeholders, it should be noted that in the case of orders, related notifications are only exchanged between the parties directly involved in the negotiation due to privacy considerations. Therefore, new product orders will be only exchanged between the triggering domain (i.e., the domain acting as consumer) with the domain acting as provider of the offer being acquired.

3.2.1.3.4 Searching for Offers and Capture of Product Orders

Stakeholders acting as Marketplace customers (e.g., Communication Service Providers) will access the Catalogue in order to find an offer suitable for their needs while, for instance, designing end-to-end services on behalf of their vertical customers. To do so, the basic procedure is shown in Figure 3-20.

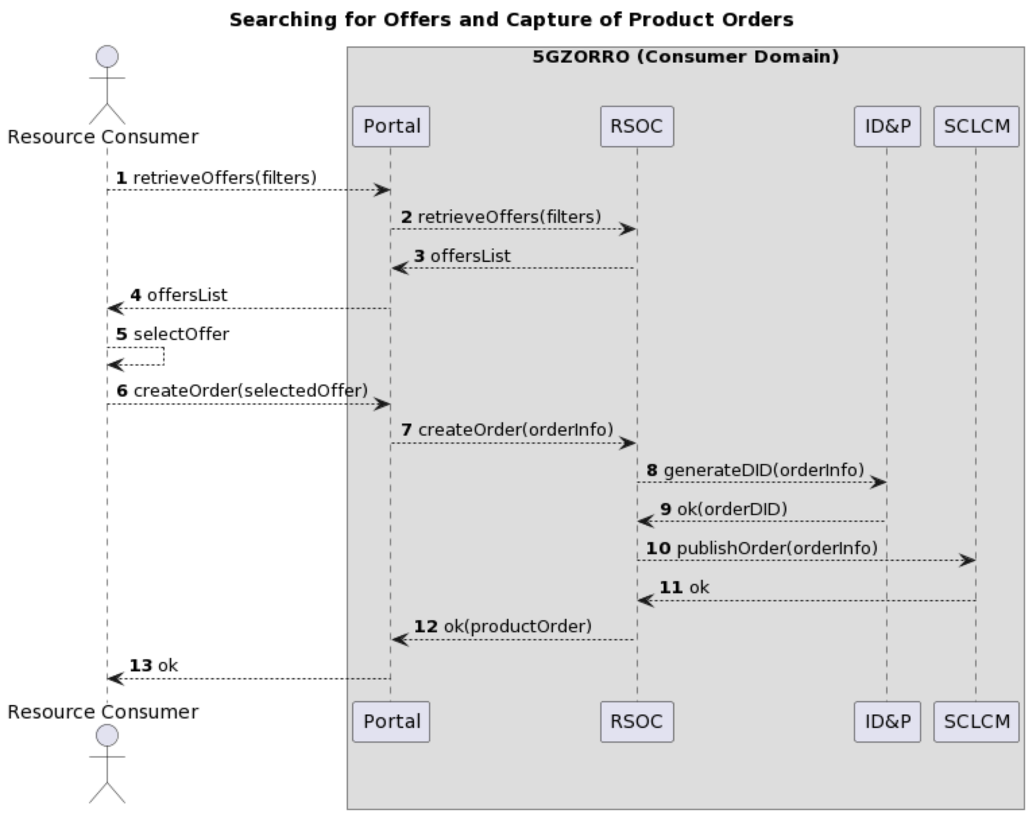

Figure 3-20: Searching for offers and capture of product orders workflow

Step 1-4: Advertised product offers can be searched by filtering them based on several criteria such as category, provider, and geographic location, among others.

Step 5-7: After performing a selection, the consumer places a request specifying the corresponding offer and additional order information (e.g., name, description, validity period, etc.).

Step 8-9: The Catalogue interacts with the Identity and Permissions (ID&P) module, to retrieve the unique distributed indentifier (DID) that is associated to every order, in order to globally indentify orders across the distributed 5GZORRO ecosystem.

Step 10-11: The order is then relayed to the SCLCM for its translation as smart contract, commitment into the ledger and transmission towards the provider domain.

Step 12-13: The confirmation of order creation is replied back to the consumer, who can proceed with the required orchestration actions.

3.2.1.4 *APIs*

To facilitate the handling of digital assets, the Resource and Service Offering Catalogue is enhanced by the use of OpenAPIs proposed by TM Forum regarding Catalogue Management APIs, for the lifecycle management and browsing of catalogue offers, as well as Ordering API, for placing and/or cancelling an order.

The list of TM Forum APIs that are adopted for the implementation of the 5GZORRO Catalogue includes:

- ***Resource Catalog Management API (TMF634)*** [7]: provides the models, dependencies, lifecycle management operations (i.e., create, update, delete, read and list) and event notification capabilities to handle entities of the Resource catalogue.
- ***Service Catalog Management API (TMF633)*** [8]: provides the models, dependencies, lifecycle management operations (i.e., create, update, delete, read and list) and event notification capabilities to handle entities of the Service catalogue.
- ***Product Catalog Management API (TMF620)*** [9]: provides the models, dependencies, lifecycle management operations (i.e., create, update, delete, read and list) and event notification capabilities to handle entities of the Product catalogue.
- ***Product Order API (TMF622)*** [10]: provides the models, dependencies, lifecycle management operations (i.e., create, update, delete, read and list) and event notification capabilities to issue (and cancel) a product order regarding some advertised offer.

3.2.2 Smart Contracts Lifecycle Manager

This module comprises a set of services for managing agreements, such as SLAs and licensing terms, together with products and spectokens both within the domain of the provider and on the 5GZORRO marketplace DLT where appropriate. A component of the broader module is a Smart Contract Lifecycle Service that manages the interactions and events between service layer and DLT, utilising ledger-centric drivers to map abstract interfaces to DLT specific functionalities. This enables the module to meet the needs of broader business-centric workflows, whilst remaining agnostic to the DLT implementation, a key objective of the architecture. Agreements, such as SLA, encapsulate TM Forum API specifications and related information models and utilise traditional local storage as necessary to allow stakeholders to manage these foundational marketplace domain-level entities. These are covered in detail in the sections that follow.

3.2.2.1 *5GZORRO Specific Enhancements*

In 5GZORRO the Smart Contract Lifecycle Manager acts as not only the gateway between the service layer and the DLT layer but also exposes functionalities for managing agreement definitions within a stakeholder's own domain. The central focus of this module is to manage lifecycle of 5GZORRO entities deployed to the underlying ledger, expose the necessary abstract interfaces for managing these on the DLT and publish associated lifecycle events for subscribing applications to consume.

A core design principle is that the 5GZORRO marketplace remains agnostic to the DLT used, although there would likely be a notion of approved ledgers. A "driver service" implements an abstract interface that defines the key operations towards the DLT, with the service encapsulating the DLT-specific logic required to interact with the ledger in question.

Product offer updates and their availability are disseminated to all trading stakeholder marketplace nodes as a result of being posted to the DLT. Product order requests are made to the Smart Contract Lifecycle Manager and subsequently the necessary related entities (such as SLAs and License Terms) are published to the DLT such that relevant parties enter into an immutable agreement over the terms of the order. From there the lifecycle of the order agreement and the events during their lifecycle are managed by the lifecycle manager. SLA violations and license term updates are handled, validated (by smart contract) and persisted to the DLT.

3.2.2.2 *Design Details*

The Smart Contract Lifecycle Manager Module is at the heart of the marketplace and key to attainment of the following KPIs:

- Ability for untrusted parties to negotiate, set-up and operate a new technical/commercial relationship via a Smart Contract for 3rd-party resource leasing/allocation with associated SLA (KPI target: Smart Contract for 3 or more untrusted parties).
- Automatically discover and "inventorize" various types of resources (i.e., compute, storage, network at core, edge, far-edge), spectrum and services capabilities from different domains and service providers (KPI target: distribution of resource updates and discovery in less than 10 mins).
- Implement/correlate technical service configurations and SLA monitoring interactions between multiple parties (KPI target: SLA measurements and validation from at least 3 operators involved in a multi-party service chain).

In order to support the functionalities described above, the internal architecture of the Smart Contract Lifecycle Module shown in Figure 3-21 below comprises the following main entities:

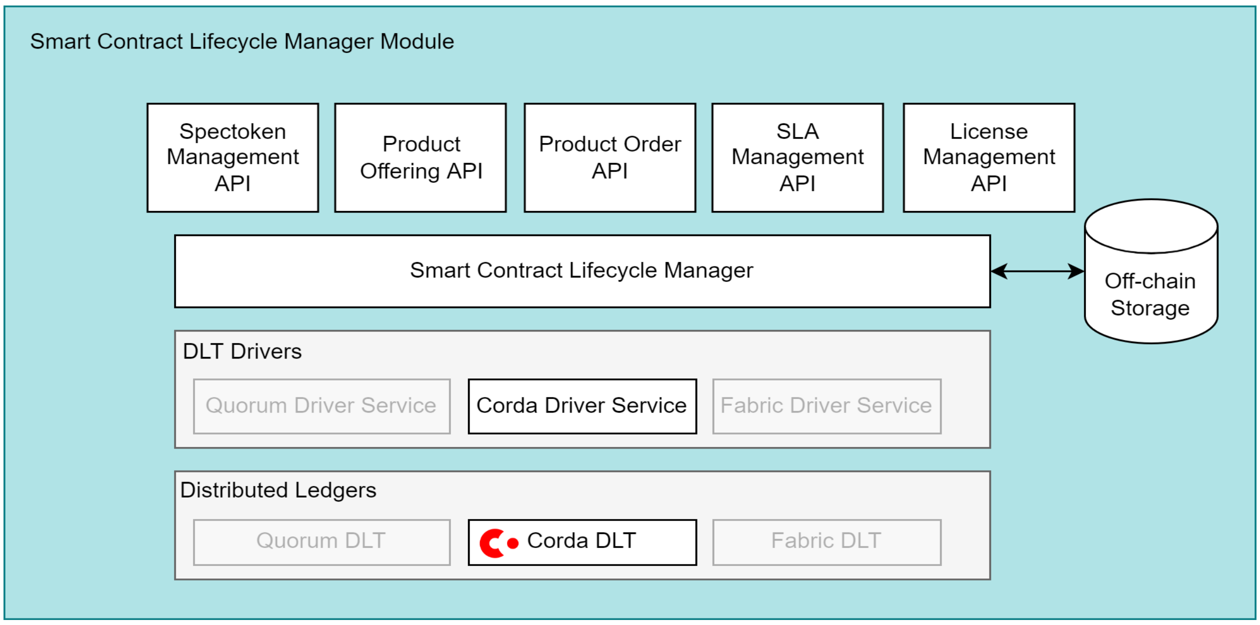

Figure 3-21: Smart Contract Lifecycle Manager Module Architecture

Product Offering API: interface implementation leveraging on the standard information models from TM Forum for offer objects as defined in TMF620 – Product catalogue management [9], that exposes CRUD capabilities pertaining to product offerings that can be used by providers to publish and manage the products they wish to offer in the catalogue. Product offers are subsequently published to the DLT and their lifecycle is managed by the Smart Contract Lifecycle Manager.

Product Order API: interface implementation leveraging on the standard information models from TM Forum for order objects as defined in TMF622 – Product Ordering Management API [10], that exposes capabilities for a consumer to purchase a product offer advertised in the catalogue. Product orders are subsequently published to the DLT and their lifecycle is managed by the Smart Contract Lifecycle Manager.

SLA Management API: implementation of TM Forum APIs as defined in TMF623 – SLA Management API [11] for providing CRUD capabilities for SLA definitions that can be associated with product orders. SLAs are created off-chain and only published to the DLT when a product order is made.

License Management API: implementation of a custom interface for providing CRUD capabilities for Licensing definitions that can be associated with product orders. License are created off-chain and only published to the DLT when a product order is published.

Spectoken Management API: implementation of a custom interface for providing CRUD capabilities for spectokens to enable spectrum trading. In 5GZORRO, spectokens are all NFTs that are persisted on the DLT.

Smart Contract Lifecycle Manager: functional entity that implements the core logic of this module. It is responsible for managing the read/write of off-chain database entries, publishing and subscribing to lifecycle events to/from the Communication Fabric for domain entities managed on the DLT and for providing the mapping capabilities from service layer to DLT using appropriate drivers.

Smart Contract Lifecycle Manager Storage: Database where SLA and License entities are persisted and queried from to support their inclusion in product offering definitions.

DLT Drivers (Corda Driver Service): for the 5GZORRO marketplace to remain DLT agnostic, an abstract interface is defined that encapsulates the high-level functions towards the DLT. A driver service implementation of this interface for a given DLT, encapsulates all DLT-specific logic to realise the underlying implementation. For 5GZORRO, an R3 Corda Driver service will be developed, thus integrating the marketplace with a CORDA DLT implementation.

Distributed Ledgers (Corda DLT): as previously stated, the 5GZORRO is to remain agnostic to the DLT, but at this time an R3 Corda implementation will be developed.

3.2.2.3 *Specific and relevant workflows*

3.2.2.3.1 SLA and License definition

Resource providers create SLA and License definitions based on templates defined in the Legal Prose Management Repository such that they can then be incorporated into product offerings and published in the catalogue. The creation of SLAs and Licenses will incorporate relevant parties and metrics and be populated from models defined in legal prose templates with a reference to the template on which the SLA/License is based upon, tying the technical definition to the human-readable prose equivalent.

3.2.2.3.1 (Primitive) Spectoken generation

With the aim to enable spectrum trading, in 5GZORRO, two spectoken types are defined, as follows:

a) Primitive spectoken: corresponds to an approved spectrum license, is generated by the regulator and is owned by the authorized Spectrum Resource Provider (SRP). As it represents a license from the regulator, it underpins all further Spectokens derived from it.

b) Derivative spectoken: corresponds to an acquired spectrum offer (i.e., via order), is associated to its "parent" primitive spectoken, is generated by the spectrum provider and is owned by the consumer of the offer.

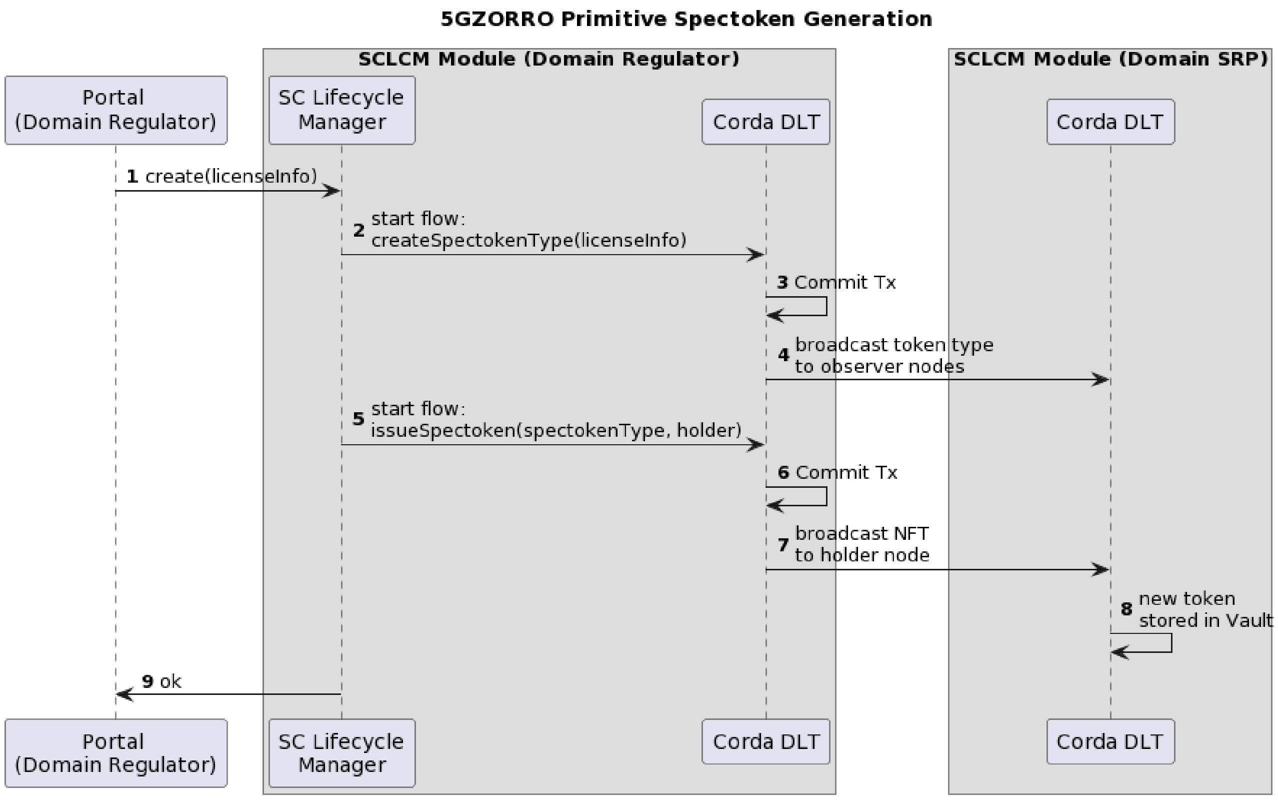
5GZORRO Primitive Spectoken Generation
SCLCM Module (Domain Regulator)
SCLCM Module (Domain SRP)
Portal (Domain Regulator)
SC Lifecycle Manager
Corda DLT
Corda DLT
1 create(licenseInfo)
2 start flow: createSpectokenType(licenseInfo)
3 Commit Tx
4 broadcast token type to observer nodes
5 start flow: issueSpectoken(spectokenType, holder)
6 Commit Tx
7 broadcast NFT to holder node
8 new token stored in Vault
9 ok
Portal (Domain Regulator)
SC Lifecycle Manager
Corda DLT
Corda DLT

Figure 3-22: Generate a Primitive Spectoken

For each type, the emission procedure on the DLT includes two steps: token type creation and NFT issuance. Figure 3-22 illustrates the generation of a primitive spectoken. In particular, based on the information included in the spectrum license (such as frequency range, area, and time), the regulator domain handles the emission (creation + issuance) of the primitive spectoken. In the former (see Steps 2-4), the new entity type is defined with its attribute, while in the later (see Steps 5-8) a concrete instance of such type is issued and stored in the DLT vault of the corresponding spectrum provider domain. A similar procedure is followed for the generation of derivative spectokens as result of the acquisition of some spectrum offer (see Figure 3-24).

3.2.2.3.2 Publish a product offering

Having assembled a product offer, the Catalogue publishes the offer to the marketplace DLT using the Smart Contract Lifecycle Manager. This in turn leverages a DLT driver to translate the request into DLT-specific implementation to realise the operation. The culmination of these operations is the verification and submission of the offer transaction to the ledger and the propagation of the catalogue update to all participants such that it can be reflected in their own copy of the catalogue. The Figure 3-23 below illustrates the steps involved.

