5GZORRO

Grant Agreement 871533

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

D3.1: 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: **871533**	Project Acronym: **5GZORRO**	Project title: **Zero-touch security and trust for ubiquitous computing and connectivity in 5G networks.**

Lead Beneficiary: **IBM**	Document version: **V1.3**

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

Deliverable title: **D3.1: Design of the evolved 5G Service layer solutions**

Start date of the project: **01/11/2019** **(duration 30 months)**	Contractual delivery date: **31.01.2021**	Actual delivery date: **31.01.2021**

Editor(s) K. Meth (IBM)

List of Contributors

Participant	Short Name	Contributor
IBM Israel Science and Technology	IBM	K. Meth, K. Barabash, D. Breitgand
Nextworks	NXW	Pietro Giuseppe Giardina, Juan Brenes, M. De Angelis, G. Carrozzo
Fundaciò i2CAT	I2CAT	A. Fernández-Fernández, C. Herranz Claveras, J. Fernández Hidalgo, M. S. Siddiqui, A. Betzler, L. A. Ochoa, A. Ruiz Amate
Telefonica Investigación Investigacióny Desarrollo	TID	D. R. López, C. Rodríguez Cerro
Ubiwhere	UW	Pedro Diogo
Fondazione Bruno Kessler	FBK	
Universidad de Murcia	UMU	G. Martínez Pérez, M. Gil Pérez, P. M. Sánchez Sánchez, J.M. Jorquera Valero
Bartr Holding Limited	BTL	James Taylor
Altice Labs	ALB	Paulo Chainho, Bruno Santos, Miguel Silva
Intracom	ICOM	Alexios Lekidis, Vasileios Theodorou, Theodoros Bozios, Marinela Mertiri
Atos Spain	ATOS	Fernando Bravo
Malta Communications Authority	MCA	Jean-Marie Mifsud, Antoine Sciberras

List of Reviewers

Participant	Short Name	Contributor
Intracom	ICOM	M. Mertiri
Bartr Holding Limited	BTL	J Taylor
Nextworks	NXW	G. Carrozzo
Fundaciò i2CAT	I2CAT	M. S. Shuaib

Change History

Version	Date	Partners	Description/Comments
0.0	25-09-2020	IBM, NXW	Table of Contents (and editing assignments)
0.1	01-12-2020	IBM, BTL, I2CAT, ICOM	Background sections; initial technical content
0.2	15-12-2020	IBM, ALB, NXW, i2CAT, ICOM	SW Module component specifications

0.3	30-12-2020	IBM	Additional SW Module component specifications
0.4	05-01-2021	UW	Additional SW Module component specifications
0.5	07-01-2021	BTL, ICOM, ALB	Additional SW Module component specifications
0.6	12-01-2021	UMU, IBM, NXW, i2CAT, ICOM	Refinements
0.7	14-01-2021	i2CAT, BTL, ICOM, IBM	Refinements, formatting
0.8	17-01-2021	TID, MCA, ATOS, ALB, IBM	Refinements, formatting
0.9	19-01-2021	UW, BTL, i2CAT, IBM	Refinements, formatting; sent to internal reviewers
0.10	24-01-2021	ALB, BTL, ICOM, IBM	Reviewers comments; formatting
1.0	27-01-2021	All	Addressing reviewers' comments
1.1	28-01-2021	IBM	Final edits
1.2	29-01-2021	NXW, i2CAT	Final QA
1.3	31.01.2021	i2CAT	Final submission version

DISCLAIMER OF WARRANTIES

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

Table of Contents

List of Tables

List of Figures

Executive Summary

This deliverable specifies the architecture of data-driven solution for DLT and distributed intelligent resource discovery and management in use in 5GZORRO project.

The report contains the domain-specific systems architecture of DLT and 5G Operational Data Lake elements, and describes key interactions of their constituent modules to implement the envisaged Zero-Touch SLA Smart Contract, resource discovery, allocation and provisioning services offered by the 5GZORRO platform.

The 5GZORRO software platform is divided in four major software platforms, as established in deliverable D2.2: (i) *Marketplace*; (ii) *Governance*; (iii) *Cross-domain Analytics & Intelligence for AIOps*; (iv) *Zero-touch Service Management and Orchestration*. The aim of this deliverable is to describe the first version of the software platform design for the Marketplace and the Governance and the Cross-domain Analytics & Intelligence for AIOps, which provide the 5GZORRO platform the capabilities to establish a multi-domain marketplace of resources and services with automated service management capabilities.

Using these components, the 5GZORRO platform can support the following three 5GZORRO use cases:

1. Use case 1 – Smart Contracts for Ubiquitous Computing/Connectivity;
2. Use case 2 – Dynamic spectrum allocation;
3. Use case 3 – Pervasive virtualized Content Distribution Network (vCDN) Services.

The outputs of this deliverable serve as input for the implementation work that is being carried within the project and will documented in next deliverables D3.2 (*Prototypes of evolved 5G Service layer solutions*) and D4.3 (*Final prototype of Zero Touch Service Management with Security and Trust*). Performance evaluation and validation of the design artifacts described in this document will occur through workpackage WP5 (*Validation through Use Cases*).

1 Introduction

This document describes the architecture of data-driven solution for Distributed Ledger Technologies (DLT) and distributed intelligent resource discovery and management for the 5GZORRO software platform.

Based on the overall 5GZORRO architecture described in deliverable D2.2 [1], this document focuses into the per-domain and cross-domain management layers, and the boundary between them.

This design report contains domain-specific systems architecture of DLT and 5G Operational Data Lake elements to be used in 5GZORRO, and describes key interactions of their constituent modules to implement the envisaged Zero-Touch SLA Smart Contract, resource discovery, allocation and provisioning services offered by 5GZORRO platform.

1.1 Document scope and objectives

5GZORRO tries to address the following three use cases:

- Use case 1 – Smart Contracts for Ubiquitous Computing/Connectivity;
- Use case 2 – Dynamic spectrum allocation;
- Use case 3 – Pervasive virtualized Content Distribution Network (vCDN) Services.

As discussed in deliverable D2.2, to address these use cases, 5GZORRO incorporates solutions based on three novel concepts:

- Data-driven and Artificial Intelligence (AI) based solutions which can enable automatic and autonomous network operations following AI Operations (AIOps) paradigm.
- Distributed Ledger Technologies (DLT) which enable trust and security in multi-party end-to-end service/slice implementation.
- Cloud-Native technologies which once integrated into Software Defined Network (SDN) and Network Function Virtualisation (NFV) environments can increase the level of flexibility required by advanced 5G based services (e.g., scalability, resilience).

The combination of these three concepts is the basis for realization of the three main 5GZORRO innovations:

- Zero-touch/Automated Resource discovery using DLT/Blockchains.
- Intelligent 3rd party resource selection, request and access/usage.
- Trust establishment among multiple parties.

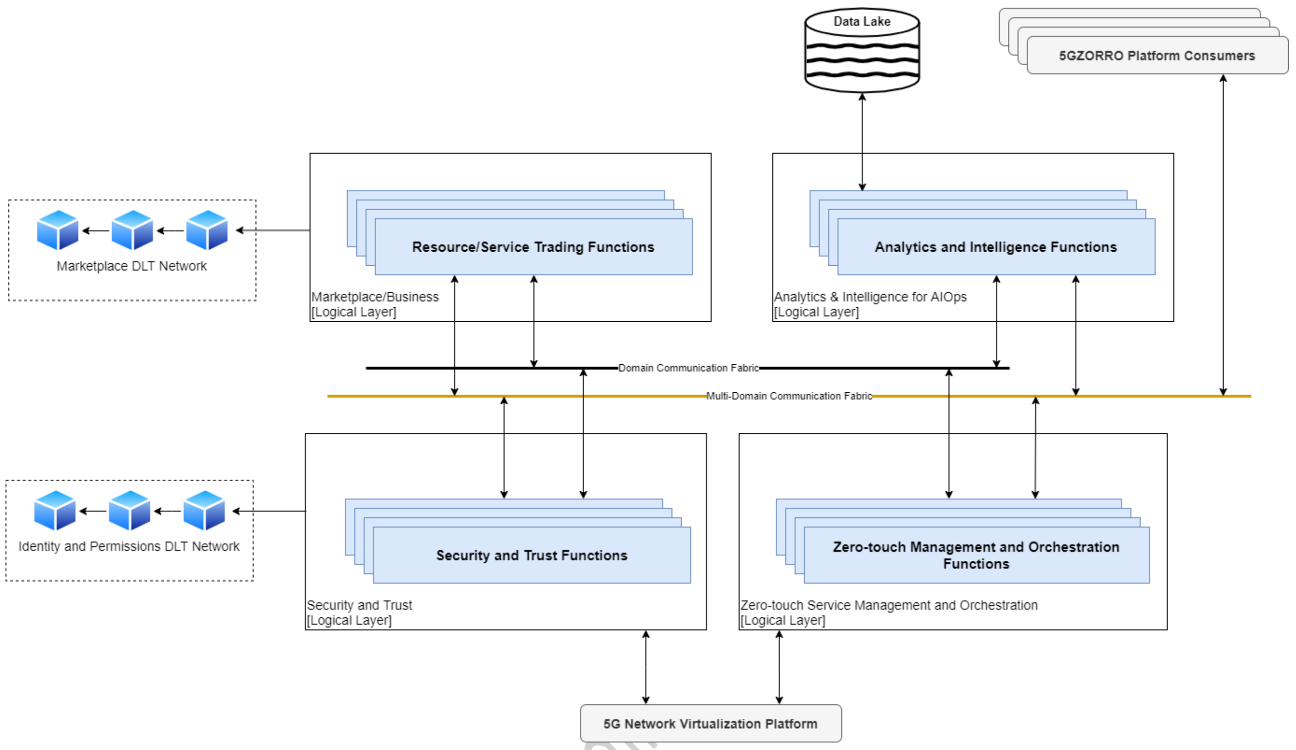
1.1.1 Deliverable Scope with respect to the 5GZORRO Architecture specification

A simplified high-level view of the 5GZORRO architecture is shown in Figure 1-1.

The 5GZORRO software platform is divided in four major software platforms, as established in D2.2: (i) Marketplace; (ii) Governance; (iii) Cross-domain Analytics & Intelligence for AIOps; (iv) Zero-touch Service Management and Orchestration. The Zero-touch Service Management and Orchestration embraces all the software modules involved in the service and Network Slice lifecycle management, both at the intra and the inter-domain layers, and the required components in order to interact with the underlying NFVI. This platform part is described in deliverable D4.1 [3].

The Marketplace holds the resource and service offers, containing the high-level descriptions of the resources and services together with the business level SLAs and the pricing mechanisms established for the offer. The Marketplace relies on different DLT technologies for a secure synchronization of the offers on the catalogue across domains, and for the establishment of smart contracts reflecting the offer acquisition. The Governance platform supplies the identity management and authentication and authorization functionalities across domains, and contains the catalogue of the legal statement templates to be attached to the resource and service offers in the Marketplace.

The Cross-domain Analytics & Intelligence for AIOps provides the software functionality to take automated decisions, based on AI control loops, to optimize the service deployment across multiple domains and to predict possible SLA outbreaks.



Data Lake
5GZORRO Platform Consumers
Marketplace DLT Network
Resource/Service Trading Functions
Marketplace/Business [Logical Layer]
Analytics and Intelligence Functions
Analytics & Intelligence for AIOps [Logical Layer]
Domain Communication Fabric
Multi-Domain Communication Fabric
Identity and Permissions DLT Network
Security and Trust Functions
Security and Trust [Logical Layer]
Zero-touch Management and Orchestration Functions
Zero-touch Service Management and Orchestration [Logical Layer]
5G Network Virtualization Platform

Figure 1-1: 5GZORRO High Level reference architecture

The scope of this deliverable is to describe the first version of the software platform design for the Marketplace, the Governance and the Cross-domain Analytics & Intelligence for AIOps. The aim is to provide more in depth details on the platform capabilities expected to establish a multi-domain marketplace of resources and services with automated service management capabilities.

Further to deliverable D2.2, this document fulfils the objective of providing the next level of design details to serve as input for software development activities within the workpackage WP3.

The document also defines a 5G-oriented DLT infrastructure and a 5G Operational Data Lake capable of supporting multi-party value layer for Smart Contracts and AI-based automated resource management.

1.1.2 Related 5GZORRO Project Objectives

The operative objectives addressed through this deliverable are:

- Design and implementation of a 5G DLT infrastructure to serve as a common enabler for the various 5GZORRO distributed services.
- Define and implement APIs and intelligent resource management components to build a shared operational data repository (i.e., the 5G Operational Data Lake) to enable innovative 5G management functionalities and services.
- Design and development of a Multi-Party Value Layer in which Smart Contracts can be defined and activated across multiple parties.
- Develop a distributed multi-party resource discovery (extended to spectrum), and intelligent resource brokerage & selection.

- Design and development of Smart Contract management to support service applications.

The below table describes the main technical objectives to realize the 5GZORRO marketplace for autonomous trading of 5G-orientated infrastructure and spectrum.

Table 1-1: 5GZORRO Technical Objectives in scope

5GZORRO Technical Objective	D3.1 Contribution
OBJ-3. Define a Smart Contract ecosystem anchored on a native distributed ledger to allow commercial and technical data provided by 3rd-party users to be standardised and mapped into Smart Contracts, which can be initiated "at will" between multiple untrusted parties.	• **Smart Contract system design**
OBJ-4. Define solutions for secure, automated and intelligent resource discovery, brokerage and selection, operation with SLA to facilitate workload offloading to 3rd-party resources supporting pervasive computing across multiple 5G domains.	• **AI-driven Resource Discovery & Management solution design**
OBJ-5. Define and prototype a secure shared spectrum market to enable real-time trading of spectrum allocations between parties that do not have a pre-established trust relationship.	• **Spectrum Market application design**

1.2 Document outline

The document is structured as follows:

- Section 2 covers the baseline Distributed Ledger Technology (DLT) platforms. Many of the 5GZORRO concepts are dependent on the DLT.
- Section 3 covers the software modules that provide Governance of a 5GZORRO deployment.
- Section 4 presents functionalities and components of the 5GZORRO Marketplace Platform, aimed to facilitate multi-party collaboration in dynamic 5G environments.
- Section 5 describes the modules that enable cross-domain analytics, including the cross-domain Data Lake.
- In Section 6 the data models used by the various interacting components are decribed.
- In Section 7 it is summarize how the presented material contributes to the 5GZORRO objectives and KPIs.
- Section 8 is the deliverable appendix in which examples of offer types Information Elements are collected.

2 Baseline DLT platforms

Based on a thorough review of the suitability of DLTs for the 5GZORRO platform, two categories of DLT requirements can be 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 [40], Hyperledger Fabric [42] and Quorum [41] 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 [22] 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 use cases, which would have involved significant effort to design and implement what is already offered out-of-the-box by Hyperledger Indy.

2.1 DLT for Smart Contracts and Resource offering

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 participants, oracles 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.

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

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 Relevant Entities and Modules

The Corda DLT infrastructure will be 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).

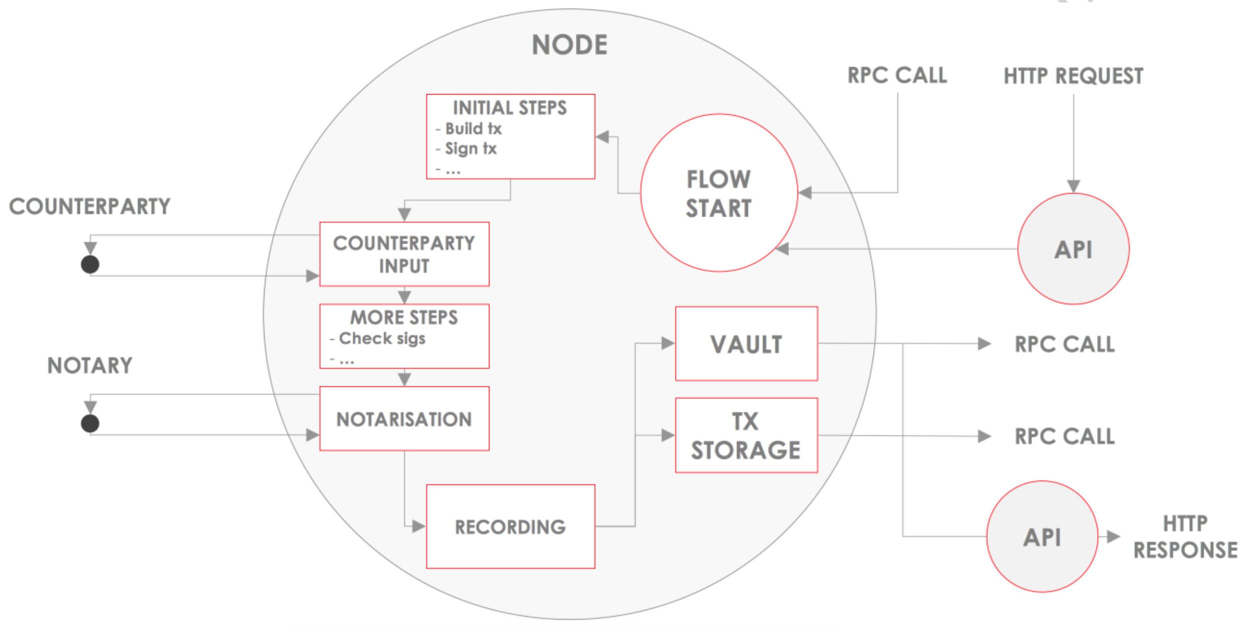


Figure 2-1: Corda Node

<u>Contract States & Vault</u>: 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 5GZORRO products, SLAs and Agreements.

<u>Contracts</u>: 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.

<u>Flows</u>: Flows will be 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.

<u>CorDapp & RPC</u>: CorDapps are distributed applications that run on each corda node. They encapsulate the flows, contracts and 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.

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

<u>Network Map (Cordite)</u>: 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.

<u>Participant Nodes</u>: 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 in order to discover other network participants, notaries and oracle services.

<u>Notary</u>: In order 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.

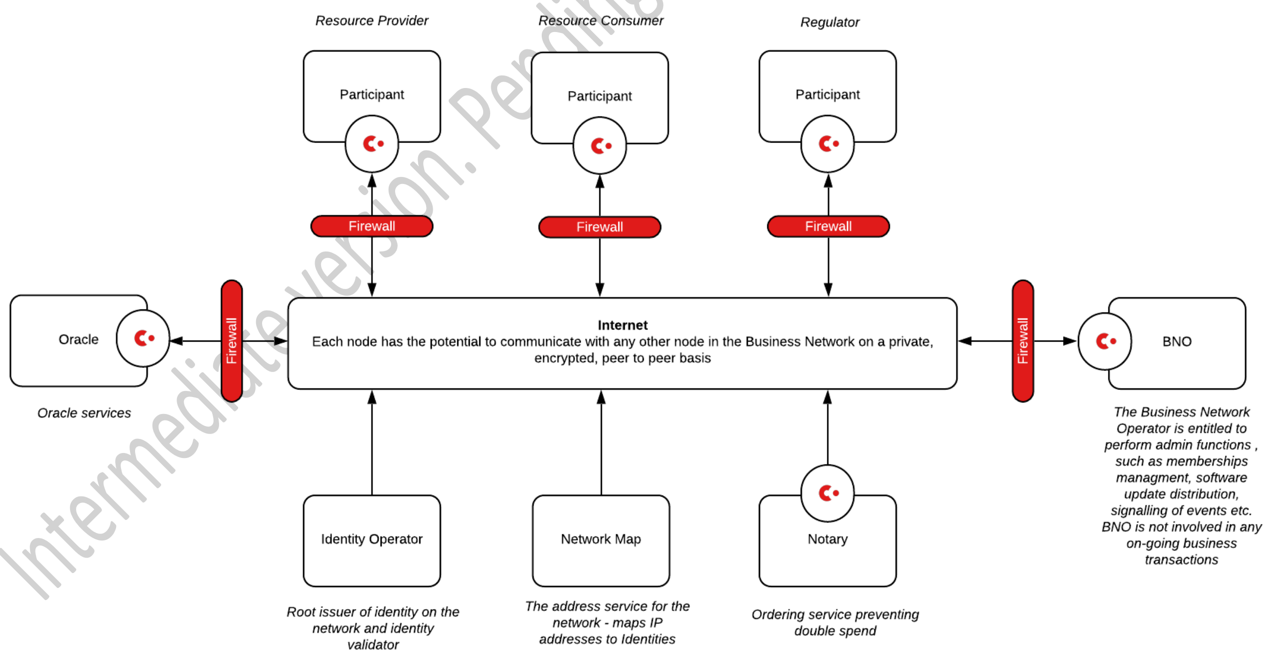

Resource Provider
Resource Consumer
Regulator
Participant
Participant
Participant
Firewall
Firewall
Firewall
Oracle
Firewall
Internet
Each node has the potential to communicate with any other node in the Business Network on a private, encrypted, peer to peer basis
Firewall
BNO
Oracle services
The Business 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
Network Map
Notary
Root issuer of identity on the network and identity validator
The address service for the network - maps IP addresses to Identities
Ordering service preventing double spend

Figure 2-2: High-level Corda Network Architecture

2.1.2 Oracles

There are several Oracle services required by the 5GZORRO DLT to provide access to off-chain data and external attestation to ledger facts. The following Oracles and associated use cases have been identified:

Resource Proxy Oracle

DEPLOYED BY: Governance Admin Operator

An oracle service will be deployed by Governance Admin operators for confirming the successful deployment of one or more resources under an agreement prior to entering the monitoring phase. This therefore allows the attestation to the fact that a resource has been provisioned successfully as an integral element of the ledger transaction. The oracle will utilise service descriptors found in the Decentralised Identifier (DID) document associated with the resource in order to determine whether it has been provisioned successfully and on successfully testing all endpoints will sign the transaction. In turn, the transaction can be validated and committed to the ledger.

DID Smart Contract Oracle

DEPLOYED BY: Governance Admin Operator

A trusted oracle service will be deployed by the network operator such that attestation can be provided as to the validity (verification of) verifiable credentials for certain transactions. This oracle will utilise a DID agent belonging to the operator for the purposes of verifying credentials. It will optionally also verify signed content as part of its attestation as needed. Example uses will be for verifying spectrum rights when a provider wishes to publish an offer in the marketplace, and verification of whether an SLA Violation being posted to the DLT by the permitted monitoring service declared in the contract.

2.2 DLT for distributed identities

As introduced in deliverable D2.2, 5GZORRO has adopted the W3C Decentralised Identifiers (DIDs)[18] and associated Verifiable Credentials (VCs)[19] 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.

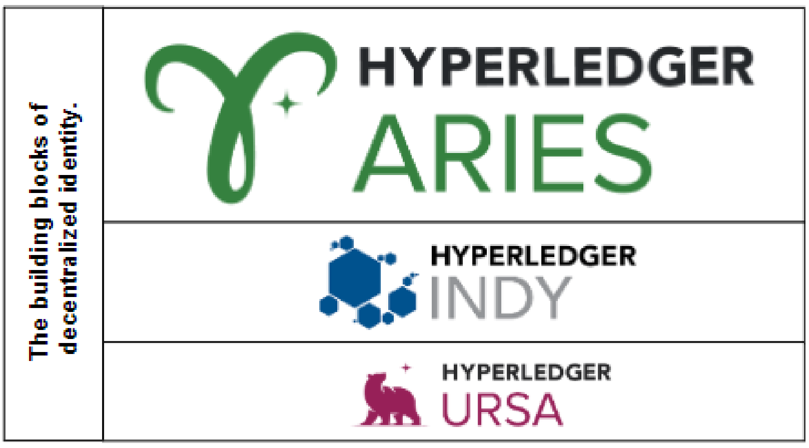


Figure 2-3: Hyperledger Identity "stack": URSA, INDY and ARIES

The Linux Foundation Hyperledger Identity "stack" was created from original Sovrin Foundation[20] code base and is comprised of three main projects presented in Figure 2-3:

- The **Hyperledger URSA** [21] 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**[22] 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**[23] 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)[24] is another initiative around decentralised identities aiming to enable interoperability between any DID independently of the DLT used.