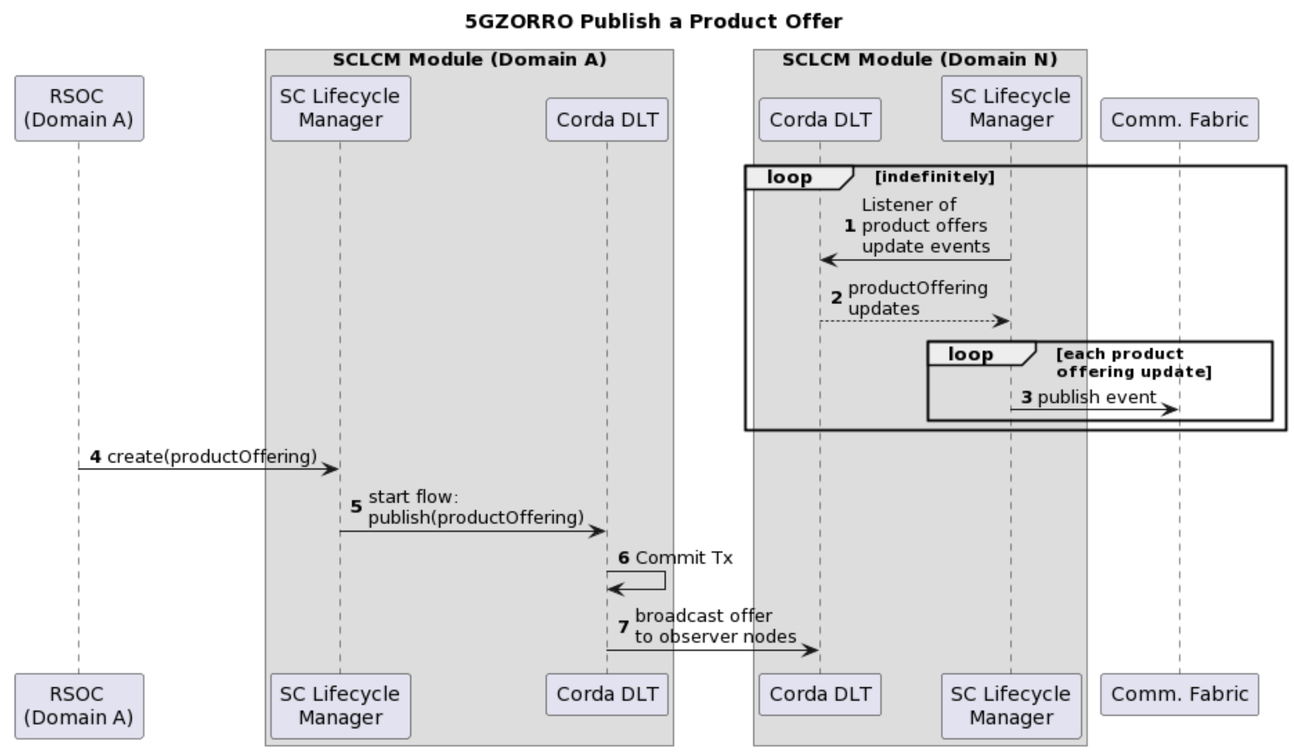
5GZORRO Publish a Product Offer
SCLCM Module (Domain A)
SCLCM Module (Domain N)
RSOC (Domain A)
SC Lifecycle Manager
Corda DLT
Corda DLT
SC Lifecycle Manager
Comm. Fabric
loop [indefinitely]
1 Listener of product offers update events
2 productOffering updates
loop [each product offering update]
3 publish event
4 create(productOffering)
5 start flow: publish(productOffering)
6 Commit Tx
7 broadcast offer to observer nodes

Figure 3-23: Publish a Product Offer

3.2.2.3.3 Product order (create agreement)

On discovering product offers that meet the needs of a consumer, they must compose a product order and submit it to the catalogue. In turn, this order is submitted to the Smart Contract Lifecycle manager to be published to the DLT; this marks the beginning of the lifecycle of the order. As per the TM forum information model specified in TMF622 [10], the order comprises of metadata, resource or service offers and SLAs. The publication of the order to the DLT makes the order state available to both consumer and provider and prompts the provider to perform checks as to the viability of servicing the order. If the provider has the capacity, they can choose to accept the order and the process of provisioning can begin. If they are unable to service the order or do not agree to the terms presented, they are able to reject the order.

If the order comprises a Spectrum offer, the regulator is named as a related party in the order and as such would have visibility of the order by being noted as an observer of the transaction, but also have rights granted to terminate the order at any point in its lifecycle should it be determined that the consumer is not permitted to hold the spectrum allocation. Additionally, the reception of a spectrum order also determines the issuance of a derivative spectoken by the resource provider to be held by the consumer in question.

The workflow steps are illustrated in Figure 3-24.

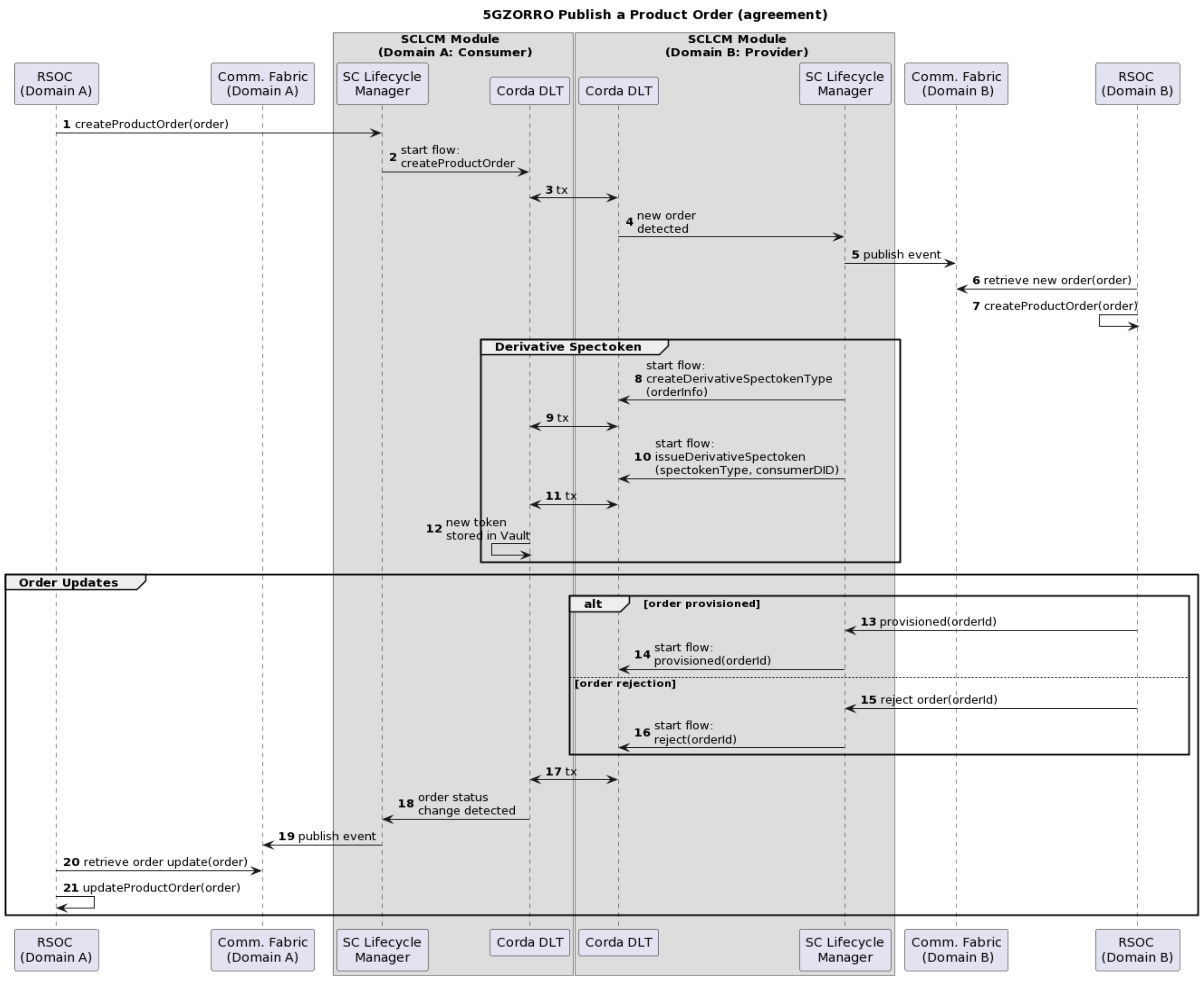
5GZORRO Publish a Product Order (agreement)
SCLCM Module (Domain A: Consumer)
SCLCM Module (Domain B: Provider)
RSOC (Domain A)
Comm. Fabric (Domain A)
SC Lifecycle Manager
Corda DLT
Corda DLT
SC Lifecycle Manager
Comm. Fabric (Domain B)
RSOC (Domain B)
1 createProductOrder(order)
2 start flow: createProductOrder
3 tx
4 new order detected
5 publish event
6 retrieve new order(order)
7 createProductOrder(order)
Derivative Spectoken
8 start flow: createDerivativeSpectokenType (orderInfo)
9 tx
10 start flow: issueDerivativeSpectoken (spectokenType, consumerDID)
11 tx
12 new token stored in Vault
Order Updates
alt [order provisioned]
13 provisioned(orderId)
14 start flow: provisioned(orderId)
[order rejection]
15 reject order(orderId)
16 start flow: reject(orderId)
17 tx
18 order status change detected
19 publish event
20 retrieve order update(order)
21 updateProductOrder(order)

Figure 3-24: Publish a Product Order Agreement

Step 1: A request is made to the Lifecyle manager to create a product order.

Step 2-3: Lifecycle Manager begins the Corda *createProductOrder* flow by utilising the DLT driver.

Step 4: Transaction is committed to the ledger and is signed by provider and consumer.

Step 5-7: Provider is notified of the new order.

Step 8-12: (optional) If the order refers to a spectrum offer, a Derivative Spectoken is issued and the transaction is signed by provider and consumer. Note that also the regulator is involved as observer in the creation step.

Step 13-16: Further order updates, such as rejecting the order or marking it as provisioned by the provider, leads to initiation of subsequent flows (e.g., reject or provisioned) via the Smart Contract Lifecycle Manager.

Step 17-21: The status of the order is updated on the DLT and the consumer is notified.

3.2.2.3.4 SLA Lifecycle Management

The Smart Contact Lifecycle Manager will monitor agreements and whenever a significant update occurs (e.g., creation or termination), events will be published for subscribing applications to react accordingly. In this case the change of an SLA's state should prompt the monitoring manager to react accordingly to configure the necessary monitoring and analytics to assess the SLAs performance. The Figure 3-25 below workflow depicts this monitoring process and the configuration of monitoring pipelines by the SLA Monitoring Manager.

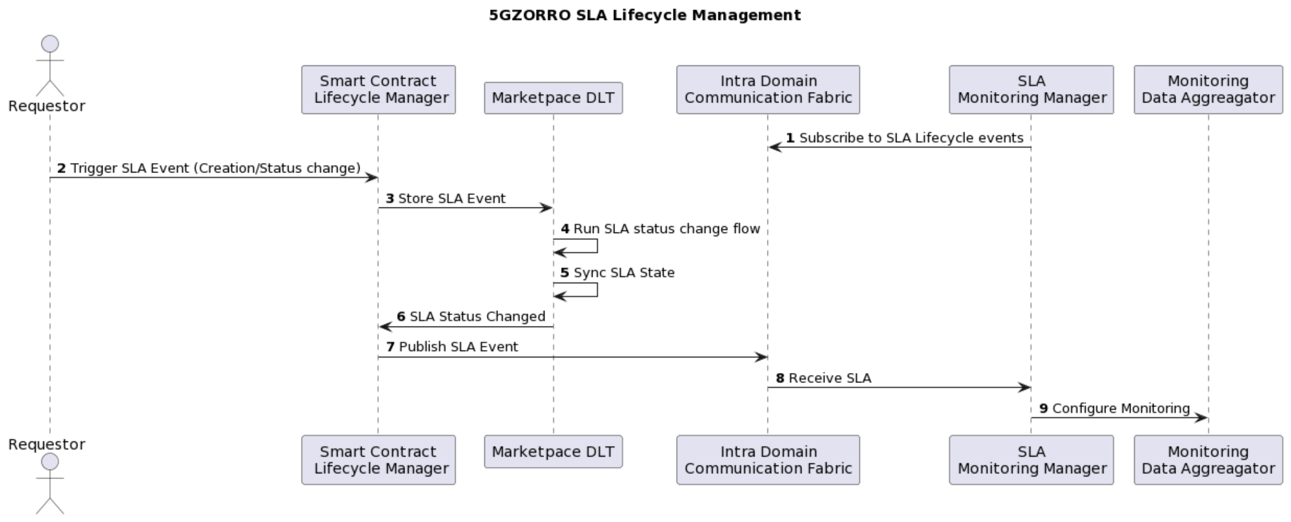
5GZORRO SLA Lifecycle Management
Requestor
Smart Contract Lifecycle Manager
Marketpace DLT
Intra Domain Communication Fabric
SLA Monitoring Manager
Monitoring Data Aggreagator
1 Subscribe to SLA Lifecycle events
2 Trigger SLA Event (Creation/Status change)
3 Store SLA Event
4 Run SLA status change flow
5 Sync SLA State
6 SLA Status Changed
7 Publish SLA Event
8 Receive SLA
9 Configure Monitoring

Figure 3-25: SLA Lifecycle Management

Step 1: The SLA Monitoring Manager registers a listener for SLA Lifecycle events (Created, Activated, Updated, Terminated).

Step 2: Smart Contract Lifecycle Manager receives SLA events from a Requestor. The Requestor is any modules triggers an SLA Creation or status change (e.g., RSOC, ISBP, etc).

Step 3-6: Lifecycle Manager stores SLA Events in Marketplace DLT (e.g., new SLA, updated SLA). This triggers the execution of a specific flow in the markeplave DLT and the syncronisation of DLT nodes

Step 7-8: The Smart Contract Lifecycle Manager publish the SLA Event through the Communication Fabric. Monitoring Manager receives changes to SLAs that are to be monitored.

Step 9: SLA Monitoring Manager configures the Monitoring Data Aggregator according to the SLAs it is required to monitor.

Note: Create/update should configure the Monitoring Data Aggregator according to the SLA type, whereas termination should teardown the aggregator. Once SLAs become active and the monitoring/analytics pipeline has been configured, monitoring data is pushed to the Data Lake for analysis as further explained in Section 3.3.2.

3.2.2.4 *APIs*

To facilitate the management of SLA definitions, License definitions, spectokens, product offerings and product orders underpinned by the DLT, this module offers APIs for these management activities and the associated lifecycle events.

The following APIs will be adopted for the implementation:

- **Product Offer API (TMF620) [**9**]:** provides the models, dependencies, lifecycle management operations (i.e., create, update, delete, read and list) and event notification capabilities to handle Product Offers.
- **Product Order API (TMF622)** [10]: provides the models, dependencies, lifecycle management operations (i.e., create, update, delete, read and list) and event notification capabilities to issue (and cancel) a product order regarding some advertised offer.

- **SLA Management API (TMF623)** [11]**:** provides the models, dependencies, lifecycle management operations (i.e., create, update, delete, read and list) and event notification capabilities to handle entities pertaining to SLA management.
- **License Management API (Custom):** provides the models, dependencies, lifecycle management operations (i.e., create, update, delete, read and list) capabilities to handle entities related to License management.
- **Spectoken Management API (Custom):** provides the models, dependencies, lifecycle management operations (i.e., create, update, delete, read and list) capabilities to handle entities related to Spectoken management.

3.2.2.4.1 DLT Driver API definition

An abstract interface implemented by DLT-specific drivers to provide the integration point between the Smart Contract Lifecycle Manager and DLTs will be developed. Below is the definition of the interface it exposes.

Product Offerings APIs

Operation name: **publishProductOffering**		
Description	API Endpoint for a provider to publish a new product offer to the DLT	
Input Parameters	**Type**	**Description**
publishOfferRequest	PublishProductOfferingRequest	Custom model incapsulating the TMF Product Offering to be published and its referred TMF Product Offering Prices, TMF Product Specification, TMF Resource Specifications, TMF Service Specifications and TMF Geographic Addresses; this custom model also includes the DID associated to the Product Offering to be published
Output Parameters	**Type**	**Description**
N/A		

Operation name: **updateProductOffering**		
Description	API Endpoint for a provider to update one of their product offerings on the DLT and subsequently be announced to all marketplace trading stakeholders	
Input Parameters	**Type**	**Description**
updateOfferRequest	UpdateProductOfferingRequest	Custom model incapsulating the TMF Product Offering to be published and its referred TMF Product Offering Prices, TMF Product Specification, TMF Resource Specifications, TMF Service Specifications and TMF Geographic Addresses; this custom model also includes the DID associated to the Product Offering to be published
Output Parameters	**Type**	**Description**
N/A		

Operation name: **removeProductOffering**	
Description	API Endpoint for a provider to retire a product offering from the marketplace catalogue by marking it as such on the DLT

Input Parameters	**Type**	**Description**
offerId	String	DID of the product offering
Output Parameters	**Type**	**Description**

Product Orders APIs

Operation name: **publishProductOrder**		
Description	API Endpoint for a consumer to create a new product order and begin the associated lifecycle	
Input Parameters	**Type**	**Description**
publishOrderRequest	PublishProductOrderRequest	Custom model incapsulating the TMF Product Order to be published, its associated DID, the DID of the supplier of the offer related to the Product Order and the time period that define the validity period of the encapsulated Order.
Output Parameters	**Type**	**Description**
N/A		

Operation name: **approveProductOrder**		
Description	API Endpoint for a provider to accept a product order request and mark it as such on the DLT	
Input Parameters	**Type**	**Description**
orderId	String	DID of the product order
Output Parameters	**Type**	**Description**
N/A		

Operation name: **rejectProductOrder**		
Description	API Endpoint for a provider to reject a product order request	
Input Parameters	**Type**	**Description**
orderId	String	DID of the product order
rejectionReason	String	The reason for rejecting the order e.g., no capacity
Output Parameters	**Type**	**Description**
N/A		

Operation name: **changeProductOrder**		
Description	API Endpoint for a provider or consumer to submit a product order update proposal	
Input Parameters	**Type**	**Description**
orderId	String	DID of the product order
request	ChangeProductOrderRequest	Custom model incapsulating the TMF Product Order to be updated, its associated DID, the DID of the supplier of the offer related to the Product Order and the time period that define

			the validity period of the encapsulated Order.
Output Parameters		**Type**	**Description**
	N/A		

Operation name: **acceptChangeProductOrder**			
Description		API Endpoint for a counterparty to accept a proposed change to a product order	
Input Parameters		**Type**	**Description**
	orderId	String	DID of the product order to accept proposed changes on
Output Parameters		**Type**	**Description**
	N/A		

Operation name: **rejectChangedProductOrder**			
Description		API Endpoint for a counterparty to reject a proposed change to a product order	
Input Parameters		**Type**	**Description**
	orderId	String	DID of the product order to reject proposed changes
Output Parameters		**Type**	**Description**
	N/A		

Operation name: **endProductOrder**			
Description		API Endpoint for a consumer or provider to terminate an existing product order	
Input Parameters		**Type**	**Description**
	orderId	String	DID of the product order to terminate
Output Parameters		**Type**	**Description**
Notes			
Operation name: **ProvisionProductOrderFlow**			
Description		API Endpoint to update the DLT once all product offerings on an order have been provisioned and as such, the agreement is then live	
Input Parameters		**Type**	**Description**
	orderId	String	DID of the product order to update
Output Parameters		**Type**	**Description**
Notes			

SLA APIs

Operation name: **createServiceLevelAgreement**			
Description		API Endpoint for a consumer to create a new Service Level Agreement following the TMF SLA Information Model	
Input Parameters		**Type**	**Description**
	sla	ServiceLevelAgreement	Service Level Agreement following the information model specified by the TMF

Output Parameters	Type	Description
SLA ID	String	ID associated to the created SLA
Notes		
The ID associaed to the created Service Level Agreement will be replaced by the DID released by the Identity and Permission Manager. The DID will be requested after the Service Level Agreement has been stored locally on the Smart Contract Lifecycle Manager Storage.		