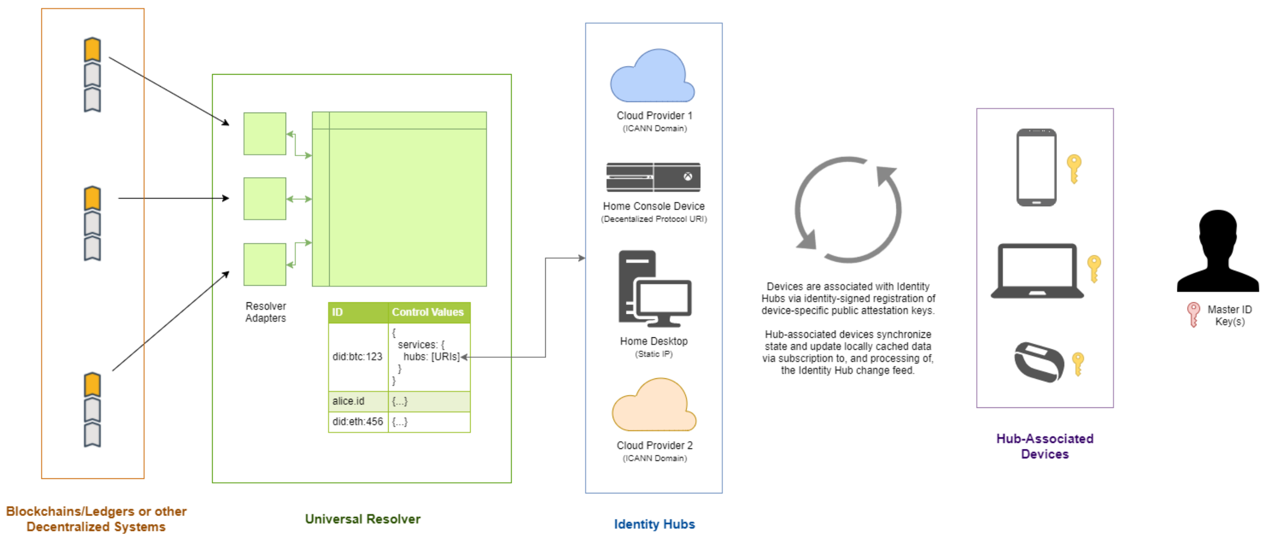

Resolver Adapters
ID
Control Values
did:btc:123
{ services: { hubs: [URIs] } }
alice.id
{...}
did:eth:456
{...}
Cloud Provider 1
(ICANN Domain)
Home Console Device
(Decentalized Protocol URI)
Home Desktop
(Static IP)
Cloud Provider 2
(ICANN Domain)
Devices are associated with Identity Hubs via identity-signed 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)
Blockchains/Ledgers or other Decentralized Systems
Universal Resolver
Identity Hubs
Hub-Associated Devices

Figure 2-4: Decentralised Identity Foundation projects scope

The following DIF projects are taken into account:

- **Universal Resolver** [25]: 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** [26]: 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** [27]: 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** [28]: 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.

Other related projects were considered and researched:

- The **Sovrin Foundation** [20] is a public service utility that aims to promote the usage of self-sovereign identity on the Internet. It operates the Sovrin Network, which in turn is built on top of Hyperledger Indy. Many relevant W3C DID and Verifiable Credential concepts were originally conceived by the Sovrin foundation. The original code base of Hyperledger INDY was itself contributed by the Sovrin Foundation.
- The **Trust over IP (ToIP) Foundation** [29], which belongs to non-profit organization Linux Foundation, aims to determine a complete architecture for Internet-scale digital trust that incorporates cryptographic trust at the machine layer with human trust at the business, legal, and social layers. The ToIP Foundation works closely with other standards development organizations (SDOs), industry foundations, and other consortia to combine their open standards, architectures, and protocols into a complete and coherent stack for Internet-scale digital trust infrastructure. Thus, the starting definition of the ToIP stack was published as Hyperledger Aries RFC 0289.
- **The Cordentity** [30] app integrates Hyperledger Indy capabilities into the Corda platform. According to its documentation, Cordentity is a self-contained CorDapp that integrates Hyperledger Indy, for decentralised identity, with the R3 Corda Platform. This lab creates interoperability of two purpose-built ledger technologies, each with a focus on privacy. Corda is designed to enable private transact and Indy is a ledger built specifically for self-sovereign identity.
- **SOFIE** (Secure Open Federation for Internet Everywhere) [31] European Union's HORIZON 2020 project presented and evaluated how the Identification, Authentication, and Authorization (IAA) component can be used to manage IoT devices using Decentralised Identifiers (DIDs). Security in SOFIE is managed by the IAA, and Privacy and Data Sovereignty (PDS) components, which encompasses Hyperledger Indy, JSON web tokens, and OAuth2.0.
- **PyDentity** [32] is a project of OpenMined community whose goal is to make the world more privacy-preserving by lowering the barrier-to-entry to private AI technologies. PyDentity implements secure identification and authentication procedures using W3C DIDs and VCs by leveraging Hyperledger Aries and Indy.
- **TrustID** [33] solution generates a decentralised identity model using Public Key Infrastructures (PKIs) and signatures using the standard DID from the W3C where users and services are identified through a DID. The basic features are:
  - Authenticated system interactions using PKI and signatures and following the JOSE standard (Json Object Signing and Encryption).
  - Verified identities by a set of controllers.
  - The possibility for the user to guard his/her own keys is contemplated.
  - HSM integration for the key custodian.

After analysing the current approaches using DIDs and VCs, a straightforward trend has been observed in the determination of Hyperledger Indy as the most widely used distributed ledger to incorporate the registration of W3C DIDs and Verifiable Credentials.

2.2.1 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 D2.2. According to this experimental evaluation the following components were selected (see Figure 2-5):

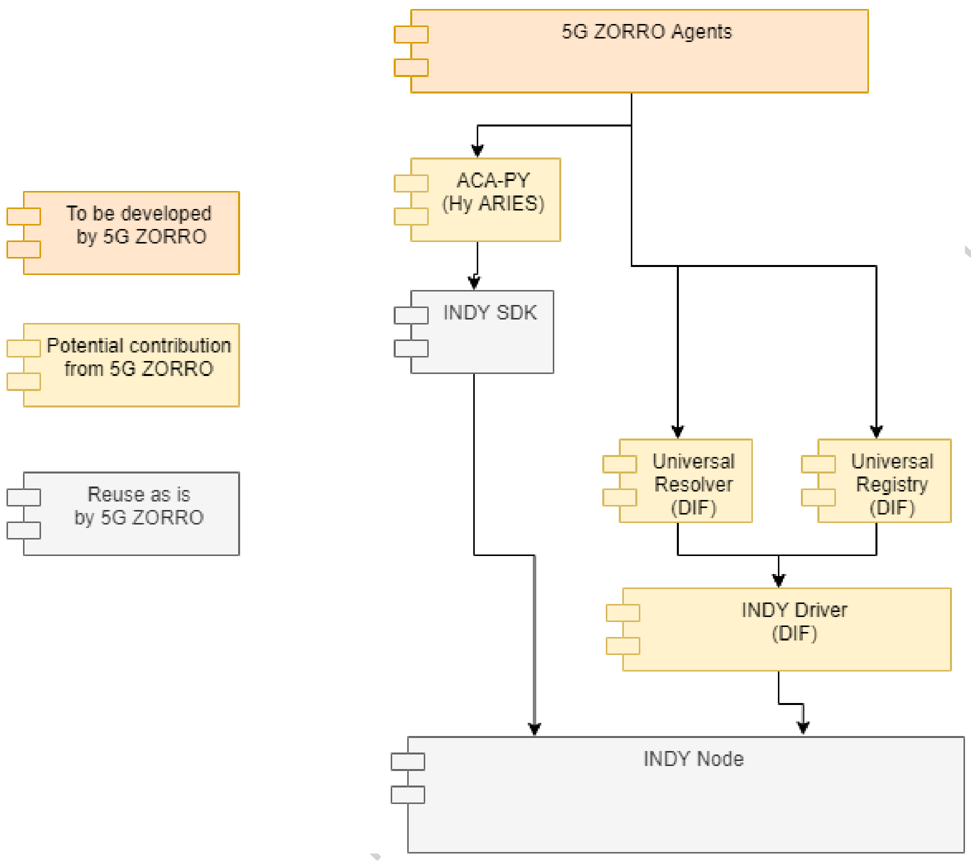


Figure 2-5: Existing DLT Modules to be Leveraged by 5GZORRO Identity and Permissions Manager

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) [34] was selected as the main baseline for 5GZORRO Identity and Permission manager (Id&P) features development. According to ACA-Py architecture (see Figure 2-6), the development of 5GZORRO Id&P business logic should be done 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 of 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.

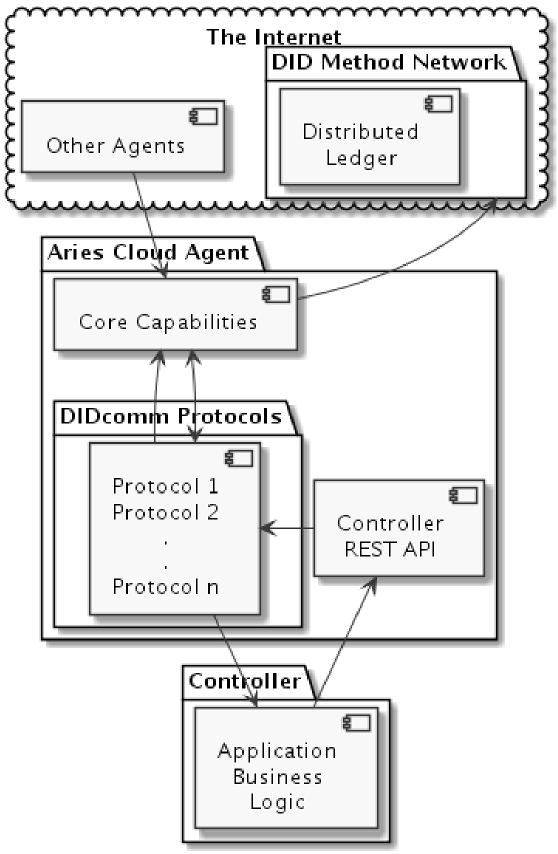


Figure 2-6: Hyperledger ARIES Cloud Agent Python Architecture

Since the Hyperledger ARIES Cloud Agent Python (ACA-Py) is very much focused on the management of Verifiable Credentials, the Decentralised Identity Foundation Universal Registrar and Universal Resolver have been selected as alternatives to manage DID Documents. The usage and, if required, the extension of the existing Sovrin driver with specific features required by 5GZORRO Id&P is anticipated.

3 Governance applications

The 5GZORRO governance platform comprises a set of software modules that is deployed by admin stakeholders and provides core marketplace capabilities to other stakeholder platforms. Services include marketplace governance management, decentralized identity & claims verification, and a managed repository of legal prose templates. Whilst deployed only by admin stakeholders, the services provided by these applications are made available to all 5GZORRO stakeholder platforms through service interfaces. These services can be discovered - and subsequently utilised - by non-admin stakeholders using service endpoints captured in a "Governance Board" DID doc; this is described in more detail in the sections that follow.

A governance portal is provided to enable governance admin stakeholders to perform associated workflow tasks such as vote on platform membership requests and review proposed legal prose template updates.

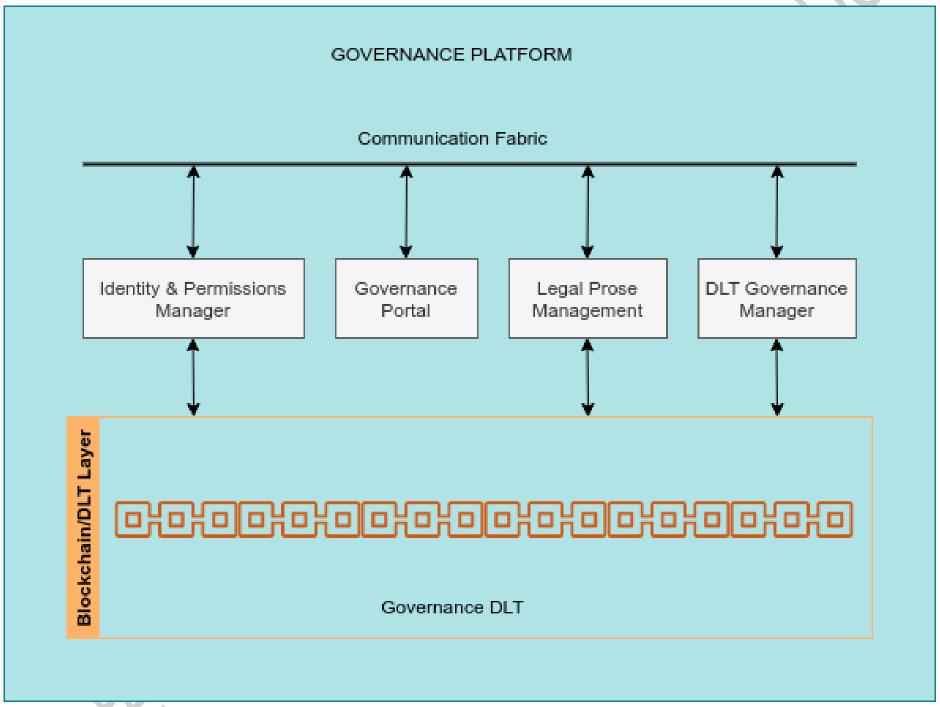


Figure 3-1: Governance Platform Architecture

3.1 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 strictly in scope for 5GZORRO, 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 5GZORRO Specific Enhancements

By leveraging DLT to underpin the marketplace governance we see the decentralized management of the platform with greater transparency and auditability. Specifically, the following processes and entities will be subject to 5GZORRO governance.

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

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 3.2.4.1. 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.

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.

3.1.2 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:

- **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 will be leveraged such that admin stakeholders can keep a synchronized view of membership and proposal status' and support query capabilities

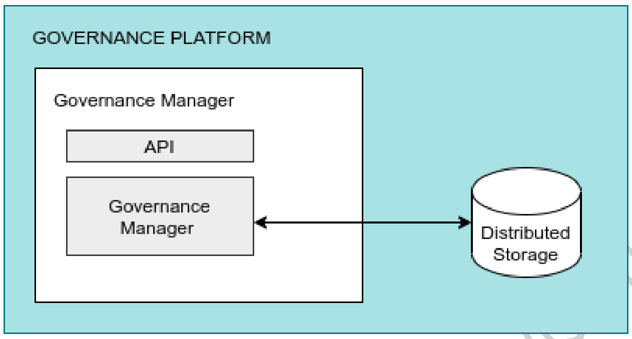


Figure 3-2: Governance Manager Module Architecture

3.1.3 Specific and relevant workflows

3.1.3.1 *General governance workflow*

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

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.

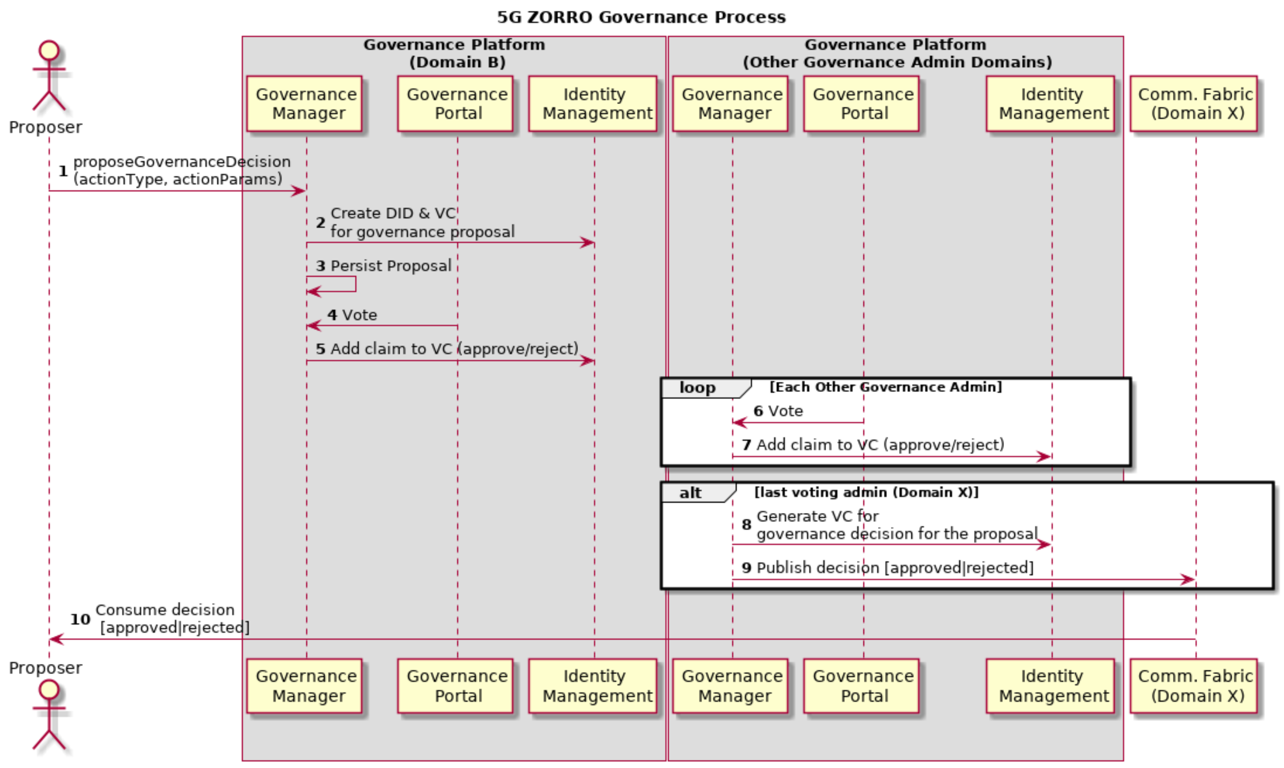

5G ZORRO Governance Process
Proposer
Governance Platform (Domain B)
Governance Manager
Governance Portal
Identity Management
Governance Platform (Other Governance Admin Domains)
Governance Manager
Governance Portal
Identity Management
Comm. Fabric (Domain X)
1 proposeGovernanceDecision (actionType, actionParams)
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 3-3: Governance Workflow

3.1.3.2 *Stakeholder membership request*

The below workflow outlines the steps relating to a membership request, whereby a request results in a membership record (and associated status) being persisted in distributed storage (accessible by each governance board member) for the purposes of supporting query functionality.

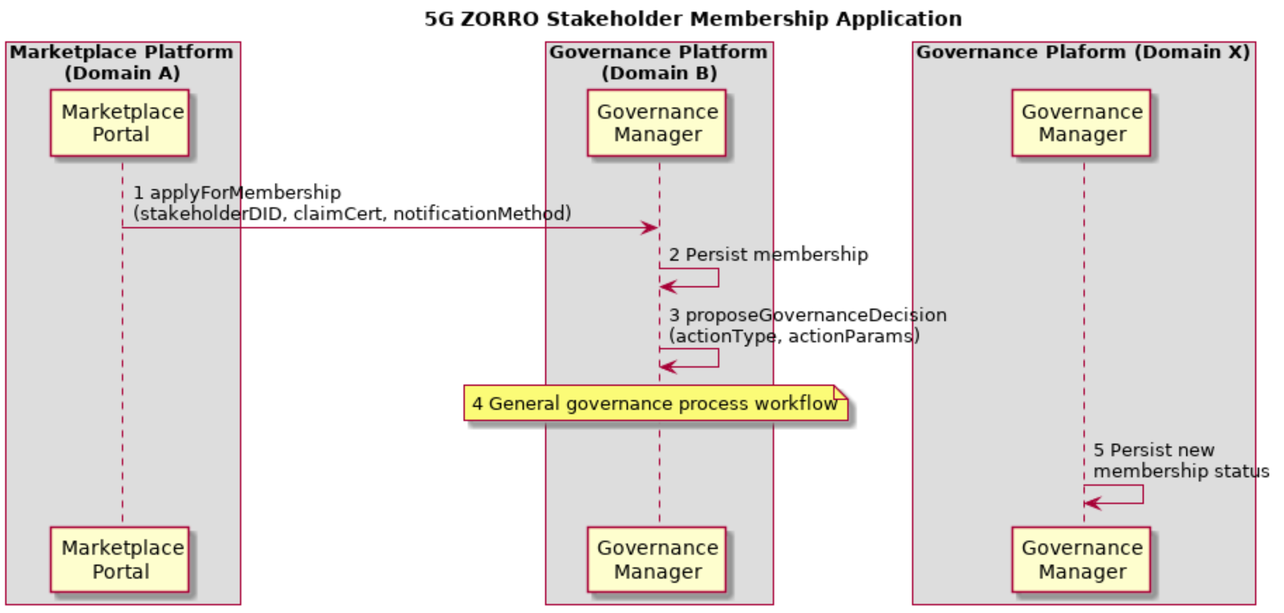

5G ZORRO Stakeholder Membership Application
Marketplace Platform (Domain A)
Marketplace Portal
Governance Platform (Domain B)
Governance Manager
Governance Plaform (Domain X)
Governance Manager
1 applyForMembership (stakeholderDID, claimCert, notificationMethod)
2 Persist membership
3 proposeGovernanceDecision (actionType, actionParams)
4 General governance process workflow
5 Persist new membership status

Figure 3-4: Stakeholder Membership Application

Pre-conditions: The new 5GZORRO platform has generated a DID and appropriate VCs that encapsulate their identity and intended roles/permissions within the 5GZORRO ecosystem.

Step 1: New 5GZORRO stakeholder makes a request to a Governance Manager endpoint deployed in the domain of a Governance Board member. Discovery, as previously described, is achieved through the DID doc of the Governance Board DID.

Step 2: A membership record is persisted in distributed storage such that it can be queried efficiently and assigned a PENDING status.

Step 3: A governance decision proposal is made on behalf of the requesting entity and the steps identified above in the general workflow proceed until a decision is reached over the request.

Step 4: The completion of the governance decision process culminates in the final admin issuing a Verifiable Credential as to whether the decision was to approve or reject. This stakeholder issues the VC and then proceeds to updating the status of the membership record.

Step 5: The deciding admin updates the status of the membership record with either ACTIVE or REJECTED.

3.1.4 APIs

Operation name	**applyForMembership**	
Description	**API endpoint to submit Marketplace membership requests to**	
Input Parameters		
Name	**Type**	**Description**
stakeholderDID	String	DID that uniquely identifies the legal entity to be onboarded
stakeholderClaimCertificate	String	Certificate containing the claims that have been attributed to the new stakeholder. For example, which components of the 5GZORRO system they have deployed
NotificationMethod	NotificationMethod	Notification details that should be used to notify the stakeholder of any governance decisions and marketplace events as appropriate
Output Parameters		
None	*None*	*None*
Notes		

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 identifies 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
Notes		

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
Notes		

Operation name	**revokeMembership**	
Description	API endpoint to revoke Marketplace access for a stakeholder	
Input Parameters		
Name	**Type**	**Description**
stakeholderDiD	String	The DID of the stakeholder whose access is to be revoked
Output Parameters		
None		
Notes		
Any requests made to this endpoint that are not for the requesting party, will be subject to a governance process. Requests to revoke your own access would be accepted automatically.		

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 identifies the proposal
Notes:		

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
Notes:		

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
Notes:		

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		
Notes:		

3.1.5 Information Models

3.1.5.1 *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

3.1.5.2 *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

3.1.5.3 *Member*

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

3.1.5.4 *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

3.1.5.5 *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

3.2 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.

In its present form, Identity Management is able to identify providers, consumers, services, resources, organizations, etc., using Decentralised Identifiers (DIDs) associated with DID Documents. What is more, DIDs can also be used for authentication through a Verifiable Credential linked to a DID Document. 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.