Operation name: **removeServiceLevelAgreement**		
Description	API Endpoint for a consumer to remove a Service Level Agreement	
Input Parameters	**Type**	**Description**
did	String	DID associated to the Service Level Agreement to be removed
Output Parameters	**Type**	**Description**
N/A		

Operation name: **updateTemplateIdentity**		
Description	API Endpoint to handle the response of the Identity & Permission Manager containing the DID to be associated to a Service Level Agreement	
Input Parameters	**Type**	**Description**
state	DIDStateCSDto	Information model that encapsulates the response of the Identity & Permission Manager containing the DID
Output Parameters	**Type**	**Description**
N/A		

Operation name: **updateSLAState**		
Description	API Endpoint to trigger the update of the Service Level Agreement in the DLT	
Input Parameters	**Type**	**Description**
request	UpdateSLAStateRequest	Custom model incapsulating the DID of the Product Order that include the Service Level Agreement to be updated, the DID of the Service Level Agreement to be updated and the new state (VIOLATED or RETIRED) of the Service Level Agreement
Output Parameters	**Type**	**Description**
N/A		

Operation name: **getServiceLevelAgreement**		
Description	API Endpoint for a consumer to retrieve a specific Service Level Agreement by ID	
Input Parameters	**Type**	**Description**
identifier	String	ID or DID of the Service Level Agreement to be retrieved
Output Parameters	**Type**	**Description**
sla	ServiceLevelAgreement	Service Level Agreement retrieved

Operation name: **getServiceLevelAgreements**

Description	API Endpoint for a consumer to retrieve Service Level Agreements	
Input Parameters	**Type**	**Description**
pageable	Pageable	A *page* can be specified to retrieve a subset of the total of the Service Level Agreements stored in the Smart Contract Lifecycle Manager Storage
Output Parameters	**Type**	**Description**
slas	PagedSlaResponse	Paged response containing the Service Level Agreements

License Terms APIs

Operation name: **createLicense**		
Description	API Endpoint for a consumer to create a new License	
Input Parameters	**Type**	**Description**
license	License	License specification following the license Information Model
Output Parameters	**Type**	**Description**
License ID	String	ID associated to the created License
Notes		
The ID associaed to the created License will be replaced by the DID released by the Identity and Permission Manager. The DID will be requested after License has been stored locally on the Smart Contract Lifecycle Manager Storage.		

Operation name: **removeLicense**		
Description	API Endpoint for a consumer to remove a License	
Input Parameters	**Type**	**Description**
did	String	DID associated to the License to be removed
Output Parameters	**Type**	**Description**
N/A		

Operation name: **updateLicenseIdentity**		
Description	API Endpoint to handle the response of the Identity & Permission Manager containing the DID to be associated to a License	
Input Parameters	**Type**	**Description**
state	DIDStateCSDto	Information model that encapsulates the response of the Identity & Permission Manager containing the DID
Output Parameters	**Type**	**Description**
N/A		

Operation name: **updateLicenseState**		
Description	API Endpoint to trigger the update of the License in the DLT	
Input Parameters	**Type**	**Description**
request	UpdateLicenseStateRequest	Custom model incapsulating the DID of the Product Order that include License to be updated, the DID of the License to be updated and the new state (VIOLATED or RETIRED) of the License
Output Parameters	**Type**	**Description**

	N/A		

Operation name: **getLicense**			
Description		API Endpoint for a consumer to retrieve a specific License by ID	
Input Parameters		**Type**	**Description**
	identifier	String	ID or DID of the License to be retrieved
Output Parameters		**Type**	**Description**
	license	License	License retrieved

Operation name: **getLicenses**			
Description		API Endpoint for a consumer to retrieve Licenses	
Input Parameters		**Type**	**Description**
	pageable	Pageable	A *page* can be specified to retrieve a subset of the total of the Licenses stored in the Smart Contract Lifecycle Manager Storage
Output Parameters		**Type**	**Description**
	licenses	PagedLicenseResponse	Paged response containing the Licenses

Spectoken APIs

Operation name: **createPrimitiveSpectoken**			
Description		API Endpoint for a regulator to create a new primitive spectoken on behalf of the entitled SRP. NB. Derivative spectokens are handled as part of the internal logic of the Smart Contract Lifecycle manager. The API to create them is the one for product ordering in the case spectrum resources are included.	
Input Parameters		**Type**	**Description**
	spectrumLicenseInfo	primitiveSpectokenRequest	Custom model incapsulating the details of the spectrum license for which a Primitive Spectoken is to be generated. This includes its associated license, the DID of the CSP, and characteristics of the encapsulated spectrum license (such as country, startTime, endTime, startDL, endDL, startUL, endUL, duplexMode, band, technology).
Output Parameters		**Type**	**Description**
	-	-	-

Operation name: **removePrimitiveSpectoken**			
Description		API Endpoint for a regulator to remove a primitive spectoken	
Input Parameters		**Type**	**Description**
	identifier	UID	ID associated to the spectoken to be removed
Output Parameters		**Type**	**Description**

Operation name: **getPrimitiveSpectoken**			
Description		API Endpoint for an SRP to retrieve a specific primitive spectoken	
Input Parameters		**Type**	**Description**
	identifier	UID	ID associated to the spectoken to be retrieved
Output Parameters		**Type**	**Description**
	primitiveSpectoken	Object	Primitive spectoken object (see Section 4.2.5)

Operation name: **listPrimitiveSpectokens**			
Description		API Endpoint for an SRP to retrieve all owned primitive spectokens	
Input Parameters		**Type**	**Description**
	-	-	-
Output Parameters		**Type**	**Description**
	primitiveSpectokensList	List of Objects	List of primitive spectoken objects (see Section 4.2.5)

3.2.3 Marketplace portal

This portal is the module that comprises the unique 5GZORRO's web GUI: Governance (Section 3.1.4) and Marketplace portal. As mentioned in D2.4 [1], this component allows 5GZORRO's stakeholders to submit and browse the different offers to be available in the marketplace - which, in turn, is managed by the Catalogue Manager component - and the placing and reviewing of related orders, having in mind a particular set of business terms. For this reason, when allowing stakeholders to onboard, fetch and order Marketplace products, the GUI must also interface directly with both the Resource and Service Offering Catalogue (Section 3.2.1) and the Smart Contract Lifecycle Manager (Section 3.2.2). Furthermore, this GUI interacts also directly with the Smart Resource and Service Discovery (Section 3.3.3), allowing stakeholders to intelligently query and retrieve catalogue offers based on high-level intents, abstracting the underlying complex ML-based offers classification and intent recognition.

3.2.3.1 *5GZORRO Specific Enhancements*

While other web portals facilitate already 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, (iii) by extending this marketplace following a Governance model which is clear and transparent for all stakeholders involved (acceptance of placed orders under specific business terms, allowing usage of spectrum by a real licensed Regulator, etc.) and, finally, (iv) the alignment with other initiatives such as TM Forum's set of APIs for a Telecom Marketplace, by enhancing it and extending it to a user-friendly web application.

3.2.3.2 *Marketplace Portal Design Details*

Following the same approach as in the Governance portal (Section 3.1.4), the Table 3-3 below indicates the User Stories that drive the implementation of this 5GZORRO module (Web GUI).

Table 3-3: Definition of Marketplace Portal's User Stories

US id	**User Story** (US)	**SW Module & Interface/Operation** (Interface)		**Inputs and notes**	**Outputs**
US1	As a Resource Provider, I must be able to on-board resources into the Catalogue	Module	Any Resource Manager and Resource and Service Offering Catalogue	Resource categories to be fetched from xRM: • Edge • Cloud • Spectrum • RAN • VNF	Resource Specification created at the Catalogue
		Operation	Fetch available assets from xRM and Create resource method at the Catalogue		
US2	As a Service Provider, I must be able to on-board services into the Catalogue	Module	Any Resource Manager and Resource and Service Offering Catalogue	Serice categories to be fetched from xRM: • Network Slice • Network Service	Service Specification created at the Catalogue
		Operation	Fetch available assets from xRM and Create service method at the Catalogue		
US3	As a Resource and Service Provider, I must be able to create customized SLAs and License Terms	Module	Legal Prose Repository and SC Lifecyle Manager	Template and user provided parameters for SLA / License Terms customized definition	SLA and License Term created at the SCLCM
		Operation	Fetch available templates from LPR and Create SLA / License Term method at SCLCM		
US4	As a Resource and Service Provider, I must be able to create a Product Offering Price with attached License Terms	Module	SC Lifecyle Manager and Resource and Service Offering Catalogue	Product Offering Prices can optionally contain attached license terms for license SW related offers (VNF/CNF)	Product Offering Price created at the Catalogue
		Operation	Fetch available license trens from SCLCM and Create price method at the Catalogue		
US5	As a Resource and Service Provider, I must be able to compose a Product Offering and publish it to the Marketplace	Module	SC Lifecyle Manager and Resource and Service Offering Catalogue	When creating a Product Offer, it is expected that the user: a) selects the service or resource; b) supplies basic data (name, description, location, validity period); c) selects a price; d) selects customized SLA definition	Product Offer
		Operation	Fetch SLA from SCLCM and Publish Product Offer method at Catalogue		
US6	As a Resource and Service Provider, I must be able to	Module	Resource and Service Offering Catalogue	Modified Product Offering	Updated Product Offer

US id	**User Story** (US)	**SW Module & Interface/Operation** (Interface)		**Inputs and notes**	**Outputs**
	update a previously published Product Offer	Operation	updateProductOffer()		
US7	As a Resource and Service Consumer, I must be able to browse the most suitable available offerings, based on specific criteria	Module	Resource and Service Offering Catalogue or Smart Resource and Service Discovery (SRSD) application	Discovery Criteria (category, location, price, preference for provider)	List of Offers
		Operation	getOffers(filters) method at the Catalogue or discoverOffers() method at the SRSD		
US8	As a Resource and Service Consumer, I must be able to place an order on a particular Product Offering	Module	Resource and Service Offering Catalogue	Product Order IM	Product Order
		Operation	createProductOrder()		
US9	As a Resource and Service Consumer, I must be able to browse all my active Product Orders	Module	Resource and Service Offering Catalogue		List of Product Orders
		Operation	getProductOrders()		
US10	As a Provider, I must be able to reject a Product Order Request	Module	Resource and Service Offering Catalogue	• productOrderId • rejectionReason	
		Operation	rejectProductOrder()		
US11	As a Regulator, I must be able to browse all submitted Product Orders which reference a Product Offering where spectrum is included as a resource	Module	Resource and Service Offering Catalogue	Use the allowed parameters of this method to filter out Product Orders referencing a Product Offer which has spectrum as resources	List of Product Orders
		Operation	getProductOrders()		
US12	As a Regulator, I must be able to terminate any order whose	Module	Resource and Service Offering Catalogue	If a Product Offer has a resource type of spectrum, the termination of a	

US id	User Story (US)	SW Module & Interface/Operation (Interface)		Inputs and notes	Outputs
	Product Offering contains the spectrum as an asset	Operation	terminateProductOrder()	subsequent Product Order can come not only from the Provider, but also from the Regulator. • productOrderId	

As some Information Models are quite extensive, it is foreseeable that, in some cases, the Stakeholder interacting with such GUI may have to copy and paste full-blown JSON representation of data, providing its complexity to fill each parameter manually. For these cases, a JSON format validator will be added to the web GUI, for a smoother user experience (by detecting if malformatted, before allowing the publishing of such data).

3.3 Cross-domain Analytics & Intelligence for AIOps Services

As described in D2.4 [1], 5GZORRO Cross-domain Analytics & Intelligence for AIOps Platform (see Figure 3-1) is responsible for realizing the envisioned AIOps for AI-driven operation of 5G network resources, slices, and services.

Cross-domain Analytics & Intelligence for AIOps platform is realised as a set of software modules that provide analytics and intelligence services to facilitate data-driven zero-touch automation throughout the 5GZORRO platform. These services are presented in Figure 3-26 and are covered in the following subsections: the Monitoring Data Aggregator Service in Section 3.1.13.2.13.3.1, the Intelligent SLA Monitoring and Breach Predictor Service in Section 3.3.23.1.23.2.2, and the Smart Resource and Service Discovery Service in Section 3.3.3.

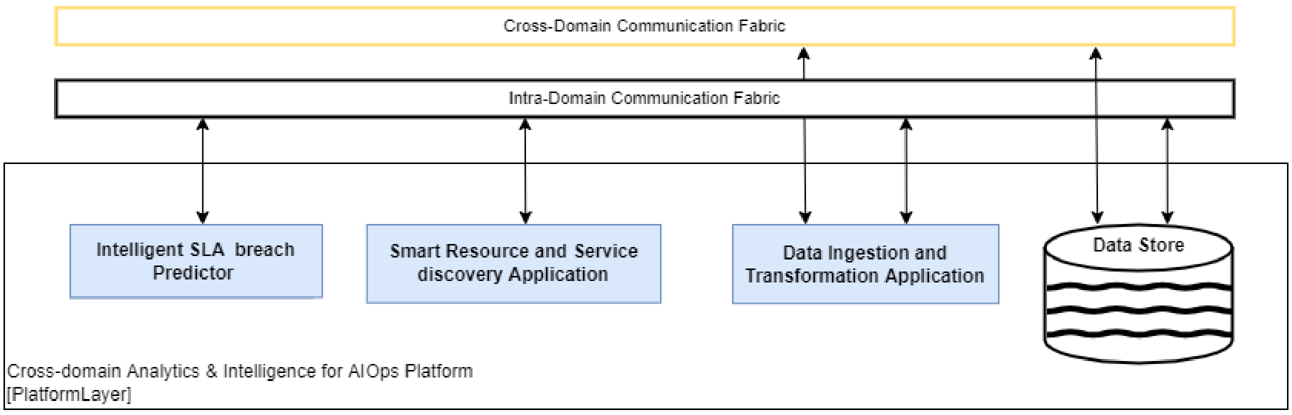
Cross-Domain Communication Fabric
Intra-Domain Communication Fabric
Intelligent SLA breach Predictor
Smart Resource and Service discovery Application
Data Ingestion and Transformation Application
Data Store
Cross-domain Analytics & Intelligence for AIOps Platform [PlatformLayer]

Figure 3-26: Cross-domain Analytics & Intelligence for AIOps platform

All the services are implemented on top of 5GZORRO Operational Data Lake platform created to ease development and deployment of such services as well as to govern and protect all the operational data of the 5GZORRO platform, facilitating multi-provider data sharing cross-platform analytics. However, 5GZORRO allows defining similar pipelines for additional analytics workflows as described in Section 2.2.5 of this document.

3.3.1 Monitoring Data Aggregator

The Monitoring Data Aggregator (MDA) is a new 5GZORRO functional block that has emerged when it was decided to have all related SLA monitoring intelligence functionalities in a single functional block by merging

"Service & Resource Monitoring" into "Intelligent SLA breach predictor". Thus, all monitoring data from several types of resources that needs to be aggregated and pushed into the Data Store will be handled by MDA.

Monitoring data is provided by each Resource and Service Provider for the resources and services controlled by that Operator. This data is stored in the Data Lake and is used to keep track of resource usage and to predict/track SLA violations. As monitoring data is provided over time, it is aggregated and made available in a suitable manner to perform the desired analytics.

3.3.1.1 *5GZORRO Specific and relevant workflow(s)*

The main workflow we have in mind is in Figure 2-11, depicting Resource Monitoring and SLA breach detection/prediction. Monitoring data is collected for each relevant resource and service. The data is aggregated over time and is used to detect or predict possible SLA violation. The interfaces and software components must provide the ability to match the metrics from a provided resource to the SLA it supports.

3.3.1.2 *MDA Design Details*

The MDA pipeline is presented in Figure 3-27.

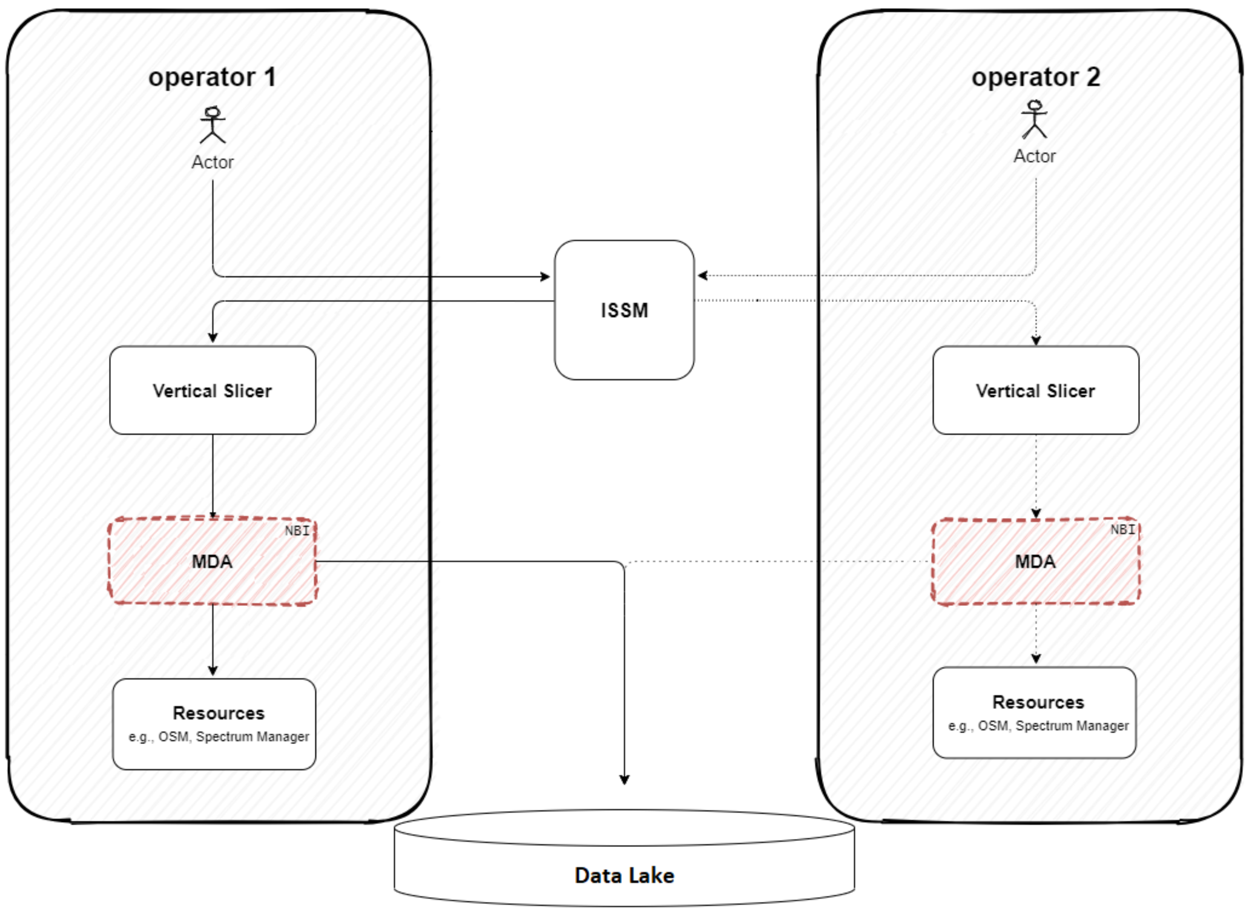

Figure 3-27: Monitoring Data Aggregator Architecture

ISSM receives a new product offer instantiation request from actor, then it requests the resource instantiation from the VS. The VS, then, sends a configuration with dynamic variables to MDA. As soon as it receives it, MDA fetches metric values from OSM, then aggregates these metrics, and signs the data with a key from the Operator. Finally, MDA posts the data into the Data Lake's Kafka topic provided by the configuration.

A particular resource may be used for one service at one time and for another service at another time. Each of these occurrences is called epoch. There may be a need to differentiate the resource metrics for the different epochs and to not aggregate metrics across epochs. In this case, a resource should be given a different resourceID for each epoch, so that the Monitoring Data Aggregator component treats them as different resources.