3.2.1 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 [24]. 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):

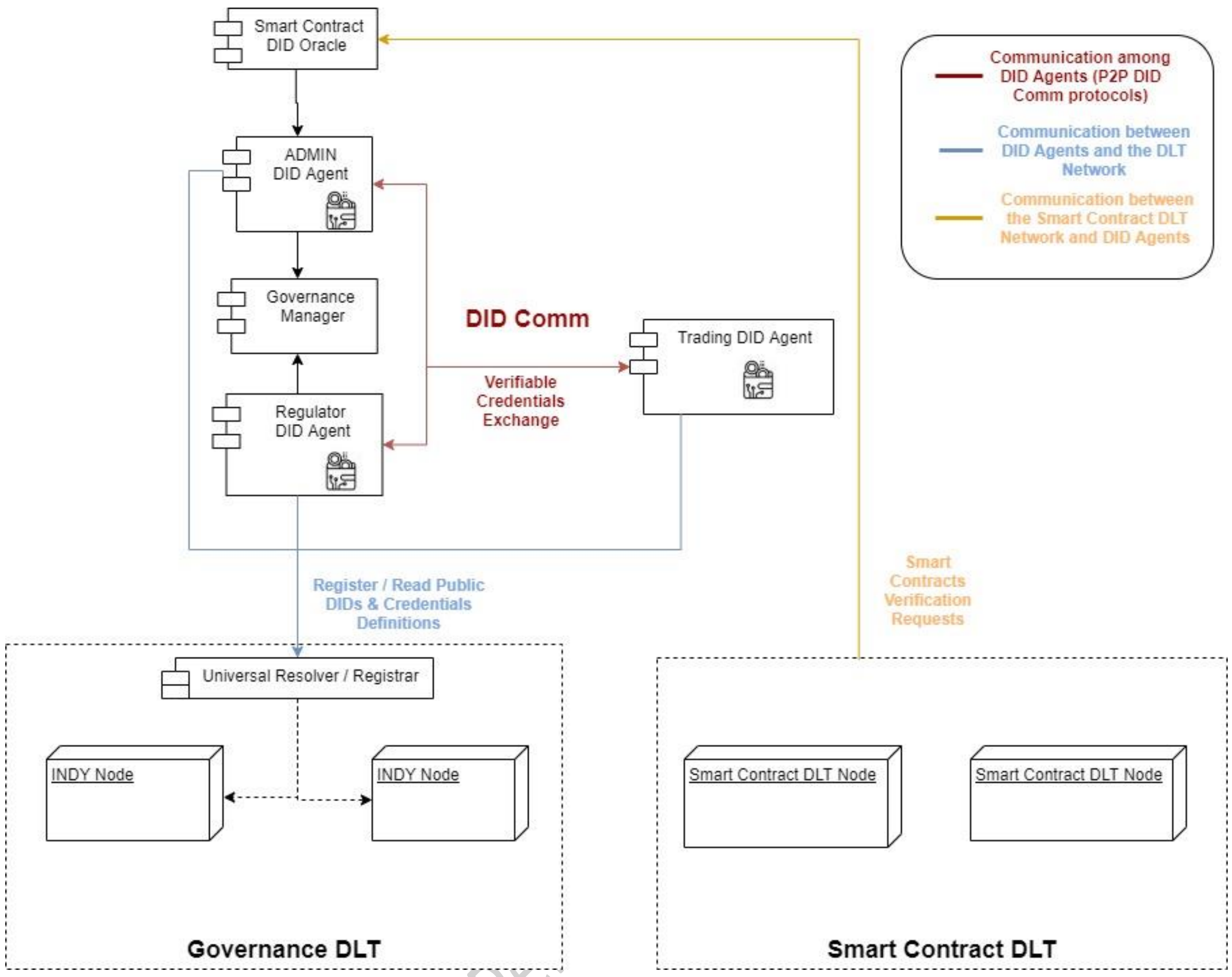
- **Admin (or Governance) DID Agent**: it provides specific business logic to be used to issue non-regulated Offers VCs on Providers request and to manage Governance Verifiable Credentials including credentials about 5GZORRO stakeholders Marketplace membership. Admin DID Agents have permissions to write 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 manage Verifiable Credentials required to support trustworthy trading of 5G Offers in 5GZORRO Marketplace. It is not expected the Trading DID Agent will require to issue a Credential and it won't have to write in the Governance DLT. 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 Holder
    2. **Consumer Trading Agent:** it plays the role of the Verifiable Credential Offer Verifier

DID Agents communicate among each other on top of P2P DID Comm protocols[28] 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.

The Admin DID Agent should also be used by the Smart Contracts DLT to handle credential verification requests as demanded by the Smart Contract execution e.g., Offer validation as well as SLA violation validation. This interface should be supported via a DID Oracle.



Smart Contract DID Oracle
ADMIN DID Agent
Governance Manager
Regulator DID Agent
DID Comm
Trading DID Agent
Verifiable Credentials Exchange
Communication among DID Agents (P2P DID Comm protocols)
Communication between DID Agents and the DLT Network
Communication between the Smart Contract DLT Network and DID Agents
Register / Read Public DIDs & Credentials Definitions
Smart Contracts Verification Requests
Universal Resolver / Registrar
INDY Node
INDY Node
Smart Contract DLT Node
Smart Contract DLT Node
Governance DLT
Smart Contract DLT

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

3.2.2 Deployment scenarios

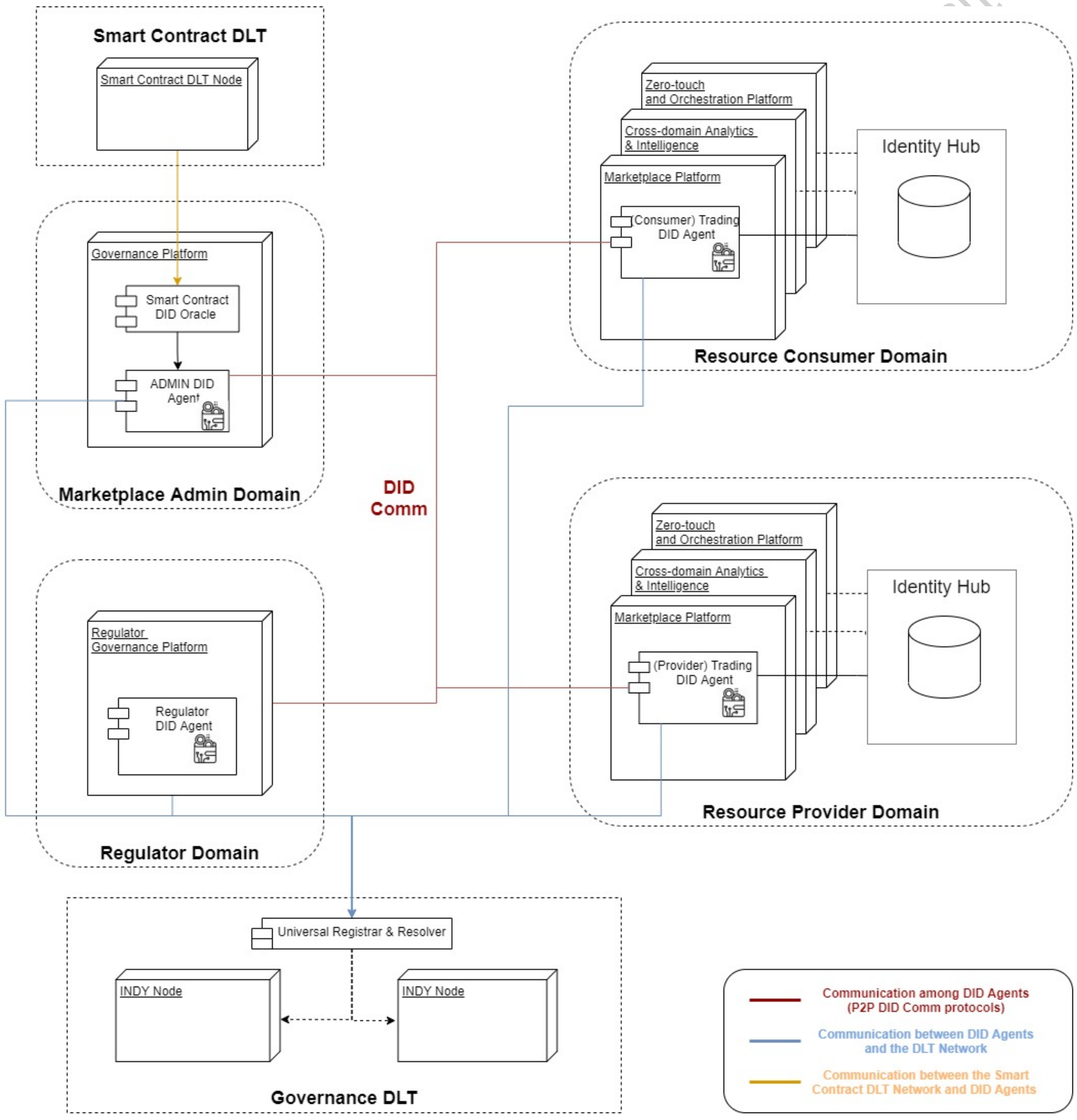
Identity and Permissions Manager DID Agents will be deployed in different 5GZORRO platforms according to their types and the scenarios to be supported. Figure 3-6depicts a possible deployment topology to support the offering and trading of a Regulated Resource in 5GZORRO Marketplace notably a Spectrum 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 deployed in the Marketplace platform 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 [27] 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 3.2.5).

- The (Consumer) Trading DID Agent is deployed in the Marketplace platform operated by the Spectrum provider 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.



Smart Contract DLT
Smart Contract DLT Node
Governance Platform
Smart Contract DID Oracle
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
(Consumer) Trading DID Agent
Identity Hub
Resource Consumer Domain
(Provider) Trading DID Agent
Resource Provider Domain
Universal Registrar & Resolver
INDY Node
INDY Node
Governance DLT
Communication among DID Agents (P2P DID Comm protocols)
Communication between DID Agents and the DLT Network
Communication between the Smart Contract DLT Network and DID Agents

Figure 3-6: Identity and Permission Agents Deployment Scenario

3.2.3 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 is envisaged to use the DIF universal DID Resolver to read public DIDs. This integration can be done at ACA-Py level or at 5GZORRO Core Agent Controller level (it is still to be decided). 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 [35] concept as introduced by reTHINK project [36].

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) [37] 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) [38] allows an agent to issue a credential to another agent.
- The **present proof protocol** (ARIES RFC 0454) [39] enables an agent to request and receive a proof from another agent.

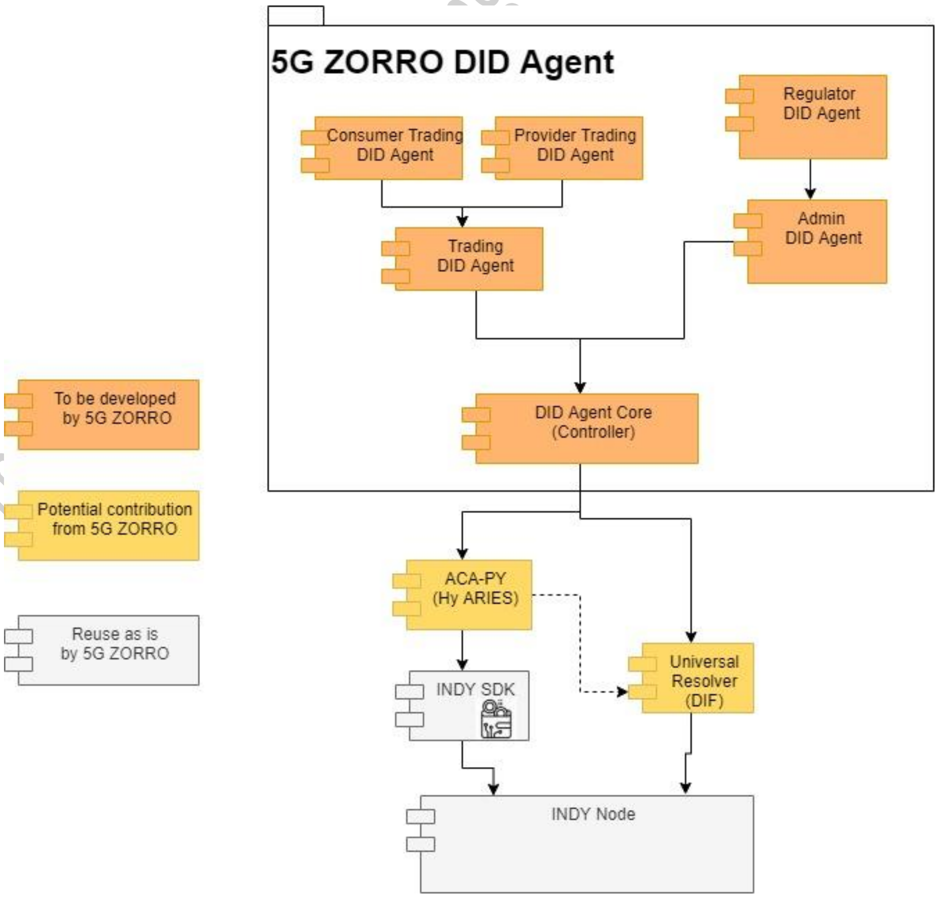
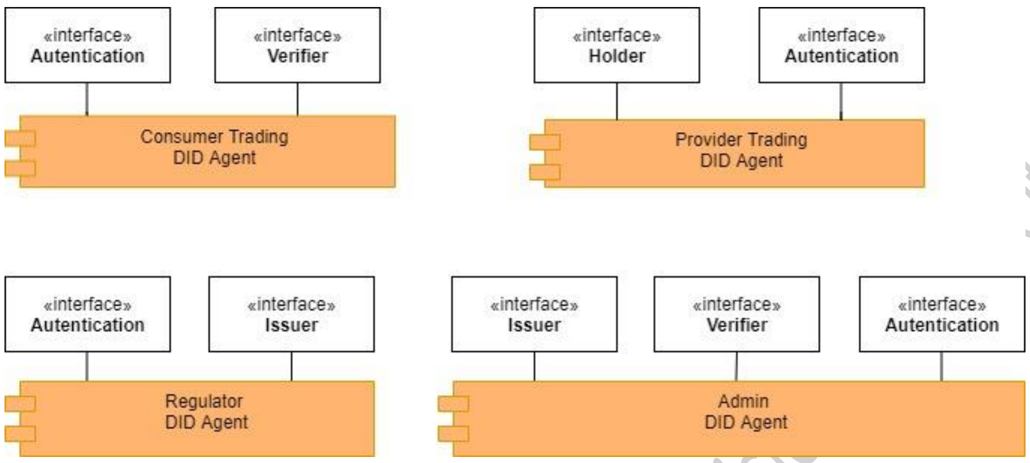


Figure 3-7: DID Agent Design

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 below and they are described in Section 3.2.5.