3.3.1.3 *MDA APIs*

Operation name: **postMonitoringData**		
Description	Operator provides monitoring data, which is saved in the Data Store and which is also forwarded to the aggregator.	
Input Parameters	**Type**	**Description**
OperatorID	string	Identity of Operator providing the data.
MonitoringData	json	Structure of monitoring data json described in 4.6.6.
StorageLocation	url	Bucket in Data Store to place the data; url provided by Data Lake in registerOperator().
DataHash	string	Hash of provided monitoring data using private key of the Operator.
Output Parameters	**Type**	**Description**
N/A		

Operation name: **aggregateMonitoringData**		
Description	Process accumulated monitoring data and aggregate into a form that is consumable by other components (e.g., SLA monitor)	
Input Parameters	**Type**	**Description**
OperatorID	string	Identity of Operator providing the data.
MonitoringData	json	Structure of monitoring data json described in 4.6.6.
StorageLocation	url	Bucket in Data Store to place the aggregated data.
Output Parameters	**Type**	**Description**
N/A		

Operation name: **startAggregateMetric**		
Description	Specify a particular metric that we want to aggregate.	
Input Parameters	**Type**	**Description**
metricName	string	Name of metric being aggregated
metricType	string	Data type of the metric
aggregationMethod	string	May be: sum, average, (other?)
Output Parameters	**Type**	**Description**
N/A		
Notes		
Additional information, such as operatorID or time interval, could be specified in future version of this API		

Operation name: **stopAggregateMetric**		
Description	Specify a particular metric that we want to stop to aggregate.	
Input Parameters	**Type**	**Description**

	metricName	string	Name of metric to stop being aggregated
Output Parameters		**Type**	**Description**
	N/A		

Operation name: **lookupMetricsByReference**			
Description		Obtain list of aggregated metrics that pertain to specified *productID*.	
Input Parameters		**Type**	**Description**
	productID	string	Reference (SLA) ID for which we want to receive list of aggregated metrics.
Output Parameters		**Type**	**Description**
	aggregatedMetrics	json	List of metrics found for *productID*.

3.3.2 Intelligent SLA Monitoring and Breach Prediction

The AIOps cycle that is followed towards the SLAs Closed-loop automation is depicted in Figure 3-28. A feedback loop starts from (1) “observation”, i.e., the gathering of monitoring data about managed resources and services as provided by the xRM; (2) it then passes through “orientation”, i.e., SLA Monitoring and Breach Prediction components where data is aggregated and formalized into SLA monitoring metrics, as well as into analysed information, notifications and alerts; (3) it subsequently passes through “decision-making” where intelligent agents receive actionable insights and notifications in order to assess the current operational status and to decide how to handle detected or forecasted issues and anomalies; (4) and finally completes with “action”, i.e., the control elements for Network Slice and Service Orchestration over the managed resources and services, before commencing again, in a continuous manner. Optionally, lower-level closed loops can also take place within domains, handled by intra-domain MANO components.

All data-plane interactions between components take place via event-based, asynchronous interactions through the *Integration Fabric*, which exposes data publication and subscription services, as well as services for the installation of data pipelines. The latter are executed in the analytics and processing engine of the Data Storage and Aggregation component, which is also responsible for persisting data, acting as the *Knowledge* Base of the feedback loop.

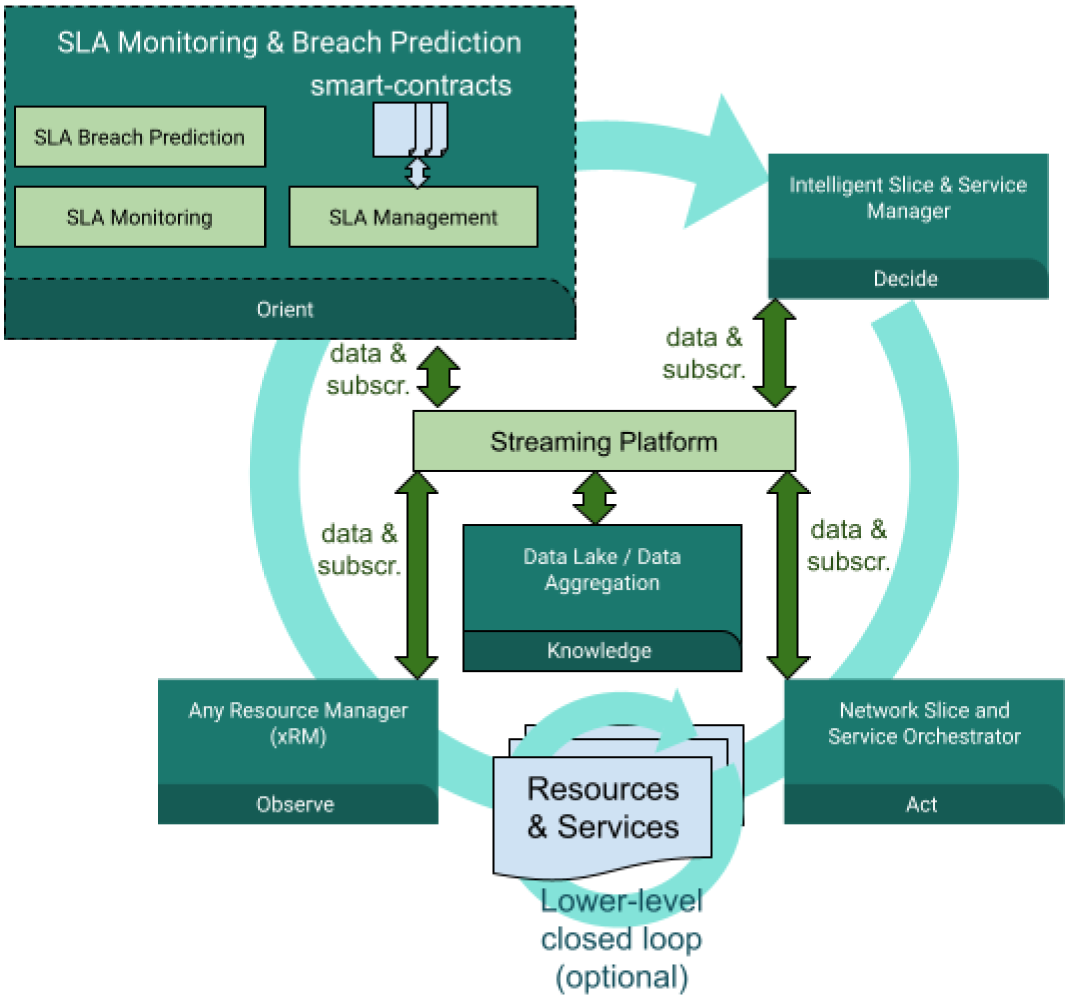
SLA Monitoring & Breach Prediction
smart-contracts
SLA Breach Prediction
SLA Monitoring
SLA Management
Orient
Intelligent Slice & Service Manager
Decide
data & subscr.
data & subscr.
Streaming Platform
data & subscr.
data & subscr.
Data Lake / Data Aggregation
Knowledge
Any Resource Manager (xRM)
Observe
Resources & Services
Network Slice and Service Orchestrator
Act
Lower-level closed loop (optional)

Figure 3-28: SLA Monitoring & Breach Prediction inter-communication Architecture

Delving deeper into the SLA Monitoring and SLA Breach Prediction components, both collect and analyse resource monitoring data from the Data Lake. While the SLA Monitoring uses this data to detect SLA violations in real-time, SLA Breach Prediction goes one step further and predicts violations before their atual occurrence by making use of ML techniques. Upon occurring SLA violations, countermeasures can be proactively taken to maintain the desired QoS for an offered service. The services that are offered by the module were described in deliverable D.2.4. Each described service includes a list of interfaces in terms of input and output parameters that are explained below.

3.3.2.1 *5GZORRO Specific enhancements*

A high-level design proposed for both SLA monitoring as well as breach prediction services is shown in Figure 3-29. Currently, the design is based on the ETSI ZSM [6] architecture and involves the Data Lake as well as the Smart Contracts Lifecycle Manager modules.

Figure 3-29 was made with the assumption that the Data Lake module is centralized and cross-domain and is not distributed for each domain. In addition, the "Subscribe" arrow includes both the triggering for establishment/configuration of data pipelines (which is subsequently handled by relevant module(s)) and the subscription to related communication and events through the Integration Fabric.

The KPI that is linked to this module is the following:

- Ability for untrusted parties to negotiate, set-up and operate a new technical/commercial relationship via a Smart Contract for 3rd-party resource leasing/allocation with associated SLA (*KPI target: Smart Contract for 3 or more untrusted parties*)
- Implement/correlate technical service configurations and SLA monitoring interactions between multiple parties. The target for this KPI is:
    - *SLA measurements and validation from at least 3 operators involved in a multi-party service chain*)

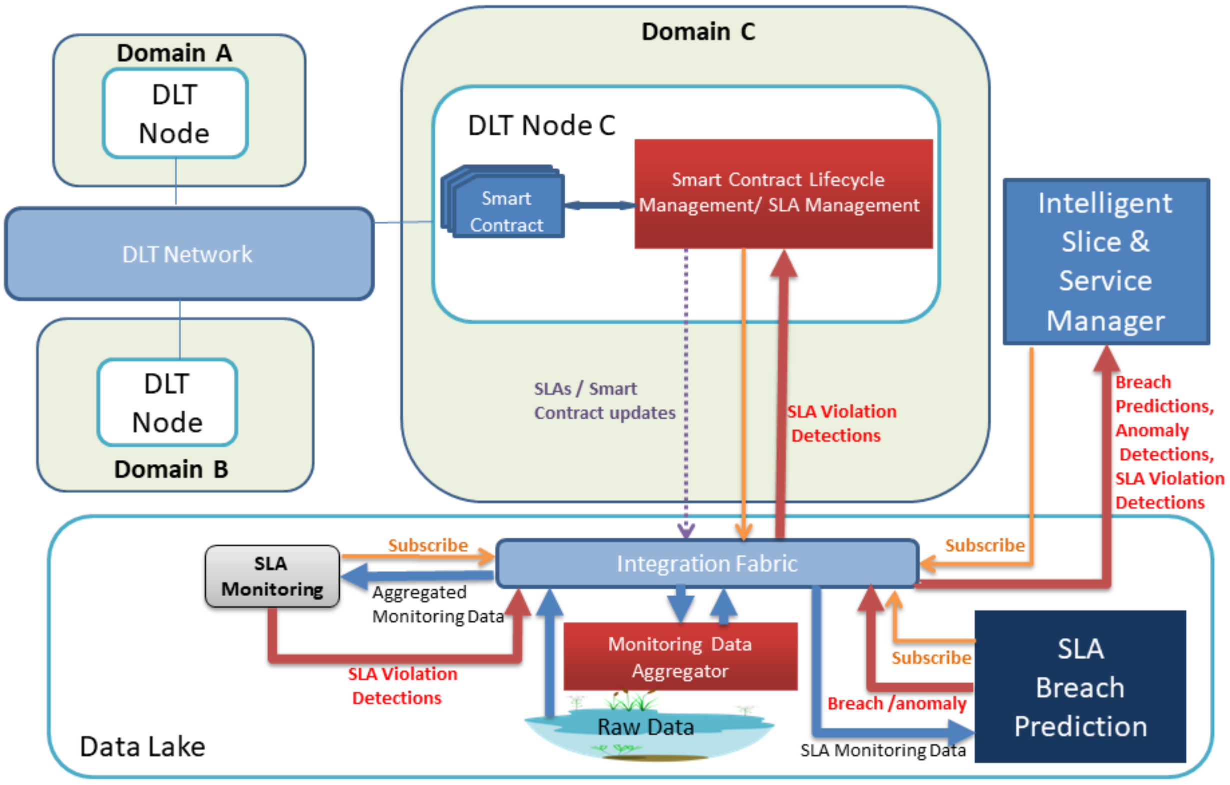

Figure 3-29: SLA Monitoring and Breach Prediction Architecture

3.3.2.2 *Specific and relevant workflow(s)*

The SLA Monitoring specific workflow and respective sequence of actions take place after step 9 described in Figure 3-25 of section 3.2.2.3.4 and can be described as follows:

Step 1: An SLA event (regarding the SLA to be monitored) is pushed to a Data Lake queue, which is then received by SLA monitoring.

Step 2: The Monitoring Data Aggregator (MDA) collects and aggregates monitoring data and pushes it to a Data Lake queue.

Step 3: The Data Lake streams the monitoring data to SLA Monitoring through a queue.

Steps 4: Analysis is performed on the aggregated data temporally using a function derived from the Legal Prose Template and present in the SLA in order to detect violations.

Step 5: If an SLA violation is detected, an event is emitted by the analysis function and the event is published to the communication fabric for subscribing apps to consume. The SLA violation published follows the SLA violation resource from the SLA Management API TMF623 [11].

It's worth mentioning that although some functions performed by SLA Monitoring were described in the past as part of the "Service & Resource Monitoring" module, this module is now deprecated. In its place MDA component has been created (encompassing all the aggregate monitoring data functionalities) and all the SLA monitoring functionalities were merged into SLA Monitoring functional block.

Going further in above mentioned steps, we can roughly divide the SLA Monitoring into two phases. The first one being the **preparation phase** (after step 1) which is used to define what SLAs are being monitored (Figure 3-30). 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 RSOC (TMF Catalog in the figure) and consequently the SLA from the SCLCM. Additionally, the component subscribes to the Datalake in order to receive the respective monitoring data. Last but not least, the SLA is stored in an intermediate storage so as to reduce the necessary calls to external components.

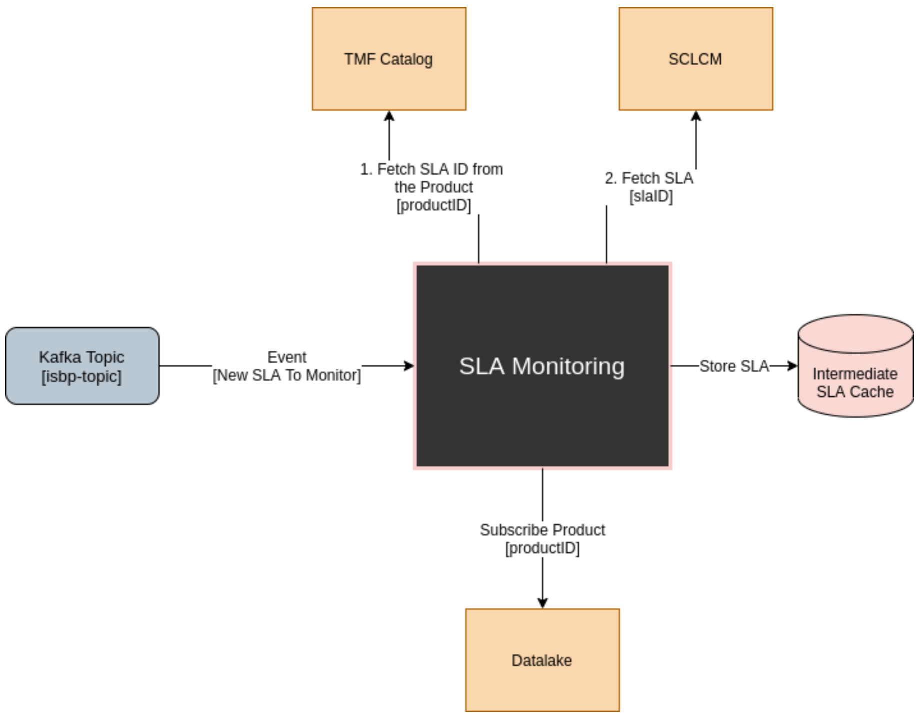

Figure 3-30: SLA Monitoring Preparation Phase

In step 3 to 5, we can identify the second phase of the SLA Monitoring component, the **monitorization phase** (Figure 3-31). At this stage, the Data Lake streams the monitoring data to SLA Monitoring and the later analyses this data by comparing the metrics (SLIs) such as availability or response time, with the thresholds defined in SLOs of SLAs, in order to detect SLA violations.

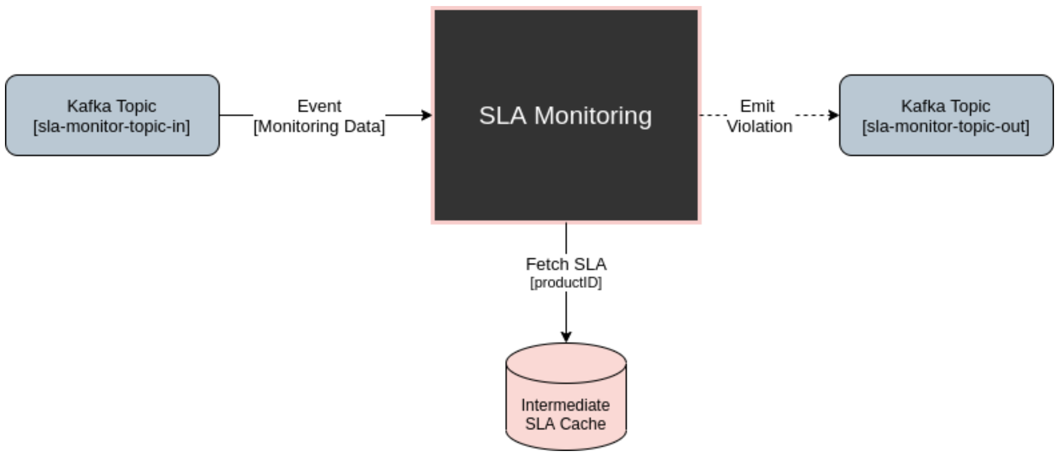

Figure 3-31: SLA Monitoring Monitorization Phase

Other components can subscribe to the SLA Monitoring output topic in the Datalake to receive notifications about the SLA Violations and such notifications are subsequently propagated to the smart contract for the purposes of re-calculating SLA status. This service is provided by the Smart Contracts Lifecycle Management functional block and works closely with the DLT and Smart Contracts to realise trustworthy SLA governance.

Regarding the SLA Breach Prediction module, the operational pattern that describes the sequence of actions that are linked to it is shown in Figure 3-32.

The workflow for the Intelligent SLA Breach Prediction consists of the following steps (it can be noted that steps 1 to 6 are similar to the SLA Monitoring flow described before):

Step 1-2: The Intelligent Slice and Service Manager (ISSM) pushes a new SLA event to a Data Lake queue, which is received by the Intelligent SLA Breach Predictor (ISBP).

Step 3–4: The Monitoring Data Aggregator (MDA) collects monitoring data on the Service Provider slice from the xRM and pushes them to a Data Lake queue.

Steps 5: The newly arrived monitoring data is processed and stored in a Data Lake database.

Step 6: The Data Lake streams the monitoring data to ISBP through a queue.

Steps 7: ISBP applies a ML model on the monitoring data from a repository of trained models and generates a prediction for every X data points.

Step 8: If the predicted value exceeds the specified threshold, an SLA breach notification message (for the specification see Table 4-57) is sent to ISSM through the Data Lake queue.

Step 9: Every data point received is compared against the prediction of the previous iteration in order to aqcuire an accuracy. The average of such accuracies is used to determine whether the model needs to be trained further.