«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
- Handler to process and forward events coming from ACA-Py webhook
- 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 OfferVC Holder features

The Consumer Trading DID Agent provides the following major features:

- Business Logic to support the OfferVC 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
- Business Logic to support VC Verifier features as requested by the Smart Contracts execution (Offers validation and SLA Violation validation).

3.2.4 Specific and Relevant Workflows

3.2.4.1 *Agent Bootstrap*

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

- seed
- the DID of the Marketplace Governance Board
- a list of all platform service endpoints that can be used to certify the Stakeholder has all required 5GZORRO platforms successfully deployed (to be defined which service endpoints are mandatory per stakeholder type)

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. As soon as the Trading Agent receives the Stakeholder Credential an Authentication Presentation file is generated to be used in the authenticate() method.

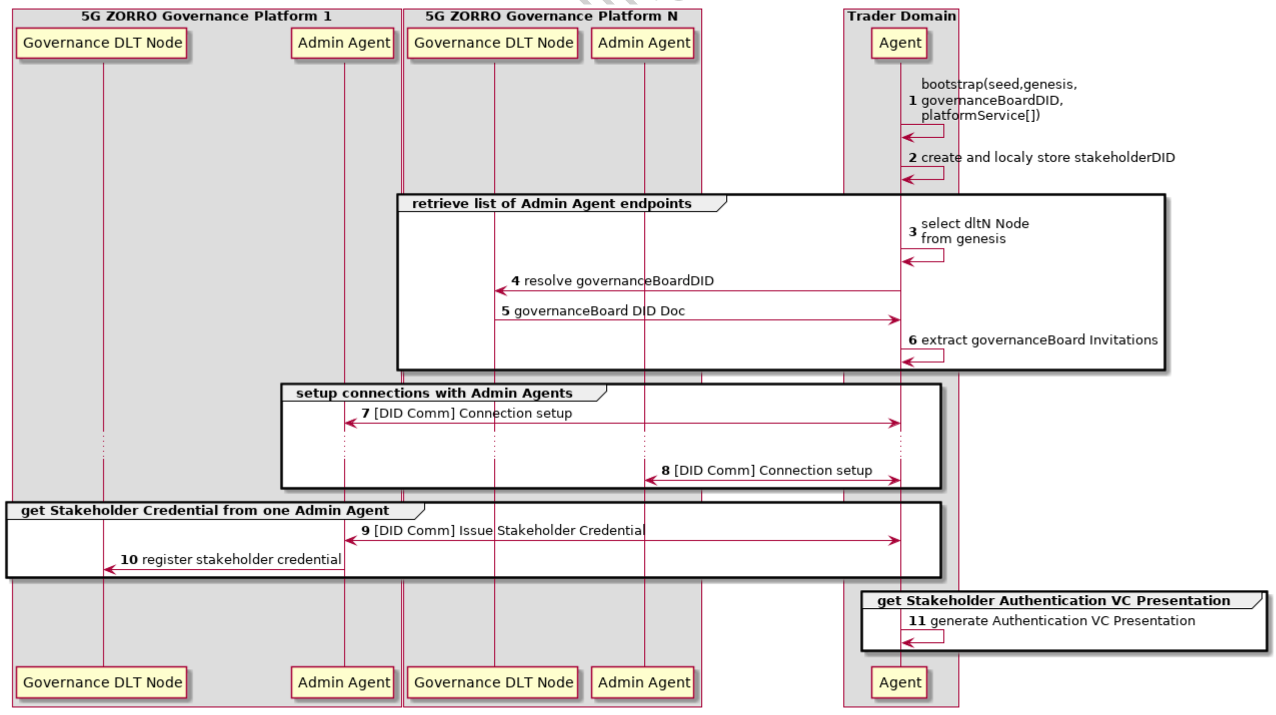

5G ZORRO Governance Platform 1
5G ZORRO Governance Platform N
Trader Domain
Governance DLT Node
Admin Agent
Governance DLT Node
Admin Agent
Agent
1 bootstrap(seed,genesis, governanceBoardDID, platformService[])
2 create and localy store stakeholderDID
retrieve list of Admin Agent endpoints
3 select dltN 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

3.2.4.2 *DID Creation (e.g., Offer DID)*

The diagram below (see Figure 3-10) describes the main steps to create a DID and associated Credentials by using the Handler Agent API from the Identity and Permissions Manager component. The Handler Agent API is implemented by the Trading Consumer Agent and by the Admin Agent and can be consumed by any 5GZORRO component to manage DIDs and associated Credentials from W3C Verifiable Credential Holder role perspective. The creation of DID Offers by the Marketplace Catalogue is a major example of DID creation. The creation of DIDs and associated Credential is performed by calling the 'createDID' function (step 1). A handler endpoint is provided as an input parameter to receive DID events. As soon as the DID is created and locally stored in the Agent Wallet with associated Service endpoints, the execution is returned with the new DID (step 3). Then, the process to issue a 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 (steps 4 and 5). 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 3.2.4.4). 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 (step 6). Optionally, the Holder Client can retrieve the Invitation object that is required by any potential 5GZORRO component interested to request the proof of claims associated to the new DID, for example, the Marketplace Catalogue component operated by any interested 5GZORRO Product Consumer (see how claims are verified by using the Verifier Agent API in Section 3.2.4.3).

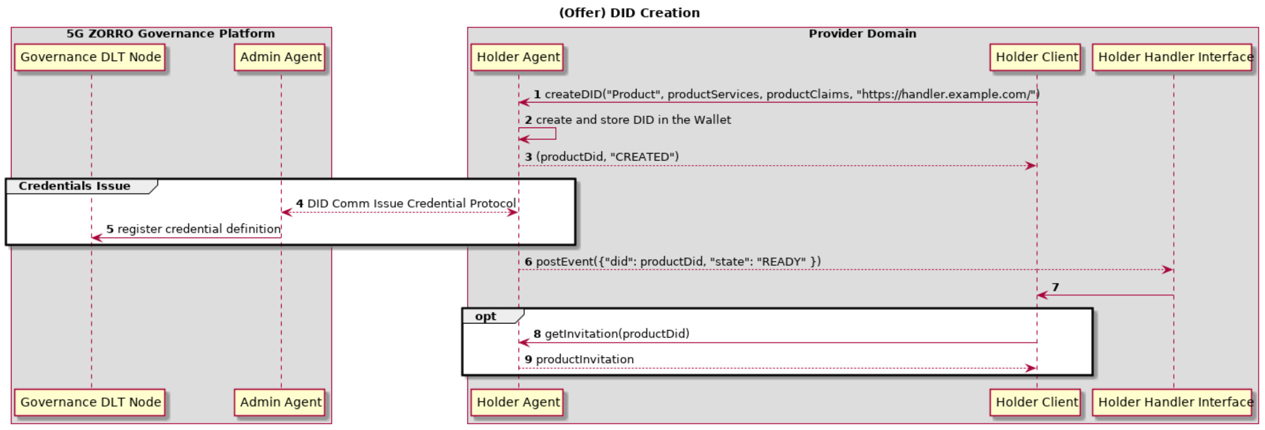

(Offer) DID Creation
5G ZORRO Governance Platform
Governance DLT Node
Admin Agent
Provider Domain
Holder Agent
Holder Client
Holder Handler Interface
1 createDID("Product", productServices, productClaims, "https://handler.example.com/")
2 create and store DID in the Wallet
3 (productDid, "CREATED")
Credentials Issue
4 DID Comm Issue Credential Protocol
5 register credential definition
6 postEvent({"did": productDid, "state": "READY" })
7
opt
8 getInvitation(productDid)
9 productInvitation
Governance DLT Node
Admin Agent
Holder Agent
Holder Client
Holder Handler Interface

Figure 3-10: DID Creation Workflow

3.2.4.3 *Consumer Verifies Claims from Provider Offer*

Figure 3-11 describes the main steps to verify DID Credentials by using the Verifier Agent API from the Identity and Permissions Manager component. The Verifier Agent API is implemented by the Trading Provider Agent and by the Admin Agent and can be consumed by any 5GZORRO component to manage DIDs Credentials from W3C Verifiable Credential Verifier role perspective. The verification of Product offer claims by a 5GZORRO Consumer is a major example of DID Credential verification. This verification is performed by the Trading Consumer Agent (e.g., the Consumer Marketplace Portal) with a single call to the 'proofRequest' function (step 1). The invitation object that is required to establish a secure connection with the Holder Agent is provided as an input parameter, as well as the handler endpoint to receive Proof events. The proof_id is returned (step 2) and the process to retrieve the Credential Presentation from the Trading Provider Agent (Holder) is executed in step 3 by using the DID Comm Present Proof protocol. To be noted the invitation object provided as input parameter in step 1 is required to establish a secure connection with the Provider Agent (in case no previous connection is already established). As soon as the Credential Presentation is received by the Trading Consumer Agent, the proof is processed by using the cryptographic verification methods expressed in the Credential presentation (step 4) as well as checking the credential definition previously registered by the Admin Agent Issuer (step 5). When the verification is finished, a proof Event is dispatched towards the Trading Consumer Handler to notify about the proof verification result: (proof is) VERIFIED (step 6) or (proof has) FAILED (step 7).

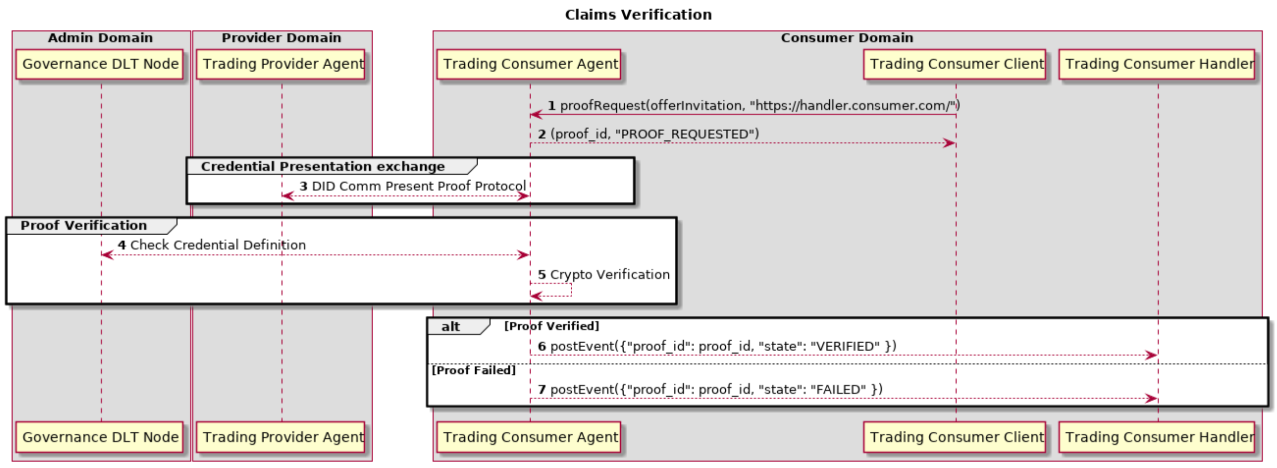


Figure 3-11: Credential Verification Workflow

3.2.4.4 *Admin Agent Handles Credential Issue Requests*

The diagram below (see Figure 3-12) describes the main steps to issue DID 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 should be mostly consumed by the 5GZORRO Governance Manager component to decide and control the issue of DIDs Credentials requested by other Holder Agents (e.g., Trading Provider Agents) from W3C Verifiable Credential Issuer role perspective. The issue of Product Credential claims as requested by a 5GZORRO Provider is a major example of DID Credential issue. Another relevant example is the issue of Spectrum Credentials by the Regulator Agent.