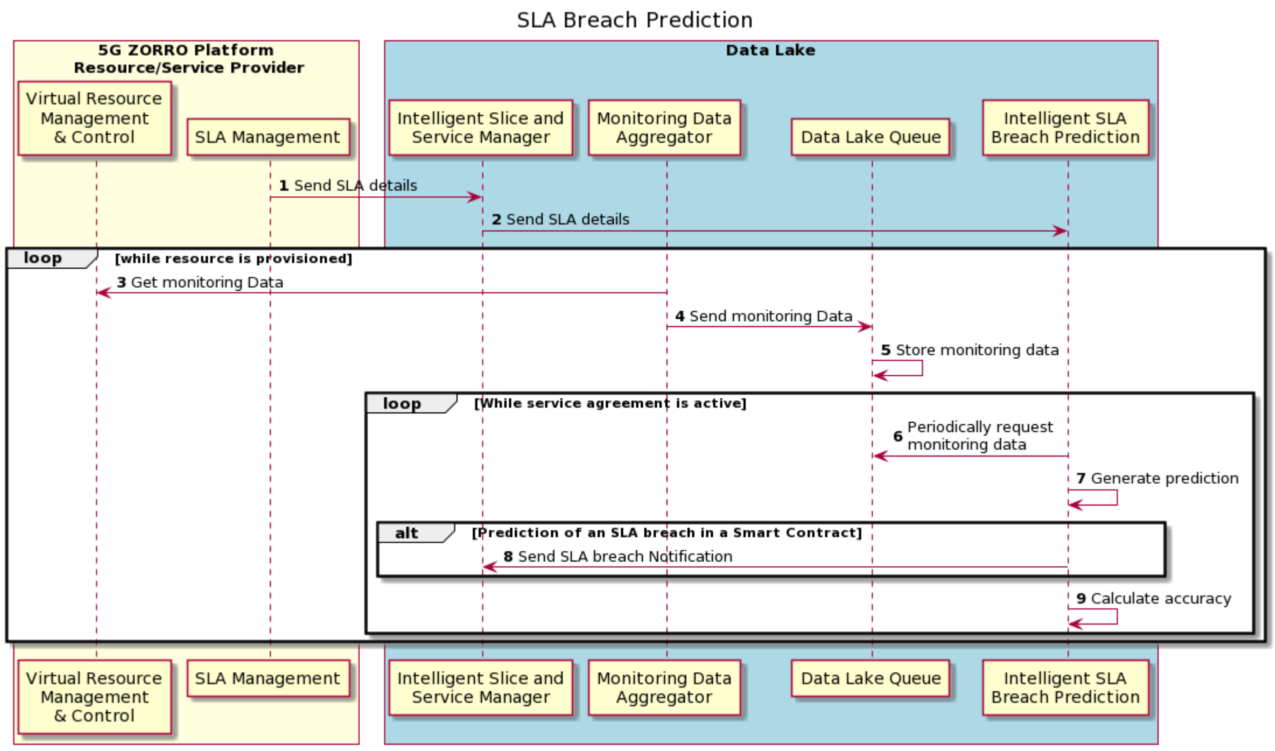
SLA Breach Prediction
5G ZORRO Platform Resource/Service Provider
Data Lake
Virtual Resource Management & Control
SLA Management
Intelligent Slice and Service Manager
Monitoring Data Aggregator
Data Lake Queue
Intelligent SLA Breach Prediction
1 Send SLA details
2 Send SLA details
loop [while resource is provisioned]
3 Get monitoring Data
4 Send monitoring Data
5 Store monitoring data
loop [While service agreement is active]
6 Periodically request monitoring data
7 Generate prediction
alt [Prediction of an SLA breach in a Smart Contract]
8 Send SLA breach Notification
9 Calculate accuracy

Figure 3-32: SLA Breach Prediction Workflow

3.3.2.3 *APIs*

Operation name: **startSLAMonitoringProcess**			
Description		This operation is used to start receiving monitoring data for specific contract SLAs.	
Input Parameters		**Type**	**Description**
	SLA Event	Json	Event that contains the event type, the transactionId, the productId and the status. This event originates in the ISBP and signals the start of an SLA monitorization process.
	Product	Json	Product fetched from the TMF Catalog component using the productId from which we will retrieve the Href field from where we will retrieve the SLA
	SLA	Json	SLA is fetched from the SCLCM component using the Href field. This SLA will be stored on an intermediate storage so as to reduce the number of calls to external components during the monitorization process.
Output Parameters		**Type**	**Description**
	N/A		
Notes			
It stands to reason that the Product and the SLA are not direct inputs but instead artefacts that are fetched by the SLA Monitoring. These were marked as inputs since they are required to be present in the TMF and SCLCM components to be used in the component.			

Operation name: **terminateSLAMonitoringProcess**			
Description		This operation is used to stop receiving and analysing monitoring data for specific contract SLAs.	
Input Parameters		**Type**	**Description**
	SLA Event	Json	Event that contains the event type, the transactionId, the productId and the status.

	Product	Json	Product fetched from the TMF Catalog component using the productId from which we will retrieve the Href field from where we will retrieve the SLA
Output Parameters		**Type**	**Description**
	N/A		
Notes			
It stands to reason that the Product is not a direct input but instead an artefact that is fetched by the SLA Monitoring. It was marked as input since it is required to be present in the TMF to be used in the component.			

Operation name: **publishSLAViolationNotification**			
Description		The component sends to the subscribed modules SLA violations as they are occurring. The violations include the reference to what SLA is violated, the SLO and the actual value of the violation.	
Input Parameters		**Type**	**Description**
	Monitoring Event	Json	Event originated from the MDA component that includes the operatorId, the networkId and the monitoringData. This event signals a violation verification.
	SLA	Json	SLA that contains the rules which will dictate when a violation occurs. To this end, we compare the monitoring data with the operator, the reference and tolerance values present in the rules.
Output Parameters		**Type**	**Description**
	Violation	Json	Each violation includes 1. Unique ID, 2. SLA Reference, 3. The rule violated, 4. Value of the violation

Table 3-4: List of Intelligent SLA Breach Prediction capabilities

Service name**: SLA Breach Prediction**		Type**: *Cross-domain***
Capabilities	**Support (O|M)**	**Description**
Start SLA Breach Prediction pipeline	**M**	The service receives a new SLA event and then waits for monitoring data from the *Monitoring Data Aggregator* service for specific contract SLAs using AI techniques in order to predict possible breaches in SLAs and detect anomalies.
Update SLA Breach Prediction	**M**	This capability is used when there is a need to change the characteristics of an SLA Breach Prediction Process. It can be used when the contracts SLAs have been changed and when there is a need to add/remove SLAs or modify Start/End Times.
Terminate SLA Breach Prediction	**M**	This capability is used to stop receiving and analysing resources monitoring data for specific contract SLAs.
Return active SLA Breach Predictions	**M**	This capability is used to create a list of active SLA Breach Predictions, their descriptions, their breach predictions and their subscribers.
Publish Notifications for SLA Breach Predictions	**M**	This capability is used to send to the subscribed modules predictions for SLA violations. The predictions include the SLOs that will be violated, when these violations will occur, to what extent the SLAs will be violated and the level of certainty for SLAs violations.
Configure Machine Learning Algorithms	**M**	This capability is used to configure the machine learning algorithms based on the types of SLA metrics that are being monitored and the associated requirements.

Operation name: **startSLABreachPrediction**		
Description	This operation is used to start receiving and analysing monitoring data for specific contract SLAs.	
Input Parameters	**Type**	**Description**
Description	String	A textual description of SLA Breach Prediction.
ContractIDs	List of Strings	The identifiers of the contracts for which predictions will be made for violations of their SLAs.
SLAMetrics	String array	List of specific SLA metrics for the case that the predictions will not be related to all SLAs of the contracts.
StartTime	DateTime	The SLA Breach Prediction could start immediately or at a specific time.
EndTime	DateTime	The SLA Breach Prediction for the specific contract/SLAs will be terminated and the END Time of when the module will receive the 'Terminate SLA Breach Prediction" command.
Output Parameters	**Type**	**Description**
SLABreachPredictionID	String	In case of success, it returns the Identifier of the new SLA Breach Prediction process.
ErrorMessageID	String	In case of error, it returns the Identifier of the error message.
ErrorDescription	String	Optionally it returns specific details for the error.

Operation name: **terminateSLABreachPrediction**		
Description	The SLA Breach Prediction is terminated immediately.	
Input Parameters	**Type**	**Description**
SLABreachPredictionID	String	The Identifier of the SLA Breach Prediction process that will be terminated.
EndTime	DateTime	The SLA Breach Prediction could be terminated immediately or at a specific time.
Output Parameters	**Type**	**Description**
Status	String	In case of success, it returns a success message, otherwise an error message.
ErrorMessageID	String	In case of error, it returns the Identifier of the error message.
ErrorDescription	String	Optionally it returns specific details for the error.
Operation name: **getActiveSLABreachPredictions**		
Description	This operation is used to create a list of active SLA Breach Predictions Operations, their descriptions, their breach predictions and their details.	
Input Parameters	**Type**	**Description**
SLABreachPredictions	String array	Select all or a list of SLA Breach Predictions.
Output Parameters	**Type**	**Description**
Description	String	A textual description of SLA Breach Prediction.
ContractIDs	String	The identifiers of the contracts of each SLA Breach Prediction.
MonitoredSLAMetrics	String array	List of specific SLAs of each SLA Breach Prediction.
StartTime	DateTime	The SLA Breach Prediction start time.
EndTime	DateTime	The SLA Breach Prediction end time.
Subscribers	String array	List of Subscriber Identifiers.
violations	String array	Each violation includes: 1. the contract ID,

			2. the SLA metrics that will be violated, 3. when these violations will occur, 4. to what extent each SLA will be violated and the level of certainty
	Status	String	In case of success, it returns a success message, otherwise an error message.
	ErrorMessageID	String	In case of error, it returns the Identifier of the error message.
	ErrorDescription	String	Optionally it returns specific details for the error.

Operation name: **publishSLA Breach PredictionsNotifications**			
Description		The module sends to the subscribed modulesdispatches predictions for SLA violations to ISSM through a Data Lake queue. The predictions include the SLA metrics that will be violated, when these violations will occur, to what extent the SLAs will be violated and the level of certainty.	
Input Parameters		**Type**	**Description**
	-	-	-
Output Parameters		**Type**	**Description**
	Violations	String array	Each violation includes: 1. the contract ID, 2. the SLA metrics that will be violated, 3. when these violations will occur, 4. to what extent each SLA will be violated and the level of certainty

3.3.3 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 infrastructure edge resources.

As previously stated, stakeholders acting as assets consumers interact with the Marketplace, and in particular with the services provided by the Resource and Service Offer Catalogue, 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 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 machine learning.

3.3.3.1 *5GZORRO Specific enhancements*

In this scope, a zero-touch offer discovery, suitable for making decisions about what resources are the most appropriate to use in each particular case, will be supported by two modules/modalities:

- **Automated Discovery:** provided by the Marketplace Resource and Service Offering Catalogue. Based on imperative constraints and considerations provided by the consumer as part of the lookup request (e.g., category, geographic location, price, provider preference). Response provides all the available offers that match the requested filters.

- **Intent-based Discovery:** provided by the Smart Resource and Service Discovery Application. Based on (high-level) intent-based priorities (such as resource and service type and requirements). Response provides the most suitable offers according to the particular intent the consumer of the application has requested.

The architecture of the Smart Resource and Service Discovery application module is illustrated in Figure 3-33. More in details, the application is composed by two functional submodules (i.e., the clustering and classification submodule and the intent-based submodule), both implemented as microservices, plus a common database. In the figure, this architecture is represented in relation with other 5GZORRO components, with which the Smart Resource and Service Discovery application interacts, such as the Resoure and Service Offering Catalogue and the ISSM, introduced in 5GZORRO D4.4 [3]. In particular, the request for this module is originating from ISSM. As a result of the request, the Smart Resource and Service Discovery returns to ISSM a list of candidate offers. The method that is used to retrieve these offers is described in the following part of this section.

In addition to responding to one of the 5GZORRO objectives (i.e., to develop ML-based intelligent discovery functions), the Smart Resource and Service Discovery application plays a fundamental role in the attainment of the following KPI:

- Support intent-based API to guide the AI-driven resource discovery system (KPI target: open 5GZORRO API specification for resource discovery)
- Automatically discover and "inventorize" various types of resources (i.e., compute, storage, network at core, edge, far-edge), spectrum and services capabilities from different domains and service providers

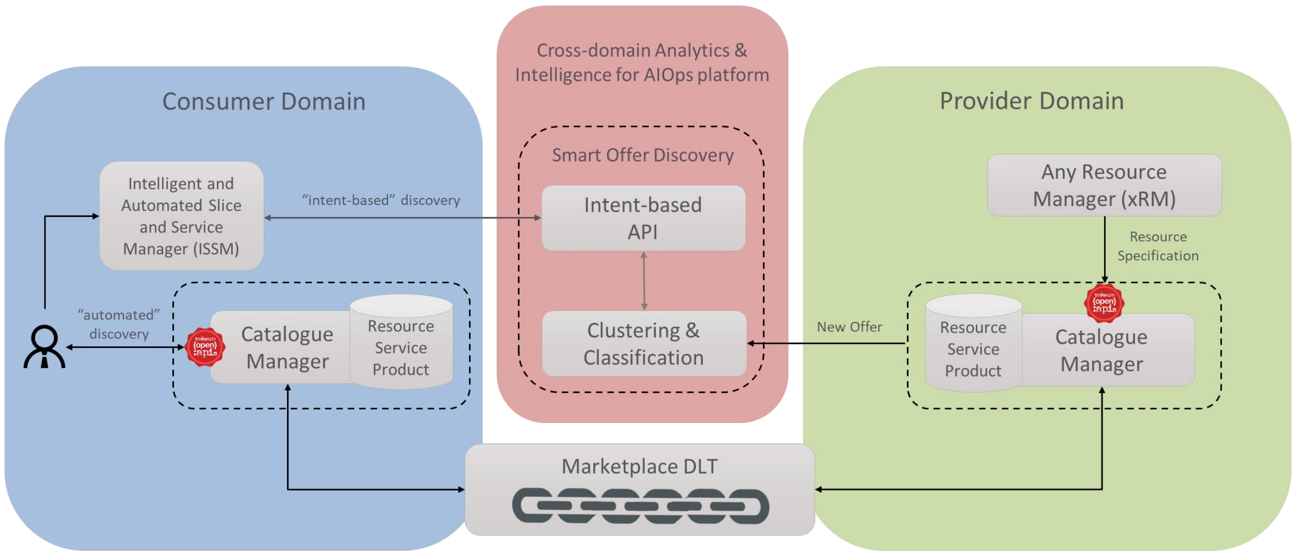

Figure 3-33: Smart Resource and Service Discovery application architecture

To achieve this goal, this module exposes an API providing intent-based support and leveraged by AI-based models to perform intelligent resource discovery. As stated in [6], an *intent-based API is primarily the abstracted and simplified API to request an intent*. Instead of specifying in detail the characteristics of the desired resource offers (e.g., via filters over the RSOC), just a prescription is given, which in this case may correspond to an offer type and/or to an aspect to be prioritized (e.g., "*I want an edge offer with 4GB of RAM*"). In particular, this component is aimed to recognize keywords in the provided intention and to translate them into specific groups/classes.

Regarding the supporting AI-based models, the intelligent discovery will be enabled by data clustering techniques where resources of a certain class (e.g., based on location) will be grouped together. In particular, for the sake of efficiency and meeting the dynamism requirement of 5GZORRO, the proposed methodology consists of two steps:

(1) Off-line clustering: a clustering algorithm would take care of taking offline a training dataset to learn the similarities from the resources properties and obtain the groups/clusters.

(2) On-line prediction: Upon the arrival of a new incoming offer, a supervised classification algorithm that is trained with the dataset of Step 1 (now labelled) will predict to which of the computed clusters it belongs.

In terms of technical implementation, clustering computation is based on a variety of features (i.e., criteria), as well as relations between them, contained in the categorical information conforming the offers. The topics that are relevant to group offers with similar features include common attributes (such as price) as well as distintic characteristics, which vary according to the offer type (i.e., the features identifying a spectrum offer, such as frequency band, would differ from the ones characterizing computational resources at a cloud site, such as RAM memory). To ensure that the considered clustering remains accurate over time, this computation can be repeated periodically (e.g., after a given number of incoming offers is reached or during planned system maintenance operations) taking as training dataset incoming offers accumulated since the last performed training.

3.3.3.2 *Specific and relevant workflows*

The workflow associated with the services offered by the Smart Resource and Service Discovery application is shown in Figure 3-34.

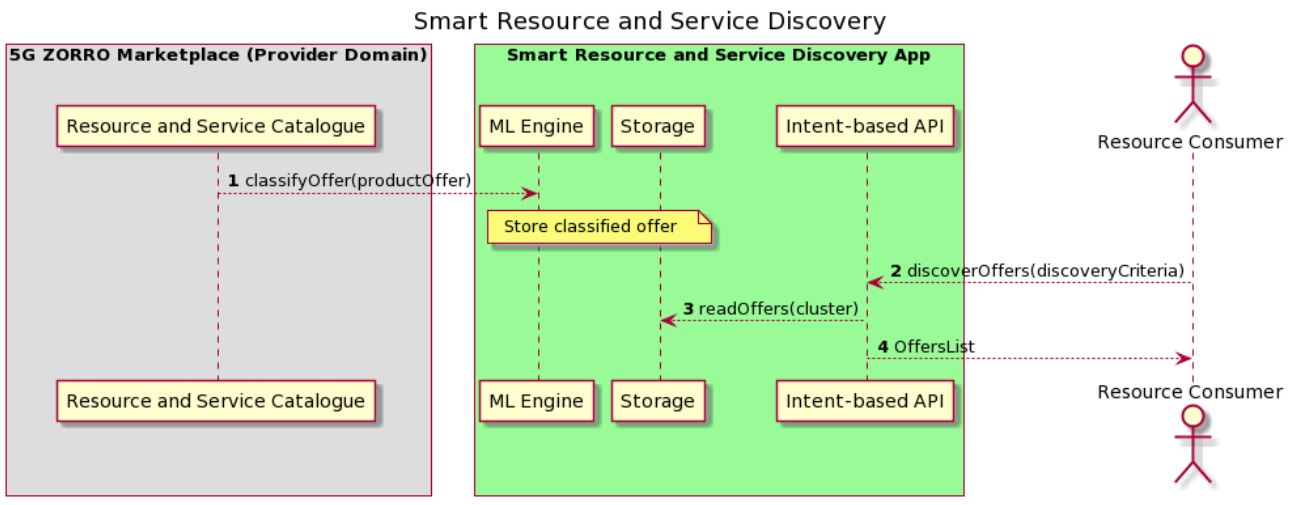
Smart Resource and Service Discovery
5G ZORRO Marketplace (Provider Domain)
Smart Resource and Service Discovery App
Resource and Service Catalogue
ML Engine
Storage
Intent-based API
Resource Consumer
1 classifyOffer(productOffer)
Store classified offer
2 discoverOffers(discoveryCriteria)
3 readOffers(cluster)
4 OffersList
Resource and Service Catalogue
ML Engine
Storage
Intent-based API
Resource Consumer

Figure 3-34: Smart Resource and Service Discovery Workflow

Step 1: Upon the placement of a new offer, a trained ML-model for online prediction, served at the Cross-domain Analytics & Intelligence for AIOps Platform, is used to define the group (i.e., cluster) to which the new offer belongs. The resulting classified offer is then stored for their consumption by the discovery requests.
Step 2-3: The information provided by the consumer request via the intent-based interface is automatically translated into the specific cluster(s) that best reflect the customer expectations.
Step 4: The resulting offers contain an indicative score defining the degree of the match that is computed based on a formula on top of the discovery criteria. Moreover, aside from the score they can then be further analysed at the consumer domain for selecting the most suitable offer following other criteria (e.g., trust and reputation assessments).

3.3.3.3 *APIs*

To perform the aforementioned functionalities, the operations supported by the Smart Resource and Service Discovery application interfaces are described below.

Operation name: **predictOffer**		
Description	Endpoint for classifying an offer, resulting in a new entry added to the application database with classified offers.	
Input Parameters	**Type**	**Description**
Product Offer	Object	Object containing the product offer information based on the considered information model (see Section 4.3)
Output Parameters	**Type**	**Description**
Classified Product Offer	Object	Object containing the product offer information extended with the predicted cluster

Operation name: **removeClassifiedOffer**		
Description	Endpoint for removing a classified offer from the application database (because of the offer being no longer available on the Marketplace).	
Input Parameters	**Type**	**Description**
Product Offer DID	String	DID of the product offer to be removed
Output Parameters	**Type**	**Description**
-	-	-

Operation name: **discoverOffers**		
Description	Endpoint for discovering available offers, returning a list of most suitable offers for final selection.	
Input Parameters	**Type**	**Description**
Discovery Criteria	String	String containing (high-level) priorities to be taken into account for the discovery. This parameter will be translated into the specific clusters that correspond with the requested priority.
Maximum No. of Samples	Integer	(Optional) Numeric value representing the maximum number of samples to be retrieved.
Output Parameters	**Type**	**Description**
List of Offers	List	A list of offers matching the supplied discovery criteria. Each offer also contains a score computed by the relevance to the discovery criteria.

Operation name: **clustersGeneration**			
Description		Endpoint for generating new clusters (e.g., after a given number of incoming offers is reached or during planned system maintenance operations).	
Input Parameters		**Type**	**Description**
	Offers dataset	Path	Set of offers to (re-)generate the clusters offline and update the classifier model for online prediction.
Output Parameters		**Type**	**Description**
	-	-	-

4 5G Service Layer Information Elements

4.1 5GZORRO DIDs

DIDs are a novel type of identifiers proposed by W3C that allows associating any subject, such as stakeholders, resources, services, organizations, entities, and so on, with a digital identity. In other words, DIDs are global and unique identifiers that do not need a centralised registration authority since they are created and/or recorded using cryptographic proofs. Furthermore, decentralised identifiers provide characteristics such as decentralised control, privacy, security, interoperability, and extensibility, making it a pioneering technology while addressing some shortcomings of current identity management systems.

In order to thoroughly understand the functionality of decentralised identifiers and the actions associated with them, several key concepts should be known beforehand. According to W3C Decentralized Identifier WG [13], DIDs can utilise multiple technologies and cryptographies in order to carry out the creation, persistence, resolutions, or interpretation of DIDs. In general, a decentralised identifier is a URL, similar to "did:5gzorro:123456789abcdefghi", composed of a URI scheme identifier, an identifier for the DID method, and a DID method-specific identifier, respectively. Normally, we can find service endpoints, public keys as well as other attributes or claims detailing DID Subject or *DID delegate* characteristics. These attributes are made up of a generic format of DID, usually a JSON schema, that is declared in the *DID Core specification*. The official syntax of a decentralised identifier, also known as *DID scheme*, is declared in a *DID method* specification. In spite of the several DID methods available, 5GZORRO is going to generate and register a new method named "5gzorro" in order to build a specific *representation* that contains the necessary *attributes*, *properties*, and *claims*. Besides, this new DID method will utilise the distributed ledger technology deployed in the ecosystem. Finally, DIDs also provide *services* that are a communication mechanism with the DID's owner (subject) such as discovery services or agent services. Nevertheless, it should be pointed out that 5GZORRO Services are different from DID Services since 5GZORRO Services are communication services offered to end-user customers that are built on top of 5GZORRO Resources.

Another concept related to decentralised identity management is Verifiable Credentials (VCs) [14]. Even though it is related to decentralised identifiers, it is different from DID Documents. A Verifiable Credential is an electronic credential which can represent the same information that physical credentials represent in real life such as driving license, passport, health insurance card, and so on. Therefore, Verifiable Credentials represent statements made by an issuer in a tamper-evident and privacy-preserving manner. In this sense, the principal dissimilarity between DID Documents and VCs is that DID Documents are usually stored in the Blockchain while VCs are stored in a subject decentralised repository, for example. Thus, DID Documents must contain public data since they could be accessed by everyone, whilst VCs are only consulted by entities that have the necessary permissions. Taking into consideration the aforementioned, and since 5GZORRO Governance DLT is a permissioned public DLT (e.g., Hyperledger INDY), it is critical that 5GZORRO DID Documents that are registered in the Governance DLT contain no personal data. This is because, a public DID Document can be retrieved from a Verifiable Registry as soon as you know its DID without having to contact its controller/agent while the VC can only be retrieved from the Holder Agent. The Verifiable Credentials also has a data model, just like DIDs, with specific syntax and sematic defined by the W3C standard.

To mitigate DID correlation risks, it is possible to use pairwise unique DIDs, i.e., by using different DID for every relationship. In this case, the DID and its associated DID Document is not registered in the DLT and it is called Pseudonymous DID. A DID that is registered in the DLT is called Public DID.

4.1.1 High Level View

4.1.1.1 *5GZORRO DIDs*

The 5GZORRO DID syntax is compliant with W3C DID syntax by using a unique method name like "5gzorro". As such, an example for a 5GZORRO DID for some 5GZORRO resource traded in the Marketplace would look like: "did:5gzorro:123456789abcdefghi".

At this point, 5GZORRO is using the DID parameters already specified at https://w3c.github.io/did-spec-registries/.

The following 5GZORRO DID Subjects have been identified:

Subject type	Public	Description
5GZORRO Stakeholder	Yes	The legal entity that operates 5GZORRO administrative domains as defined in D2.1 [2], including Resource Providers, Resource Consumers, Regulators, etcetera.
5GZORRO License	Yes	The license that enables an entity that operates in 5GZORRO to provide authorized resources and/or services
5GZORRO Product	Yes	A 5G Product offer comprised by 5GZORRO resources and / or services, that is published in 5GZORRO Catalogue.
5GZORRO Business Agreement	Yes	The Business Relationship among 5GZORRO Stakeholders to support the trade of 5GZORRO Resources and Services in the 5GZORRO Marketplace with specific SLA requirements that are controlled with DLT Smart Contracts.

4.1.1.2 *Verifiable Credentials*

The Verifiable Credentials Services to be associated to 5GZORRO DID Subjects identified above, are described and mapped to 5GZORRO services in the Table 4-1 below.

Table 4-1: Mapping of 5GZORRO DID Subjects to 5GZORRO Services

Name	WFs usage	Claims Description	Holder	Issuer	Verifier
stakeholderVC	1.1	Claim about type of 5GZORRO role including Regulator, Resource Provider, Resource Consumer, Service Provider, and Service Consumer. Claims about Resource types to be provided or consumed	Admin DID Agent, Trading DID Agent	Admin DID Agent	Admin DID Agent Trading DID Agent
licenseVC	1.1	Claims about the license to provide products in 5GZORRO	5GZORRO Platform	Regulator DID Agent	Regulator DID Agent Trading DID Agent
productVC	3.1, 3.2, 3.3	Claim that product is available and ready to be consumed by service consumers and a certain behaviour is expected.	Trading Provider DID Agent	Admin DID Agent	Trading Consumer DID Agent

agreementVC		Claim that identifies the stakeholders involved in the agreement and the different roles played by each other including the identification of Resources and Services provided in the context of the agreement.	5GZORRO Business Agreement	Admin DID Agent	Trading DID Agent

4.1.2 Detailed Information Model

The different Information Models to be used by 5GZORRO Credentials are detailed in this section which are aligned with W3C Verifiable Credential.

4.1.2.1 *Core Verifiable Credential Information Model*

The Core Verifiable Credential Information Model that is used by all 5GZORRO DID Subjects, is composed by the following main parameters: the credentialSubject represents who is the subject or subjects of claims, the issuer is the person or entity that allocates this VC, the issuanceDate indicates when VC was emitted and validated, and the credentialStatus is utilized for knowing information about the current status of a VC.

More details are provided in the table below:

Parameter		Type	Description
credentialSubject			
	DID	DID	The Subject DID
	claims	List of Claims	Set of claims about the Verifiable Credential Subject. See Subject Claims Information Model below
issuanceDate		String (of an [*RFC3339*])	Indicates when VC was emitted and validated
credentialStatus		tyoe	Utilized for knowing information about the current status of a VC, CredentialStatus type that should provide enough information to determine the current status of the credential including "active", "suspended", "revoked"

4.1.2.2 *Verifiable Credential Claims Information Model*

The Information Model for the different Claim objects used according to the different credential types and DID subjects are defined in the tables below.

Parameter	Type	Description
stakeholderClaim	StakeholderClaim Object	A Claim object to be used in Credentials of StakeholderCredential type
agreementClaim	AgreementClaim Object	A Claim object to be used in Credentials of AgreementCredential type
productClaim	ProductClaim Object	A Claim object to be used in Credentials of ProductCredential type. Compliant with Catalogue ProductOffer Information Model
licenseClaim	LicenseClaim Object	A Claim object to be used in Credentials of StakeholderLicenseCredential type

Stakeholder Claim Object

Parameter		Type	Description
stakeholderProfile		Object	Personal information regarding the Stakeholder that operates services in the 5GZORRO platform
stakeholderRoles		List of Objects	List of Roles played by the Stakeholder
	role	String	Type of 5GZORRO role including Regulator, Resource Provider, Resource Consumer, Service Provider, and Service Consumers
	assets	List of Strings	Types of assets to be provided or consumed according to Stakeholder role including Resource sub-types (InformationResource, SpectrumResource, PhysicalResource, NetworkFunction, …)

Stakeholder License Claim Object

Parameter	Type	Description
stakeholderServices	List of Services	List of 5GZORRO platform services operated by the Stakeholder

Agreement/Product Claim Object

Parameter	Type	Description
DID	DID	This is the DID being traded by the Stakeholder for any of the claims
Handler	String	Entity that performs the required actions and is dependent on a DID for the effect

4.2 Smart Contract information model

Due to the unique nature of R3 Corda's architecture, the following information model definitions encapsulate the contract state (information model), but also the flows and associated state changes and constraints to aid understanding and intention.

Because model data is persisted in various off-chain repositories, where possible, the Corda models reflect a decision to reduce duplication. Instead, immutable non-repudiation of data stored on the ledger will be achieved through the storage of hashes on the DLT and pointers to the off-chain resource.

4.2.1 Product Offering

Product offers are published to the DLT along with their associated terms, SLAs etc., which are then synchronised across trading stakeholders (providers & consumers). Below is the definition of ProductOffering state object.

Table 4-2: Product Offering Information Model

Parameter	Type	Description
identifier	UniqueIdentifier / DID	A Corda identifier made up of a Corda specific UUID and an "external id" -> the DID of the offer
name	String	The name of the resource
productOffering	ProductOffering	The Product Offering following the TMF specifications
places	List<PlaceModel>	Geographical places of significance to the offer

productOfferTerms	List<ProductOfferTerms>	Terms such as licensing associated with the offer
productOfferingPrices	List<ProductPriceOfferings>	Prices associated with the offering
productSpecification	productSpecification	The specification of the product
serviceLevelAgreements	List<SLA>	List of SLAs defined for the offer
owner	Party	Corda Party is a Node identity

Once a provider has published a product offer it is synchronised across all trading participant nodes so that it can be added to their respective catalogue. Providers can also update and retire product offers, which are again synchronised with trading stakeholders.

4.2.2 Product Order

The publishing of a product order to the ledger is the initiation of agreement negotiation. The order is encapsulated by a ProductOrder contract state and governed by a Product Order Contract to enforce business rules over the contract.

A product order contract state does not duplicate the model definitions, rather the product order model is serialized, and an attachment is created and referenced in the contract state. A hash of the model also forms part of the contract state for simple detection of changes to the contract model (attachment).

Table 4-3: Product Order Information Model

Parameter	Type	Description
identifier	Unique Identifier / DID	A Corda identifier made up of a Corda specific UUID and an "external id" -> the DID of the offer
modelHash	String	A hash of the product order model
model	AttachmentRef()	A reference to an attachment ofject containing the full product order specification
validFor	TimePeriod	A start and optional end date for the lease of the products on the order
seller	Party	A Corda node identity
buyer	Party	A Corda node identity

Once a consumer has published a product order, a provider will be able to perform the necessary checks to determine if the request is serviceable. If it is, then they will accept the order, or failing that, reject it with a reason. Acceptance of an order will mean both parties enter the agreement and during the lifetime of the agreement participants can propose changes and terminate it as they see fit.

4.2.3 SLA DLT State

When an SLA is specified in a Product Offering wrapped in a Product Order that is persisted by the Smart Contract Lifecycle Manager in the DLT, this SLA is recorded on the DLT too. The SLA DLT State references the Product Order, the Service Level Agreement and the current state of the Service Level Agreement represented.

Table 4-4: SLA DLT State Information Model

Parameter	Type	Description
Identifier	UniqueIdentifier	A Corda identifier made up of a Corda specific UUID and an "external id" -> the DID of the offer
serviceLevelAgreement	ServiceLevelAgreement	The Service Level Agreement following the TMF specifications
serviceLevelAgreementState	SLAState	The current state of the Service Level Agreement represented: IN PLACE | VIOLATED | RETIRED
productOrderDID	String	The unique Distributed Identifier of the the Product Order cointaining this Service Level Agreement

buyer	Party	A Corda Node identity
seller	Party	A Corda Node identity
governanceParty	Party	A Corda Node identity

4.2.4 License Terms

If a Product Order has License Terms associated with it then a LicenseTerms Contract State (governed by a LicenseTerms contract) will be published to the ledger in order to track the lifecycle of any licensing actions in the same fashion that is done for the Service Level Agreement.

Table 4-5: License Terms Information Model

Parameter	Type	Description
identifier	UniqueIdentifier	A Corda identifier made up of a Corda specific UUID and an "external id" -> the DID of the offer
licenseTerm	LicenseTerm	The LicenseTerm information model referenced by this state persisted on the ledger
id	String	Distributed Identifier (DID) of the License Term
href	String	Reference to the License Term information model pre-stored in the SC LCM
type	Enum	The type of license: SUB | LIMIT | PAYG
amountLimit	Integer	The max instances/users etc permitted
durationLimit	Duration	An optional duration for the license
current	Integer	A current record of provisioned instances/users
lastUsage	Duration	Period usage was last calculated over
amountType	String	The type of unit to which the current and amountLimit fields relate: USERS | INSTANCES
productOrderDID	String	The unique Distributed Identifier of the the Product Order cointaining this Service Level Agreement
buyer	Party	A Corda Node identity
seller	Party	A Corda Node identity
governanceParty	Party	A Corda Node identity

4.2.5 Primitive Spectoken

Primitive spectokens are commited to the DLT reflecting the attributes of the associated spectrum license. Below is the definition of a Primitive Spectoken state object. A hash of the spectrum license also forms part of the contract state for simple detection of changes to the contract model.

Table 4-6: Primitive Spectoken Information Model

Parameter	Type	Description
identifier	Unique Identifier	Corda specific UUID
license	String	A hash of the spectrum license
owner	String	DID of the spectrum resource provider
country	String	Country the license applies to
startTime	Time	Spectrum license lease start time
endTime	Time	Spectrum license lease end time
startDL	Double	Start DL frequency in MHz
endDL	Double	End DL frequency in MHz
startUL	Double	Start UL frequency in MHz
endUL	Double	End UL frequency in MHz
duplexMode	String	TDD or FDD
band	Integer	Band number
technology	String	Cellular technology to be used (4G or 5G)

4.2.6 Derivative Spectoken

Derivative spectokens are commited to the DLT reflecting the attributes of the leased spectrum capabilities (via ordering of a Spectrum Offer). Below is the definition of a Derivative Spectoken state object.

Table 4-7: Derivative Spectoken Information Model

Parameter	Type	Description
identifier	Unique Identifier	Corda specific UUID
primitiveID	Unique Identifier	Corda specific UUID of the associated primitive spectoken
owner	String	DID of the spectrum resource consumer
price	Float	Price of the spectrum transaction
country	String	Country the license applies to
startTime	Time	Spectrum license lease start time
endTime	Time	Spectrum license lease end time
startDL	Double	Start DL frequency in MHz
endDL	Double	End DL frequency in MHz
startUL	Double	Start UL frequency in MHz
endUL	Double	End UL frequency in MHz
duplexMode	String	TDD or FDD
band	Integer	Band number
technology	String	Cellular technology to be used (4G or 5G)

4.3 Marketplace Offers Modelling

As mentioned in Section 3.2, in 5GZORRO we use the resource, service and product offer information models proposed by TM Forum [9] as the reference models to describe the 5GZORRO assets available in the market.

In TM Forum, a resource describes a traceable or inventoried asset, which in 5GZORRO will usually refer to the resource description. These resources are usually attached to services that in general terms describe how a resource can be instantiated, deployed or used. In 5GZORRO, we use services to describe network services and network slices, while VNFs, RAN, spectrum and computing assets remain as simple TM Forum resources. Both services and resources can be exposed to 3rd parties in order to be acquired, by means of product offers.

In the case of resources, their description is made available to the catalogue using the *ResourceSpecification* entity, which defines common attributes and features. This information model is depicted in Table 4-8.

Table 4-8: Resource Specification Information Model

Parameter	Type	Description
Id	String	Unique id in the catalogue of the resource specification
href	String	Unique reference of the resource specification
name	String	The name of the resource specification
resourceSpecCharacteristic	List of Resource Specification haracteristics	Description of the resource characteristics representing key features that are relevant for customers obtaining this resource via product offers.

In a similar way, the information model of a service is based on the *ServiceSpecification* entity, which is detailed in Table 4-9.

Table 4-9: Service Specification Information Model

Parameter	Type	Description
id	String	Unique service specification id in the catalogue
href	String	Unique reference of the service specification

name	String	The name of the service specification
serviceSpecCharacteristic	List of Service Specification Characteristics	Description of the service characteristics representing key features that are relevant for customers obtaining this service via product offers.

In order to compose an offer, technical descriptions from resource and service specifications are then incorporated into the *ProductSpecification* entity, which is detailed in Table 4-10.

Table 4-10: Product Specification Information Model

Parameter	Type	Description
id	String	Unique product specification id in the catalogue
href	String	Unique reference of the product specification
name	String	The name of the product specification
relatedParty	List of references to Organization objects	A list of references to the organization entity in the Catalogue. Used to provide the identifier of the offer supplier
resourceSpecificationRef	List of references to resourceSpecification objects (see Table 4-8)	A list of references to the associated resource specifications (VNF, Edge, Cloud, Spectrum, and/or RAN elements).
serviceSpecificationRef	List of references to serviceSpecification objects (see Table 4-9)	A list of references to the associated service specifications (network service and/or network slice).

In addition to technical specifications, other business information, such as pricing, is also need to compose an offer. In particular, prices are modelled using depicted in Table 4-11. This model describes the amount, usually of money, that is asked for or allowed when a product offer is bought, rented, or leased. In the context of 5GZORRO, this model also exposes licensing constraints, duration of use permitted and the business agreements involved in the trading of the software product. This model helps configure the e-Licensing Manager (eLM) of the 5GZORRO platform [3] and enables a way to configure, monitor and report the actual costs incurred during the usage of a resource or service.

Table 4-11: Product Offering Price Information Model

Parameter	Type	Description
id	String	Unique offering price id in the catalogue
href	String	Unique reference of the offering price
name	String	The name of the product offering price
description	String	Description of the product offering price
price	Float	The amount of money that characterizes the price.
priceType	Enumerate	A category that describes the price and associates a business model. ONE TIME for a flat rate, USAGE for a Pay-as-you-Grow model and RECURRING for a subscription plan (refer to the eLM description in D4.4 [3])
prodSpecCharValueUse	ProductSpecification CharacteristicValueUse	Specificies additional properties to the price such as resource/service identification, the PriceLogic that describes the monitoring metric use to evaluate the actual costs (time, instances, an application metric ...), additional constraints to the usage of the price.
recurringChargePeriodLength	Integer	The period of the recurring charge (0 if it is not applicable, e.g., ONE TIME price)
recurringChargePeriodType	Enumerate	The period to repeat the application of the price (HOUR, DAY, MONTH, YEAR)
unitOfMeasure	Object	A quantity (Quantity). A number and unit representing how many of a ProductOffering is available at the offered

		price. Aligned with the PriceLogic to correctly report the monitoring metric of this price.
validFor	Object	A time period for which the price is valid containing endDateTime and startDateTime.

Finally, product-oriented entities previously presented are used by the information model of the resulting product offer as depicted in Table 4-12.

Table 4-12: Product Offering information Model

Parameter	Type	Description
id	String	Unique internal id for the product offer in the catalogue
href	String	Unique reference of the product offer
DID	DID	Distributed identifier of the product offer
Name	String	Name of the product offer
Category	String	Group to which the product offer belongs (VNF, Edge, Cloud, Spectrum, RAN, Network Service or Network Slice)
place	List of references to Place objects	Geographical places where the product offer is valid
productSpecification	Reference to ProductSpecification object (see Table 4-10)	Reference to the product specification associated to the offer
productOfferingPrice	List of references to ProductOfferingPrice objects (see Table 4-11)	A list of references to the pricing models available for the product offer
serviceLevelAgreement	Reference to SLA object	Reference to the Service Level Agreement associated to the offer

The following subsections detail the specific characteristics considered for each one of the different 5GZORRO resources and services that extend the corresponding specification model in order to better illustrate the particularities of every modeled entity.