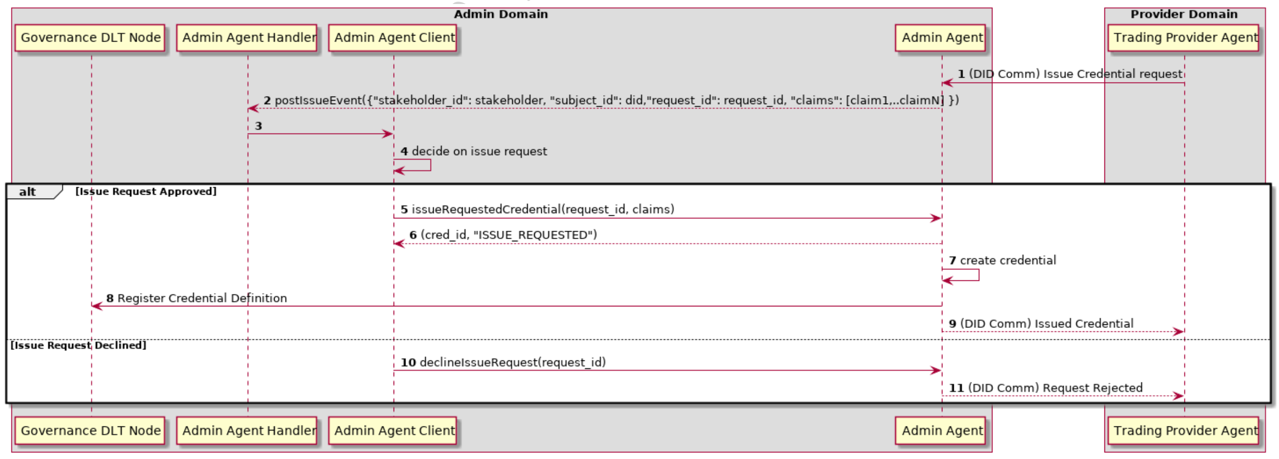

Admin Domain
Provider Domain
Governance DLT Node
Admin Agent Handler
Admin Agent Client
Admin Agent
Trading Provider Agent
1 (DID Comm) Issue Credential request
2 postIssueEvent({"stakeholder_id": stakeholder, "subject_id": did,"request_id": request_id, "claims": [claim1,..claimN] })
3
4 decide on issue 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
Governance DLT Node
Admin Agent Handler
Admin Agent Client
Admin Agent
Trading Provider Agent

Figure 3-12: 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 3.2.4.2) (step 1) and an Issue Event is dispatched towards the Admin Agent Handler (step 2). The Admin Agent Handle should also be implemented by the 5GZORRO Governance Manager component and decides on the credential issue request according to the adopted governance model.

In case the request is approved, 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).

3.2.5 APIs

3.2.5.1 *Authentication*

Operation name	**authenticate**	
Description	To authenticate a stakeholder by using the Stakeholder Credential Authentication Presentation	
Input Parameters		
Name	**Type**	**Description**
id_token	str	The *id_token* that is included in the Stakeholder Credential Authentication Presentation that is generated when the Agent is executed for the 1st time or returned by the refresh function.
Output Parameters		
response	boolean	Authentication result as a boolean, *true* if the authentication was successful, *false* if failed
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	**refresh**	
Description	To refresh an id token	
Input Parameters		
Name	**Type**	**Description**
id_token	str	The id_token to be refreshed
refresh_token	object	The *refresh_token* that is included in the Stakeholder Credential Authentication Presentation
Output Parameters		
id_token	str	The refreshed new id_token
reason_code	object	If refresh fails the reason should be provided here
Notes		

3.2.5.2 *Holder Agent*

Operation name	**createDID**	
Description	Request to create a private 5GZORRO DID and all associated Claims	
Input Parameters		
Name	**Type**	**Description**
type	str	5GZORRO Subject type including "Resource", "Service", "Product", "Agreement"
services	object	Services to be associated with the new DID
claims	object	List of claims to be associated with the new DID

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
Notes		
The request to create a new DID is asynchronous and the Handler endpoint has to 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	**readDIDStatus**	
Description	Request to read DID status	
Input Parameters		
Name	**Type**	**Description**
did	str	The target DID
Output Parameters		
state	str	The DID Status
Notes		

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		

Operation name	**removeDID**	
Description	This method is utilised to remove a DID	
Input Parameters		
Name	**Type**	**Description**
did	str	String that identifies the subject to be removed
Output Parameters		
State	str	Information on the completion status of the current action.
Metadata	object	Descriptive information about the action.
Notes		
The remove of the DID will perform the revocation of all associated credentials.		

Operation name	**updateDID**	
Description	Request to update a private 5GZORRO DID and all associated Claims	
Input Parameters		
Name	**Type**	**Description**
did	str	String that identifies the subject to be updated
services	object	Services to be updated with the new DID

claims	object	List of claims to be updated
Output Parameters		
state	str	State of the DID update request
Notes		

Operation name	**getInvitation**	
Description	Retrieves Invitation Object to be used to connect with Agent for a specific DID	
Input Parameters		
Name	**Type**	**Description**
did	str	String that identifies the entity
Output Parameters		
invitation	object	Invitation Object to be used to connect with Agent for the provided DID
Notes		
For Offers DID, the catalogue should provide the invitation object to let interested Trading Consumers Agents to connect with the Trading Provider Agent to retrieve the Credential presentation for verification purposes.		

Operation name	**revokeCredential**	
Description	This method is utilised to remove a credential	
Input Parameters		
Name	**Type**	**Description**
did	str	String that identifies the subject associated to the credential to be revoked
credential_id	str	String that identifies the credential to be revoked
Output Parameters		
state	str	Information on the completion status of the current action.
metadata	object	Descriptive information about the action.
Notes		
This is an asynchronous function where the result will be returned to the DID Handler in the credential topic.		

3.2.5.3 *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 3-13):

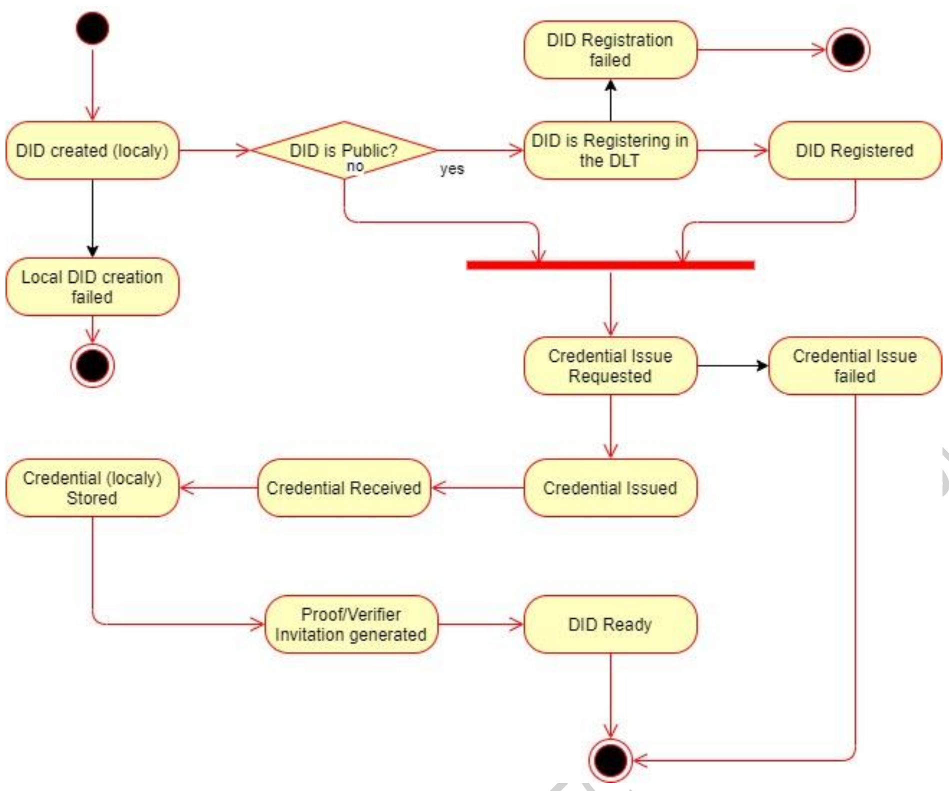


Figure 3-13: DID State Machine

3.2.5.4 *Issuer Admin Agent*

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	**revokeCredential**
Description	Revokes a credential issued by the agent

Input Parameters		
Name	**Type**	**Description**
cred_id	str	Credential Identifier
Output Parameters		
N/A	--	--
Notes		

Operation name	**addHandler**	
Description	Adds an Issuer Handler	
Input Parameters		
Name	**Type**	**Description**
handler	str	Webhook url
Output Parameters		
N/A	--	--
Notes		

Operation name	**readCredentials**	
Description	Lists Credentials stored on the agent	
Input Parameters		
Name	**Type**	**Description**
N/A	--	--
Output Parameters		
results	object	List of all credentials
Notes		

Operation name	**readCredential**	
Description	List Credential issued by the agent	
Input Parameters		
Name	**Type**	**Description**
cred_id	str	Credential Identifier
Output Parameters		
credential	object	Requested Credential
created_at	str	Date of Credential creation
state	str	State of the Credential sent
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		
request_id	str	Issue Request Identifier
declined_at	str	Date of Credential rejection
state	str	State of the Credential rejection
Notes		
The specified output fields are among the most important ones.		

Operation name	**readDeclinedIssueRequest**	
Description	Lists specific declined Credential provided by the agent	
Input Parameters		
Name	**Type**	**Description**
request_id	str	Credential Request Identifier
Output Parameters		
claim	str	Credential Exchange Identifier
declined_at	str	Date of Credential rejection
state	str	State of the Credential rejection
Notes		
The specified output fields are among the most important ones.		

3.2.5.5 *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.

Issue Event (`/issue`)

Parameter	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

Revoke Event (/revoke)

Parameter	Type	Description
credential_id	DID	Identifier of the credential to be revoked

3.2.5.6 *Verifier Agent*

Operation name	**proofRequest**	
Description	Requests the verification of DID claims	
Input Parameters		
Name	**Type**	**Description**
invitation	object	Invitation
claims	object	List of claims Id to be verified
handler	str	Handler (callback) endpoint to process Proof status events dispatched by the Agent
Output Parameters		
proof_id	str	Identifier of the Proof request
state	str	State of the Proof Request
Notes		
The request to verify DID claims is asynchronous and the Handler endpoint has to be provided to receive events about the proof process. A secured connection with Holder Agent is established by using the Invitation Object which will be used to request the credentials presentation. When the credentials presentation containing the claims proof are received, the Verifier Agent will verify the claims proof registered in the Governance DLT.		

Operation name	**readProofStatus**	
Description	Request to read DID claims Proof status	
Input Parameters		
Name	**Type**	**Description**
proof_id	str	Identifies the Proof
Output Parameters		
state	str	The Proof Status
Notes		

Operation name	**readProofPresentation**	
Description	Retrieves a Proof Presentation previously provided by a Holder Agent	
Input Parameters		
Name	**Type**	**Description**
proof_id	str	Proof Identifier
Output Parameters		
state	str	State of the Proof Request
presentation	object	Object that contains the proof information
Notes		
The specified output fields are among the most important ones.		