4.3.1 Network Slice characteristics information model

As previously mentioned, offers for network slices are modelled in a standardised manner based on the TM Forum Information Framework. Following the TM Forum service categories, network slices in 5GZORRO are mapped to Resource Facing Services (RFS) that reference to infrastructure resource objects (modelling compute or RAN elements) required to establish the considered network slice. The set of specific characteristics used to define additional features for network slices, following the information model of *ServiceSpecCharacteristic* in [8], is described from Table 4-13 to Table 4-21.

Table 4-13: Isolation level Characteristic for Network Slice Service Specification

Parameter		Type	Description
name		String	Isolation level
description		String	The level of isolation of the considered network slice, i.e., how is it separated from other slices
resourceSpecCharacteristicValue		List of objects	
	value	String	Isolation level of this network slice (e.g., logical)

Table 4-14: Data network access Characteristic for Network Slice Service Specification

Parameter	Type	Description
name	String	Data network access
description	String	This attribute defines how the data network of the considered network slice handle the user data
resourceSpecCharacteristicValue	List of objects	
value	String	Data network access of this network slice (e.g., Internet)

Table 4-15: Area of service Characteristic for Network Slice Service Specification

Parameter	Type	Description
name	String	Area of service
description	String	This attribute specifies the area where the UEs can access the considered network slice
resourceSpecCharacteristicValue	List of objects	
value	String	Two letter country codes (e.g., ES)

Table 4-16: Access Technology Characteristic for Network Slice Service Specification

Parameter	Type	Description
name	String	Access technology
description	String	This attribute defines the access technology included in the network slice
resourceSpecCharacteristicValue	List of objects	
value	String	Access technology of this network slice (e.g., 5G)

Table 4-17: Radio spectrum Characteristic for Network Slice Service Specification

Parameter	Type	Description
name	String	Radio spectrum
description	String	This attribute defines the radio spectrum in which the network slice should be supported
resourceSpecCharacteristicValue	List of objects	
value	String	Spectrum band of this network slice (e.g., n77)

Table 4-18: Maximum uplink throughput Characteristic for Network Slice Service Specification

Parameter	Type	Description
name	String	Maximum uplink throughput
description	String	This attribute describes the maximum uplink data rate supported by the considered network slice
resourceSpecCharacteristicValue	List of objects	
value	Numeric	Maximum uplink throughput of this network slice (e.g., 100 000)
unitOfMeasure	String	Maximum uplink throughput unit (e.g., kbps)

Table 4-19: Maximum uplink throughput per UE Characteristic for Network Slice Service Specification

Parameter	Type	Description
name	String	Maximum uplink throughput per UE
description	String	This attribute describes the maximum uplink data rate per UE of the considered network slice
resourceSpecCharacteristicValue	List of objects	

	value	Numeric	Maximum uplink throughput per UE of this network slice (e.g., 10 000)
	unitOfMeasure	String	Maximum uplink throughput per UE unit (e.g., kbps)

Table 4-20: Maximum downlink throughput Characteristic for Network Slice Service Specification

Parameter		Type	Description
name		String	Maximum downlink throughput
description		String	This attribute describes the maximum downlink data rate supported by the considered network slice
resourceSpecCharacteristicValue		List of objects	
	value	Numeric	Maximum downlink throughput of this network slice (e.g., 100 000)
	unitOfMeasure	String	Maximum downlink throughput unit (e.g., kbps)

Table 4-21: Maximum downlink throughput per UE Characteristic for Network Slice Service Specification

Parameter		Type	Description
name		String	Maximum downlink throughput per UE
description		String	This attribute describes the maximum downlink data rate per UE of the considered network slice
resourceSpecCharacteristicValue		List of objects	
	value	Numeric	Maximum downlink throughput per UE of this network slice (e.g., 10 000)
	unitOfMeasure	String	Maximum downlink throughput per UE unit (e.g., kbps)

4.3.2 Network Service characteristics information model

Network services are abstracted as Customer Facing Services (CFS) that can be acquired and consumed by other 3rd party domains. These service entities are supported by RFS including network slices and VNF/CNF services. The set of specific characteristics used to define additional features for network services, following the information model of *ServiceSpecCharacteristic* in [8], is described from Table 4-22 to

Table 4-26.

Table 4-22: nsdId Characteristic for NS Service Specification

Parameter	Type	Description
name	String	nsdId
description	String	ID of the NS descriptor
resourceSpecCharacteristicValue	List of objects	
value	String	Identifier

Table 4-23: serviceType Characteristic for NS Service Specification

Parameter	Type	Description
name	String	Service Type
description	String	Network Service managed by this NS
resourceSpecCharacteristicValue	List of objects	
value	String	Service Type of this VNF

Table 4-24: vCPURequirements Characteristic for NS ServiceSpecification

Parameter	Type	Description
name	String	vCPU Requirements
description	String	vCPU lower bound and upper bound
resourceSpecCharacteristicValue	List of objects	
alias-lower-bound	String	min-vCPU
value-lower-bound	String	Minimum vCPU requirements for this NS
alias-upper-bound	String	max-vCPU
value-upper-bound	String	Maximum vCPU requirements for this NS

Table 4-25: virtualMemoryRequirements Characteristic for NS Service Specification

Parameter	Type	Description
name	String	Virtual Memory Requirements
description	String	Virtual Memory lower bound and upper bound
resourceSpecCharacteristicValue	List of objects	
alias-lower-bound	String	min-virtual-memory
value-lower-bound	String	Minimum Virtual Memory requirements for this NS
alias-upper-bound	String	max-virtual-memory
value-upper-bound	String	Maximum Virtual Memory requirements for this NS
unitOfMeasure	String	GB

Table 4-26: storageRequirements Characteristic for NS Service Specification

Parameter	Type	Description
name	String	Storage Requirements
description	String	Storage lower bound and upper bound
resourceSpecCharacteristicValue	List of objects	
alias-lower-bound	String	min-storage
value-lower-bound	String	Minimum Storage requirements for this NS
alias-upper-bound	String	max-storage
value-upper-bound		Maximum Storage requirements for this NS
unitOfMeasure	String	GB

4.3.3 Edge/Cloud characteristics information model

The set of specific characteristics used to define additional features for Edge/Core resources, following the information model of *ResourceSpecCharacteristic* in [9][7], is described from Table 4-27 to Table 4-30.

Table 4-27: CPU Characteristic for Edge/Cloud Resource Specification

Parameter		Type	Description
name		String	cpu
description		String	The number of (virtual) cores of the considered computing resource
resourceSpecCharacteristicValue		List of objects	
	value	Double	Numeric value of CPU (e.g., 4)

Table 4-28: Memory Characteristic for Edge/Cloud Resource Specification

Parameter	Type	Description
name	String	memory

description	String	The amount of RAM of the considered computing resource
resourceSpecCharacteristicValue	List of objects	
value	Double	Numeric value of RAM (e.g., 8)
unitOfMeasure	String	RAM unit (e.g., GB)

Table 4-29: Storage Characteristic for Edge/Cloud Resource Specification

Parameter	Type	Description
name	String	storage
description	String	The amount of storage of the considered computing resource
resourceSpecCharacteristicValue	List of objects	
value	Double	Numeric value of storage (e.g., 40)
unitOfMeasure	String	Storage unit (e.g., GB)

Table 4-30: Bandwidth Characteristic for Edge/Cloud Resource Specification

Parameter	Type	Description
name	String	bandwidth
description	String	The amount of bandwidth of the computing resource
resourceSpecCharacteristicValue	List of objects	
value	Double	Numeric value of bandwidth (e.g., 100)
unitOfMeasure	String	Bandwidth unit (e.g., Mbps)

4.3.4 VNF/CNF characteristics information model

The set of specific characteristics used to define additional features for VNF/CNF resources, following the information model of *ResourceSpecCharacteristic* in [7], is described from Table 4-31 to Table 4-36.

Table 4-31: vnfdId Characteristic for VNF/CNF Resource Specification

Parameter	Type	Description
name	String	vnfdId
description	String	ID of the VNF descriptor
resourceSpecCharacteristicValue	List of objects	
value	String	Identifier

Table 4-32: functionType Characteristic for VNF/CNF Resource Specification

Parameter	Type	Description
name	String	FunctionType
description	String	Network function managed by this VNF
resourceSpecCharacteristicValue	List of objects	
value	String	Function Type of this VNF

Table 4-33: vCPURequirements Characteristic for VNF/CNF Resource Specification

Parameter	Type	Description
name	String	vCPU Requirements
description	String	vCPU lower bound and upper bound
resourceSpecCharacteristicValue	List of objects	
alias-lower-bound	String	min-vCPU
value-lower-bound	String	Minimum vCPU requirements for this VNF
alias-upper-bound	String	max-vCPU

value-upper-bound	String	Maximum vCPU requirements for this VNF

Table 4-34: virtualMemoryRequirements Characteristic for VNF/CNF Resource Specification

Parameter	Type	Description
name	String	Virtual Memory Requirements
description	String	Virtual Memory lower bound and upper bound
resourceSpecCharacteristicValue	List of objects	
alias-lower-bound	String	min-virtual-memory
value-lower-bound	String	Minimum Virtual Memory requirements for this VNF
alias-upper-bound	String	max-virtual-memory
value-upper-bound	String	Maximum Virtual Memory requirements for this VNF
unitOfMeasure	String	GB

Table 4-35: storageRequirements Characteristic for VNF/CNF Resource Specification

Parameter	Type	Description
name	String	Storage Requirements
description	String	Storage lower bound and upper bound
resourceSpecCharacteristicValue	List of objects	
alias-lower-bound	String	min-storage
value-lower-bound	String	Minimum Storage requirements for this VNF
alias-upper-bound	String	max-storage
value-upper-bound		Maximum Storage requirements for this VNF
unitOfMeasure	String	GB

Table 4-36: nExtCpd Characteristic for VNF/CNF Resource Specification

Parameter	Type	Description
name	String	nExtCpd
description	String	Number of external connection points
resourceSpecCharacteristicValue	List of objects	
value	String	Number of external connection points

4.3.5 RAN (active & passive) characteristics information model

The set of specific characteristics used to define additional features for RAN resources, following the information model of *ResourceSpecCharacteristic* in [7], is described from Table 4-37 to Table 4-44.

Table 4-37: RAN type Characteristic for RAN Resource Specification

Parameter	Type	Description
name	String	ranType
description	String	Expresses the RAN type to differentiate Wi-Fi access points and cells
resourceSpecCharacteristicValue	List of objects	
value	String	Type of RAN (e.g., cell access point)

Table 4-38: Central downlink frequency Characteristic for RAN Resource Specification

Parameter		Type	Description
name		String	centralDLFrequency
description		String	The central downlink frequency of the radio resource
resourceSpecCharacteristicValue		List of objects	
	value	Double	Numeric value of frequency (e.g., 3625)
	unitOfMeasure	String	Megahertz [MHz]

Table 4-39: Central uplink frequency Characteristic for RAN Resource Specification

Parameter		Type	Description
name		String	centralULFrequency
description		String	The central upnlink frequency of the radio resource
resourceSpecCharacteristicValue		List of objects	
	value	Double	Numeric value of frequency (e.g., 3625)
	unitOfMeasure	String	Megahertz [MHz]

Table 4-40: Bandwidth Characteristic for RAN Resource Specification

Parameter		Type	Description
name		String	bandwidth
description		String	The bandwidth provided by the radio resource
resourceSpecCharacteristicValue		List of objects	
	value	Double	Numeric value of bandwidth (e.g., 50)
	unitOfMeasure	String	Megahertz [MHz]

Table 4-41: Duplex mode Characteristic for RAN Resource Specification

Parameter		Type	Description
name		String	duplexMode
description		String	Description of the spectrum duplexing mode to be employed
resourceSpecCharacteristicValue		List of objects	
	value	String	Duplex mode value (e.g., TDD or FDD)

Table 4-42: Technology Characteristic for RAN Resource Specification

Parameter		Type	Description
name		String	technology
description		String	The technology to use with the spectrum resource
resourceSpecCharacteristicValue		List of objects	
	value	String	Technology value (e.g., 4G, 5G, Wi-Fi)

Table 4-43: Band Characteristic for RAN Resource Specification

Parameter		Type	Description
name		String	band
description		String	The standard band number
resourceSpecCharacteristicValue		List of objects	
	value	Integer	Numeric value of band number (e.g., 78)

Table 4-44: Transmission power Characteristic for RAN Resource Specification

Parameter		Type	Description
name		String	txPower
description		String	The maximum radiated power, including the maximum beamforming gain if available (EIRP)
resourceSpecCharacteristicValue		List of objects	
	value	Double	Numeric value of maximum radiated power e.g., 32.0)
	unitOfMeasure	String	dBm

4.3.6 Spectrum (licensed & non-licensed) characteristics information model

The set of specific characteristics used to define additional features for spectrum resources, following the information model of *ResourceSpecCharacteristic* in [7], is described fromTable 4-45 to Table 4-51.

Table 4-45: Start downlink frequency Characteristic for Spectrum Resource Specification

Parameter		Type	Description
name		String	startFreqDl
description		String	The start downlink frequency of the spectrum resource
resourceSpecCharacteristicValue		List of objects	The start downlink frequency of the spectrum resource
	value	Double	Numeric value of frequency (e.g., 3600)
	unitOfMeasure	String	Megahertz [MHz]

Table 4-46: End downlink frequency Characteristic for Spectrum Resource Specification

Parameter		Type	Description
name		String	endFreqDl
description		String	The end downlink frequency of the spectrum resource
resourceSpecCharacteristicValue		List of objects	
	value	Double	Numeric value of frequency (e.g., 3650)
	unitOfMeasure	String	Megahertz [MHz]

Table 4-47: Start uplink frequency Characteristic for Spectrum Resource Specification

Parameter		Type	Description
name		String	startFreqUl
description		String	The start uplink frequency of the spectrum resource
resourceSpecCharacteristicValue		List of objects	
	value	Double	Numeric value of frequency (e.g., 3600)
	unitOfMeasure	String	Megahertz [MHz]

Table 4-48: End uplink frequency Characteristic for Spectrum Resource Specification

Parameter		Type	Description
name		String	endFreqUl
description		String	The end uplink frequency of the spectrum resource
resourceSpecCharacteristicValue		List of objects	A list of central frequency values
	value	Double	Numeric value of frequency (e.g., 3650)
	unitOfMeasure	String	Megahertz [MHz]

Table 4-49: Duplex mode Characteristic for Spectrum Resource Specification

Parameter	Type	Description

name		String	duplexMode
description		String	Description of the spectrum duplexing mode to be employed
resourceSpecCharacteristicValue		List of objects	
	value	String	Duplex mode value (e.g., TDD or FDD)

Table 4-50: Technology Characteristic for Spectrum Resource Specification

Parameter		Type	Description
name		String	technology
description		String	The technology to use with the spectrum resource
resourceSpecCharacteristicValue		List of objects	
	value	String	Technology value (e.g., 4G, 5G)

Table 4-51: Band Characteristic for Spectrum Resource Specification

Parameter		Type	Description
name		String	band
description		String	The standard band number
resourceSpecCharacteristicValue		List of objects	
	value	Integer	Numeric value of band number (e.g., 78)

4.4 Data Lake Information Models

4.4.1 Data returned by Data Lake RegisterOperator

When an Operator calls the RegisterOperator() interface, one of the returned fields is *availableResources*. This field returns a list of resources (e.g., existing pipelines, selected tools, topics) that the Data Lake provides to the newly registered Operator. This includes a list of URLs to allow the Operator to directly access the APIs of the Kubernetes, the Data Store, the Kafka, and other components composing the Data Lake. For example, this allows the Operator to access the Data Store bucket into which the Operator's data is stored. The *availableResources* field is provided in json format and is easily extensible.

Table 4-52: Data Lake availableResources

Parameter		Type	Description
pipelines		**List of pipelines**	The pipelines that are pre-defined in the Data Lake and available for use.
	name	String	Descriptive name of the pipeline. Example: “resourceMetricsIngestPipeline”
	pipelineId	String	The unique ID assigned to the pipeline.
topics		**List of topics**	The Kafka topics that are pre-defined in the Data Lake and available for use.
	name	String	Descriptive name of the topic. Example: “resourceMetricsIngestTopic”
	topicId	String	The actual Kafka- assigned name of the topic.
urls		**List of URLs**	URLs of services in the Data Lake available for use.
	name	String	Descriptive name of the url. Example: “k8sUrl”, “objectStoreUrl”, “kafkaUrl”
	urlValue	String	The actual url of a service.

4.4.2 Data returned by Data Lake listPipelines

The *listPipelines*() interface returns a *pipelines* field that provides a list of pipelines and their properties in json format.

Table 4-53: listPipelines output structure

Parameter		Type	Description
pipelines		**List of pipelines**	The pipelines that were defined by the user.
	pipelineId	String	The name of the pipeline.
	inputTopic	String	The Kafka input topic for the pipeline.
	outputTopic	String	The Kafka output topic for the pipeline.
	workflowTemplate	json	The Argo template for the pipeline.

4.4.3 Additional data related aspects

4.4.3.1 *Kafka topics*

Kafka topics are used for Operators to send data to a pipeline, to receive data from a pipeline, and for other messaging required in 5GZORRO. Kafka documentation can be found at: https://kafka.apache.org/documentation/.

4.4.3.2 *Pipelines template definition*

The format to define a pipeline (also called a workflow) is taken from the Argo tool. Details of defining pipelines can be found at: https://argoproj.github.io/argo/workflow-templates/.

4.4.3.3 *Kubernetes interface*

An Operator may need to define workflows that do not fit in with the 5GZORRO Data Lake model (pipelines). In this case, the Operator may directly access the underlying Kubernetes support to deploy specialised analytics components. The Kubernetes API can be found at: https://kubernetes.io/docs/reference/

4.4.3.4 *Data Integrity*

As a general rule, when data is sent to the Data Lake, it should be accompanied with a metadata field that contains a signed hash of the content in order to be able to check authenticity and correctness of the data.

4.5 Operational Data Models

This section contains the information models passed to the Data Lake by various 5GZORRO components for storage and/or processing. These models represent the stakeholders that make use of the 5GZORRO applications but may also be specifications of products offered by the stakeholders of the platform, or informational data exchanged between components.

4.5.1 VNF Monitoring

Table 4-54 shows the information related to the monitoring of a running VNF that can be collected and stored in the Data Lake.

Table 4-54: VNF Monitoring information

Parameter	Type	Description
transactionID	String	Unique id of the transaction

productID	String	Unique id of the service offered in the SLA
resourceID	String	Id of the resource as offered in the service
instanceID	String	Unique id of the instance where the service is offered
metric	String	Possible values: • cpu_utilization • average_memory_utilization • disk_in/out_operations • packet_in/out_loss
value	Number	Numerical value retrieved from the NFVO [5] monitoring platform

The initial set of IDs is required to uniquely identify the VNF itself in different contexts such as Marketplace, Service and Resource catalogues and running enviroments (through InstanceID), while the last two fields define the metric type and its value respectively. The set of metrics proposed is aligned with the capability in terms of performance management offered by OSM [34] which collects information directly from the VIM. In this sense, the metrics collected reflect the VNF usage of resources in terms of computation, memory and network resource consumption. This information is used by the 5GZORRO platform to determine if a network service/slice (which runs one or more VNFs) requires more resources in order to timely react accordingly.

4.5.2 E-Licensing Usage

Table 4-55 specifies the e-Licensing tracking information that is persisted in the data lake. These actions are collected by the eLM and represent the live usage performed by purchasers of the software. They have all the information related to the metrics that are being observed in each instance that compose the Network Service. The set of metrics to collect are deducted from the type of license agreement accepted at the time of signing the contract, resulting in a watcher. The purchasers are identified by the product ID since they are part of the transaction executed in the marketplace.

Table 4-55: e-Licensing Usage information

Parameter	Type	Description
timestamps	object	Contains datetimes objects for the keys: createdAt, updatedAt, timeStart, timeEnd
instanceId	String	Identification of the NF instance that originated this action
nfvLevel	Enum	Type of xNF related to this action: VNF, CNF, NS
offeringPriceId	String	Identification of the productOfferingPrice that originated this action
offeringId	String	Identification of the productOffering that originated this action
triggerKind	Enum	Reason for the delivery of the action; ERROR, EXPIRATION, SCHEDULED
Status	Enum	Current state of the action; RUNNING, SENT
Metric	Object	Metric value obtained according to the productOfferingPrice containing value and unit keys

4.5.3 RAN Monitoring

The RAN monitoring information in the datalake serves as a proof for RAN providers and consumers of the status of the radio infrastructure. The RAN Controller in the 5GZORRO platform continuously monitors its supervised RAN infrastructure and pushes the telemetry information to the platform. The RAN monitoring information is given at a base station granularity as structured in Table 4-56. As it can be seen, each base station is identified by its resource ID in the 5GZORRO platform. RAN data is structured on two differentiated levels. On one hand, the information model contains relevant physical parameters, including the location of the base station, its radio technology, the operation band, the exact starting and ending frequencies of the downlink and uplink bands, and the transmission power of the base station. On the other hand, the RAN monitoring data model includes the list of services that are instantiated on the base station. This information includes, for each service, the instance ID, product ID and transaction ID in the 5GZORRO platform, and the real percentage of downlink and uplink radio resource of the base station allocated to the service.

Table 4-56: RAN Monitoring information

Parameter		Type	Description
productID		String	Unique id of the resource offered in the SLA
resourceID		String	Unique id of the resource as offered in the service
address		GeographicalAddress object [12][37]	Location (address and geographical coordinates) of the radio station
technology		String	The base station radio technology: Wi-Fi, 4G, 5G
operationBand		String	The base station operation band (cellular) or channel (Wi-Fi)
startDlFrequency		Number	Starting downlink frequency (MHz)
startUlFrequency		Number	Starting uplink frequency (MHz)
endDlFrequency		Number	Ending downlink frequency (MHz)
endUlFrequency		Number	Ending uplink frequency (MHz)
txPower		Number	Transmission power (dBm)
services		List of objects	Information of the services deployed on the base station
	instanceID	String	Unique id of the instance where the service is offered
	transactionID	String	Unique id of the transaction
	dlQuota	Number	Percentage of the total downlink radio resources allocated to the service
	ulQuota	Number	Percentage of the total uplink radio resources allocated to the service

4.5.4 SLA Breach Notification

The Breach Notification message is JSON formatted and contains the fields of the SLA it refers to. These are the transactionID, productID, resourceID, instanceID, ruleID and metric. These are complimented with the prediction value, as well as the date of the prediction and the date the prediction refers to. This message is not stored in the Data Lake as other information but rather pushed to a Data Lake message queue.

Table 4-57: SLA Breach Notification

Parameter	Type	Description
slaID	String	Unique internal id of the SLA
transactionID	String	Unique id of the transaction
productID	String	Unique id of the service offered in the SLA
resourceID	String	Id of the resource as offered in the service
instanceID	String	Unique id of the instance where the service is offered
ruleID	String	String literal of the rule of the SLA
metric	String	URL of the rule
value	Number	Predicted value of the violation
datetimeViolation	Timestamp	Date when the violation will approximately take place
datetimePrediction	Timestamp	Date when the prediction was generated

4.5.5 Trust and Reputation

Owing to the fact that no trust information models are available in state of the art, Table 4-58, Table 4-59 and Table 4-60 propose a novel information model considering the 5G-enabled Trust and Reputation Management Framework (5G-TRMF) capabilities. In particular, Table 4-58 represents a subset of general characteristics that may be linked to any trust instance, regardless of the type of offer. Table 4-59 depicts information with respect to the service or resource provider. Therefore, this information is obtained from third-party recommendations. Finally, Table 4-60 introduces trustworthy data associated with the trustor who is the entity in charge of determining the most reliable provider to establish a new connection.

Table 4-58: Trust and Reputation Management Framework Instance Information Model

Parameter	Type	Description
trustorDID	String	Unique identifier for a resource or service consumer.
trusteeDID	String	Unique identifier for a resource or service provider.
trustValue	Double	Current trust value assigned.
currentInteractionNumber	Int	Number of interactions between the trustor and the trustee.
initEvaluationPeriod	TimeStamp	The time when trust value was generated.
endEvaluationPeriod	TimeStamp	The time when trust value will be over and has to be reassigned if required.

Table 4-59: Trustee Entity Information Model

Parameter	Type	Description
trusteeDID	String	Unique identifier for a resource or service provider.
offerDID	String	Unique identifier for a particular product offer of the provider.
type	String	Kind of offer (RAN, spectrum, VNF/CNF, slice, or edge)
trusteeSatisfaction	Double	Truestee's satisfaction after x interactions with other providers.

Table 4-60: Trustor Entity Information Model

Parameter	Type	Description
trustorDID	String	Unique identifier for a resource or service consumer.
trusteeDID	String	Unique identifier for a resource or service provider.
offerDID	String	Unique identifier for a particular product offer.
type	String	Kind of offer (RAN, spectrum, VNF/CNF, slice, or Edge)
history	List (double)	Set of trust evaluations about an entity.
directParameters	List of key-value features	Dictionary with direct trust data to calculate trust level.
directWeighting	Double	Direct weighting parameter.
userSatisfaction	Double	Internal assessment of the service or resource provided by a stakeholder (trustor).
***provider**Satisfaction*	Double	Trustor satisfaction in a third-party provider (trustee).
PSWeighting	Double	Weighting factor ∊ [0,1], PS + OS = 1
offerSatisfaction	Double	Trustor satisfaction in a particular kind of offer of a third-party provider.
OSWeighting	Double	Weighting factor ∊ [0,1], PS + OS = 1

providerReputation	List (double)	Set of previous trust evaluations about a provider.
offerReputation	List (double)	Set of previous trust evaluations about a specific kind of offer of a provider.
availableAssets	Integer	The available assets (services and resources) of the trusteeDID when the trustor is determining the reputation.
totalAssets	Integer	The total assets of the trusteeDID when the trustor is determining the reputation (active and inactive).
availableAssetLocation	Integer	The available assets of the trusteeDID, at a specific location, when the trustor is determining the reputation.
totalAssetLocation	Integer	The total assets of the trusteeDID, at a specific location, when the trustor is determining the reputation (active and inactive).
managedViolations	Integer	The total number of predicted SLA violations that were finally managed successful, associated with the trusteeDID.
predictedViolations	Integer	The total number of predicted SLA violations associated with the trusteeDID.
executedViolations	Integer	The total number of predicted SLA violations that were finally managed unsuccessful (violation), associated with the trusteeDID.
nonpredictedViolations	Integer	The number of SLA violations that were not predicted and turned out to be SLA violations, associated with the trusteeDID.
consideredOffers	Integer	The number of offers considered by the Smart Resource and Service Discovery (SRSD), for a particular type of offer, from the trusteeDID when trustor is determining the reputation.
consideredOfferLocation	Integer	The available number of a particular offer type from the trusteeDID when trustor is determining the reputation.
totalOfferLocation	Integer	The number of offers considered by the Smart Resource and Service Discovery (SRSD) for a particular type of offer from the trusteeDID, at a specific location, when trustor is determining the reputation.
managedOfferViolations	Integer	The available number of a particular offer type from the trusteeDID, at a specific location, when trustor is determining the reputation.
predictedOfferViolations	Integer	The total offer number (for a particular kind of offer) of predicted SLA violations that were finally managed successful, associated with the trusteeDID.
executedOfferViolations	Integer	The total offer number (for a particular kind of offer) of predicted SLA violations associated with the trusteeDID.
nonpredictedOfferViolations	Integer	The total offer number (for a particular kind of offer) of predicted SLA violations that were finally managed unsuccessful (violation), associated with the trusteeDID.
interactionNumber	Integer	The total offer number (for a particular kind of offer) of SLA violations that were not predicted and turned out to be SLA violations, associated with the trusteeDID.

totalInteractionNumber	Integer	Number of interactions carried out by the trusteeDID with the other domains.
feedbackNumber	Integer	Number of feedbacks made by other providers about the trusteeDID.
feedbackOfferNumber	Integer	Trustor satisfaction in a third-party provider (trustee).
location	List of GeographicalAddress objects [12][37]	It constitutes a group of GeographicalAddress.
validFor	TimePeriod	The period for which this resource or service is valid.
indirectParameters	List of key-value features	Dictionary with indirect trust data to calculate trust level.
recommendationWeighting	Double	Recommender's weighting parameter(s).
recommendations	List of recommendations	Set of recommendation about a third entity from one or more external entities.
recommender	String	Unique identifier for a recommender
trust_value	Int	Final trust score ranged from 0.0 to 1.0.
recommendation_trust	Double	Trust level of the trustor in the recommendation.
recommendation_total_number	Int	Number of recommendations of a given trustee.
average_recommendation	Double	Average of all recommendations.
last_recommendation	Double	The last recommendation.
credibility	Double	Factor that determines how accurate the recommendations are.
satisfaction	Double	Trustor satisfaction in a specific trustee.
transactionFactor	Double	Intra or inter-domain trust score and parameterTuple propagation (0 means intra, 1 means inter)
communityFactor	Double	It indicates the triggers to recompute trust score.
trustPropagation	Double	It identifies different evaluation algorithms.
reward_and_punishment	List of triggers	Increase or decrease score based on current events such as security, time-decay, etc.
trust_evaluation	List of algorithms	It identifies different evaluation algorithms such as PeerTrust reputation model.

4.6 Governance Information Models

4.6.1 Notifications

Notification Method (abstract)

Parameter	Type	Description
type	NotificationTypeEnum	The kind of notification Enum Values: EMAIL

Email Notification (extends NotificationMethod)

Parameter	Type	Description
type	NotificationTypeEnum	The kind of notification, final value of EMAIL
distributionList	List<string>	List of email addresses to be notified of marketplace events

4.6.2 MembershipStatus

Parameter	Type	Description
stakeholderId	String	DID of the stakeholder
status	MembershipStatusEnum	Current status of the stakeholder membership Enum Values: PENDING|ACTIVE|REVOKED | REJECTED
statusUpdated	DateTime	Date & Time of the last status change

4.6.3 Member

Parameter	Type	Description
stakeholderId	String	DID of the stakeholder
legalName	String	The legal name of the stakeholder
address	String	Address of the stakeholder

4.6.4 GovernanceProposal

Parameter	Type	Description
stakeholderId	String	DID of the stakeholder
status	ProposalStatusEnum	Current status of the proposal Enum Values: PROPOSED|APPROVED|REJECTED
actionType	ActionTypeEnum	The type of proposal
actionParams	ActionParams	The action parameters submitted with the proposal
statusUpdated	DateTime	Date & Time of the last status change

4.6.5 ActionParams

Parameter	Type	Description
entityIdentityId	String	The DID of the entity that is the subject of the proposal Stakeholder, SLA, Template
evidence	String	Any supporting evidence

4.6.6 LegalProseTemplate

LegalProseTemplate will be used to model SLA and Licence terms as well as the events (Violations) that an agreement or SLA clause may emit.

Parameter	Type	Description
id	String	Proposal ID of the template
did	String	DID of the template
name	String	A short name that encapsulates the nature of the template
description	String	A description of the template's contents and intention
category	TemplateCategory	The category of the template
created	DateTime	The date & time the template was created

status	String	String that denotes the current governance status of the template PROPOSED | ACTIVE | REJECTED | ARCHIVED
file	LegalProseTemplateFile	Zip file containing the templace
statusUpdated	DateTime	The date & time the template status was updated
archived	DateTime	The date & time the template was archived

4.7 Monitoring Data Information Models

Monitoring information is provided in postMonitoringData() using json format. An individual resource item monitoring information contains the following fields:

Table 4-61: Resource Monitoring information

Parameter	Type	Description
resourceId	String	Unique ID of resource for which we are collecting monitoring data.
metricName	String	Name of metric being monitored.
metricValue	String	Value of metric being monitored.
timeStamp	DateTime	The time at which the metric was measured.

The resourceID represents the resource for which we are collecting monitoring data. If the resource used to be part of one SLA and is now part of another SLA, it may make sense to use a distinct resourceID for the different epochs, so that data from one epoch is not used to influence decisions in another epoch.

The productID might be the unique service offered in the SLA identifier to which this resource is dedicated and is used to look up all relevant monitoring data for that reference.

The monitoring data provided to postMonitoringData() should be a list of such structures, with one entry per metric being recorded.

When looking up aggregated monitored data by productID, the following structure is used:

Table 4-62: Lookup Monitoring Data Structure

Parameter	Type	Description
resourceId	String	Unique ID of resource for which we are collecting monitoring data.
metricName	String	Name of metric being monitored.
metricValue	String	Value of metric being monitored.

A list of such structures is returned when calling the lookupMetricsByReference() interface.

5 Conclusions

This report delivers the detailed architectural design for the 5GZORRO Service layer solutions, specifically, the Governance Services, the Trustworthy Marketplace Services, and the Analytics & Intelligence Services. The report covers the component design of all the solution modules, their interfaces, data models, as well as the key interactions between the modules that implement the envisaged Zero-Touch SLA Smart Contract, resource discovery, allocation and provisioning services offered by 5GZORRO platform.

The architecture presented herein extends the high-level architecture of the 5GZORRO platform delivered in D2.4 [1], focusing on components, services, and interactions delevoped in WP3. Aligned with D2.4 [1], this document provides the next level of detail, specifying internal build-up of the 5GZORRO Service layer solution components.

This deliverable is a standalone presentation of the final design of 5GZORRO Service layer solution. It updates and replaces the preliminary version delivered in D3.1, following the architectural refinements reflected in D2.4 [1] and informed by the implementation, prototyping and integration activities performed in the second period of the project. This deliverable will serve as a basis for the final deliverable of WP3, D3.2, that will present the software prototypes created according to the design presented here. In addition, this design will continue to provide guiding principles for the ongoing integration and validation work of WP5 (*Validation through Use Cases*).

6 References

[1] D2.4: Final design of the 5GZORRO Platform for Security & Trust

[2] D2.1: 5GZORRO Use Cases and Requirements Definition

[3] D4.4: Final Design of Zero Touch Service Management with Security and Trust Solutions

[4] Gartner, Market Guide for AIOps Platforms, Published 7 November 2019 – ID G00378587

[5] ETSI GS NFV-IFA 011 V4.1.1, "Network Functions Virtualisation (NFV) Release 4"

[6] ETSI, Zero-touch network and Service Management and Orchestration; VNF Descriptor and Packaging (ZSM); Means of Automation, GR ZSM 005 - V1.1.1, May 2020

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

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

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

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

[11] SLA Management API REST Specification, TM Forum Specification, TMF623, Release 14.5.1, June 2015.

[12] Geographic Address Management API REST Specification, TM Forum Specification, TMF673, Release 16.0.1, May 2016.

[13] Decentralized Identifiers (DIDs) v1.0. Core architecture, data model, and representations. W3C Proposed Recommendation 03 August 2021. https://www.w3.org/TR/did-core/

[14] Sporny, M., Longleyy, D., and Chadwick, D. Verifiable Credentials Data Model 1.0. Expressing verifiable information on the Web. W3C Recommendation 19 November 2019. Available online: https://www.w3.org/TR/vc-data-model/

[15] Sovrin Fundation. Available online: https://sovrin.org/

[16] Hyperledger URSA. Available online: https://www.hyperledger.org/use/ursa

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

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

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

[20] Universal Resolver. Available online: https://github.com/decentralized-identity/universal-resolver

[21] Universal Registrar. Available online: https://github.com/decentralized-identity/universal-registrar

[22] Identity Hub. Available online: https://identity.foundation/working-groups/storage-compute.html

[23] DID Communication. Available online: https://identity.foundation/working-groups/did-comm.html

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

[25] Protocol on-the-fly Concept. Available online: https://rethink-project.github.io/specs/concepts/protofly/

[26] reTHINK (Trustful hyper-linked entities in dynamic networks). Available online: https://rethink-project.eu/

[27] Hyperledger ARIES RFC 0160 connection protocol. Available online: https://github.com/hyperledger/aries-rfcs/tree/master/features/0160-connection-protocol

[28] Hyperledger ARIES RFC 0453 issue credential protocol. Available online: https://github.com/hyperledger/aries-rfcs/tree/master/features/0453-issue-credential-v2

[29] Hyperledger ARIES RFC 0454 present proof protocol. Available online: https://github.com/hyperledger/aries-rfcs/tree/master/features/0454-present-proof-v2

[30] R3 Corda. Available Online: https://www.corda.net/

[31] Quorum. Available Online https://consensys.net/quorum/

[32] Hyperledger Fabric. Available Online https://www.hyperledger.org/use/fabric

[33] Accord Project. Available Online: https://accordproject.org/

[34] OSM - Monitoring and autoscaling, https://osm.etsi.org/docs/user-guide/05-osm-usage.html#monitoring-and-autoscaling

[35] Kaloxylos A., Gavras A., Camps Mur D., Ghoraishi M, Hrasnica H. (2020). 5G PPP Whitepaper, AI and ML – Enablers for Beyond 5G Networks. Zenodo. https://doi.org/10.5281/zenodo.4299895

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

[37] TM Forum, Information Framework (SID) https://www.tmforum.org/resources/standard/information-framework-sid-poster-v21-5/

[38] AWS Lake-Formation. https://aws.amazon.com/lake-formation

[39] GCP Cloud Storage as Data Lake. https://cloud.google.com/architecture/build-a-data-lake-on-gcp

[40] Azure Data Lake Storage. https://azure.microsoft.com/en-us/services/storage/data-lake-storage/

[41] Data Lake Solution Patterns for Big Data Use Cases. https://blogs.oracle.com/bigdata/post/4-data-lake-solution-patterns-for-big-data-use-cases

[42] Cloudera Data Platform. https://www.cloudera.com/products/cloudera-data-platform.html

[43] Zaloni Arena Data Governance Platform. https://www.zaloni.com/arena-overview/

[44] Teradata Data Lake Solutions. https://www.teradata.com/Cloud/Data-Lake

[45] Red Hat Data Services bog. What is a data lake. https://www.redhat.com/en/topics/data-storage/what-is-a-data-lake

[46] Open Data Hub. https://opendatahub.io/

[47] Apache Hadoop. https://hadoop.apache.org/

[48] Apache Spark. https://spark.apache.org/

[49] Kubernetes documentation. What is Kubernetes? https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

[50] Argo Workflows. https://argoproj.github.io/workflows

[51] Ceph project. https://ceph.io/en/

[52] Apache Kafka. https://kafka.apache.org/

[53] Kubernetes API Reference. https://kubernetes.io/docs/reference/

[54] Amazon Simple Storage Service API Reference. https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html

[55] Apache Kafka API. https://kafka.apache.org/documentation/#api

[56] Argo Workflows API. https://argoproj.github.io/argo-workflows/rest-api/

7 Abbreviations

AIOps	Artificial Intelligence for IT operations
CFS	Customer Facing Services
CNF	Cloud Native Function
DID	Decentralized Identifier
DIF	Decentralised Identity Foundation
DLT	Distributed Ledger Technology
DPKI	Decentralised Public Key Infrastructure
EC	European Commission
eLM	e-Licensing Manager
FaaS	Function as a Service
ISSM	Intelligent Slice and Service Manager
K8s	Kubernetes
LCM	Lifecycle Manager
MANO	Management and Orchestration
NBI	Northbound Interface
NFV	Networks Function Virtualization
NFVO	Networks Function Virtualization Orchestrator
NPM	Node Package Manager
NS	Network Service or Network Slice depending on the context
NSI	Network Service Instance
NSM	Network Service Mesh
PO	Product-Offering
RAN	Radio Access Network
RFS	Resource Facing Services
RSOC	Resource and Service Offering Catalogue
SC	Smart Contract
SDO	Standards Development Organization
SM	Service Mesh
UTXO	Unspent Transaction Output
VC	Verifiable Claim
VDU	Virtual Deployment Unit
VNF	Virtual Network Function
VNFM	Virtual Network Function Manager
W3C	World Wide Web Consortium
WG	Working group
WP	Work Package
ZSM	Zero Touch Service Management

<END OF DOCUMENT>