3.2.5.7 *Verifier Agent Handler API*

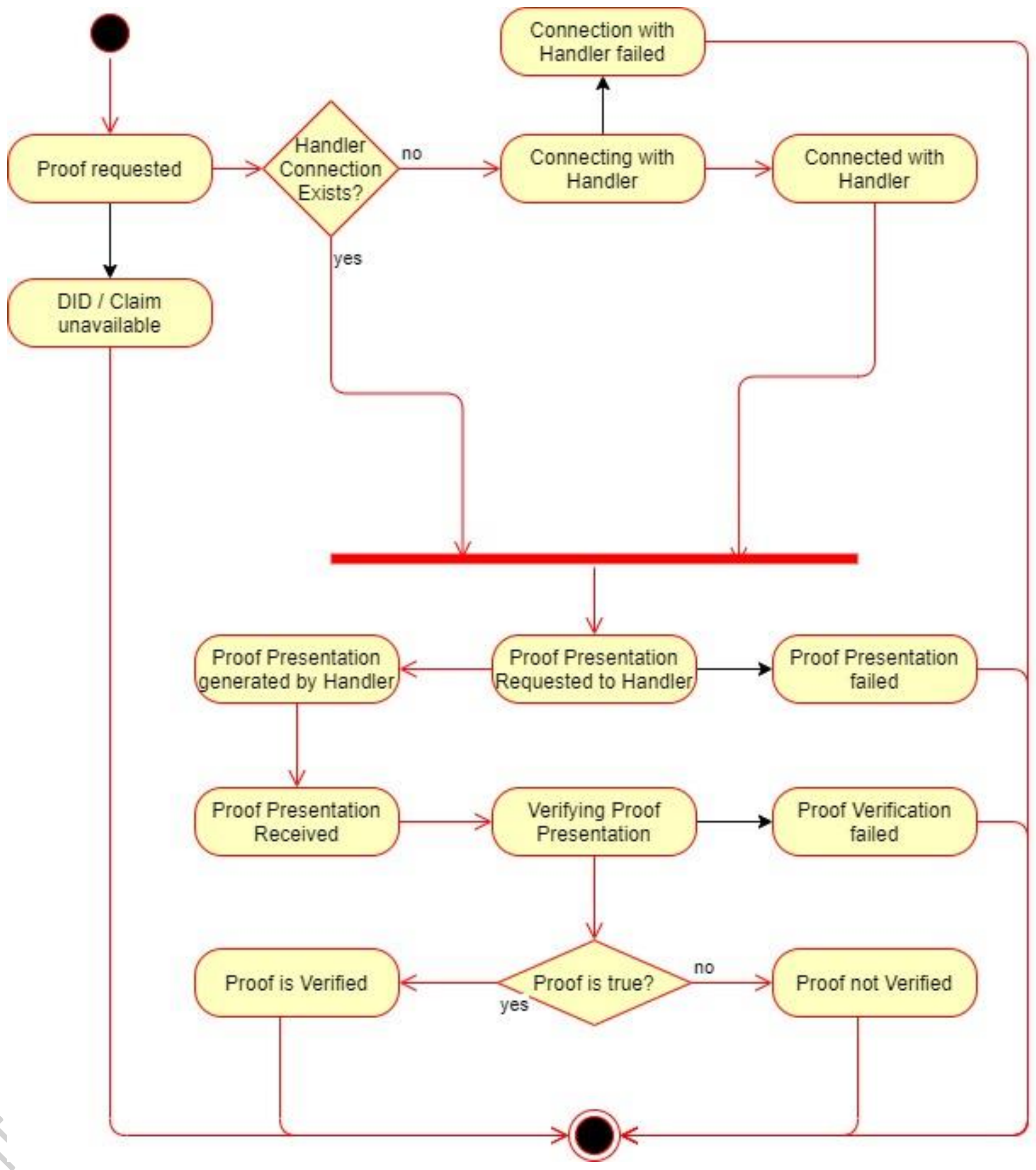
The Verifier Agent Handler API is implemented by components that request proofs about DID Credential Claims and which are clients of the Verifier Agent API.

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 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 3-14: Proof State Machine

3.3 Legal Prose Management

Legal Prose Management 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 subject to a consortium governance process prior to being approved for use in the 5GZORRO marketplace.

Templates comprise legal prose and a machine-readable model (Ricardian contract) that can be queried by stakeholder marketplace platforms to build concrete SLAs, Agreements 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.

3.3.1 5GZORRO Specific Enhancements

All agreements within the 5GZORRO ecosystem will be based on templates that have been subject to an approval process by the 5GZORRO governance board. It is believed that this will give rise to better transparency and consistency across the marketplace and efficiencies that result in the homogenous approach to contract definition.

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 whereby each one reviews and approves/rejects a proposed template. The approval of a template will be the result of this voting process according to the agreed governance criteria e.g., all or majority approve or as emanating from the applicable legal framework for example in the case of spectrum regulation and obligations.

Legal prose templates will be developed using the Accord Project [43], a framework for building smart legal agreements. It marries legally enforceable prose with machine-readable data and business logic in a technology-neutral manner. The architecture of Accord will be discussed in the next section, but the benefits that it brings is the ability to define Ricardian contracts and automate their lifecycle and execution thanks to embedded business logic. For example, an SLA agreement can have business logic embedded in it capable of determining if it has been breached based on certain inputs (state and parameters).

A template will be assigned a DID and the issuance of verifiable claims during the approval process will give rise to global cross-domain identifiability, authenticity and verifiable status of templates.

3.3.2 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*)

In order to meet the legal templating and automation requirements of the 5GZORRO platform it has been decided to utilise the Accord Project; a set of projects and libraries belonging to the Linux Foundation for building smart agreements and documents on a technology neutral platform. The following diagram illustrates the key entities that comprise the module, which will also have dependencies on the DLT Governance Manager module for providing governance services over the template repository. Definition and editing of such templates is expected to be undertaken through the Governance Portal.

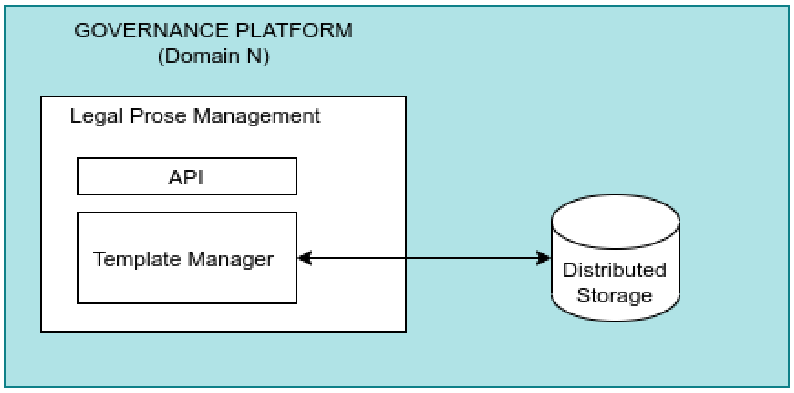


Figure 3-15: Legal Prose Management Module Architecture

- **API** – a set of endpoints to expose the capabilities of the module, such as template proposal, querying and archiving. These endpoints will be available cross-domain to non-admin stakeholders to service their legal prose needs since the broader Governance platform is only deployed by governance admin stakeholders.
- **Template Manager** – Functional entity that implements the logic of the module to ensure templates and their status are managed in accordance with the agreed governance model and updates are synchronised across all stakeholders belonging to the Governance Board.

3.3.2.1 *Accord Project for Smart Legal Contracts*

Accord Project Templates are composed of three core elements: legal prose (natural language text), the data model and executable business logic. The combination of these three elements culminates in a human-readable and machine-readable smart agreement.

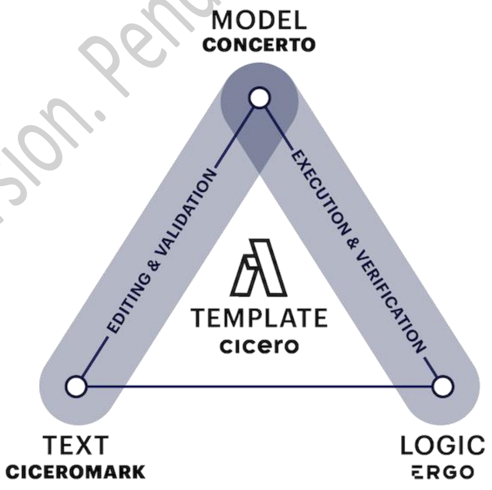
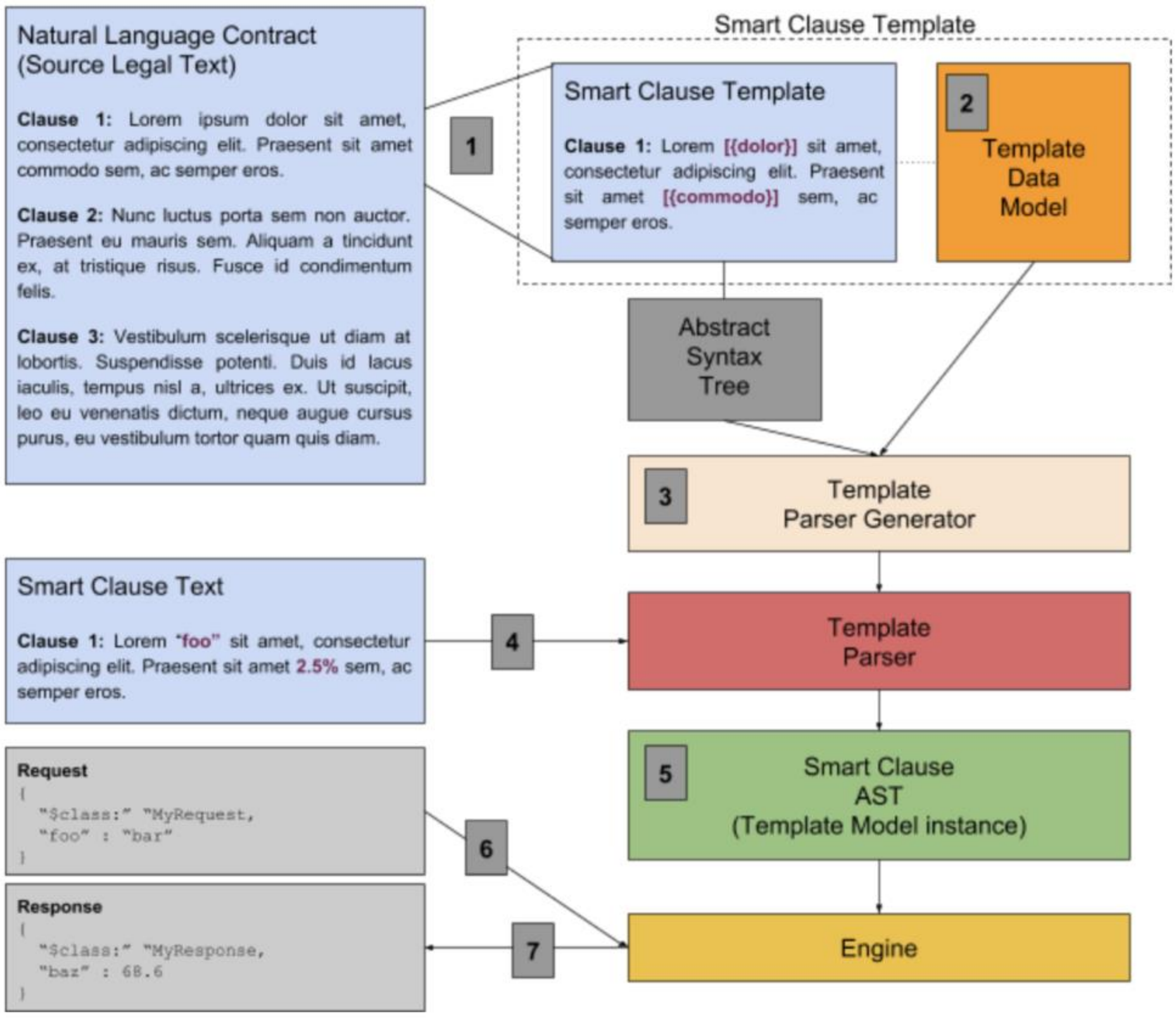


Figure 3-16: Accord Project Template Composition

Accord contracts can be thought of in two contexts, a) definition and b) execution. The definition takes the form of defining natural text and subsequently generating a template to mirror the semantics of the text. It can then be used to define a concrete instance of an agreement either passing a populated model and template, or a legal document that precisely matches the format of the template to the template parser.



Clause 1: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent sit amet commodo sem, ac semper eros.

Clause 2: Nunc luctus porta sem non auctor. Praesent eu mauris sem. Aliquam a tincidunt ex, at tristique risus. Fusce id condimentum felis.

Clause 3: Vestibulum scelerisque ut diam at lobortis. Suspendisse potenti. Duis id lacus iaculis, tempus nisl a, ultrices ex. Ut suscipit, leo eu venenatis dictum, neque augue cursus purus, eu vestibulum tortor quam quis diam.

Clause 1: Lorem [{dolor}] sit amet, consectetur adipiscing elit. Praesent sit amet [{commodo}] sem, ac semper eros.

Clause 1: Lorem "foo" sit amet, consectetur adipiscing elit. Praesent sit amet 2.5% sem, ac semper eros.

```
{
  "$class:" "MyRequest,
  "foo" : "bar"
}
```

```
{
  "$class:" "MyResponse,
  "baz" : 68.6
}
```

Figure 3-17: Accord Template & Contract definition

Step 1-2: Legal prose is broken into a template and template data model.

Step 3-4: A template parser can be generated to take a legal text (or a hydrated JSON model and a template) to output a template model instance capturing the executable context of the text.

Step 6-7: Once you have a concrete instance of the template then you can invoke with requests that represent events of significance to the clause from the outside world (e.g., post a measurement for an SLA, for the clause to subsequently calculate if a violation has occurred). Execution can result in a response corresponding with – for example – updated state, and a contract obligation (event) (e.g., SLA Violation).

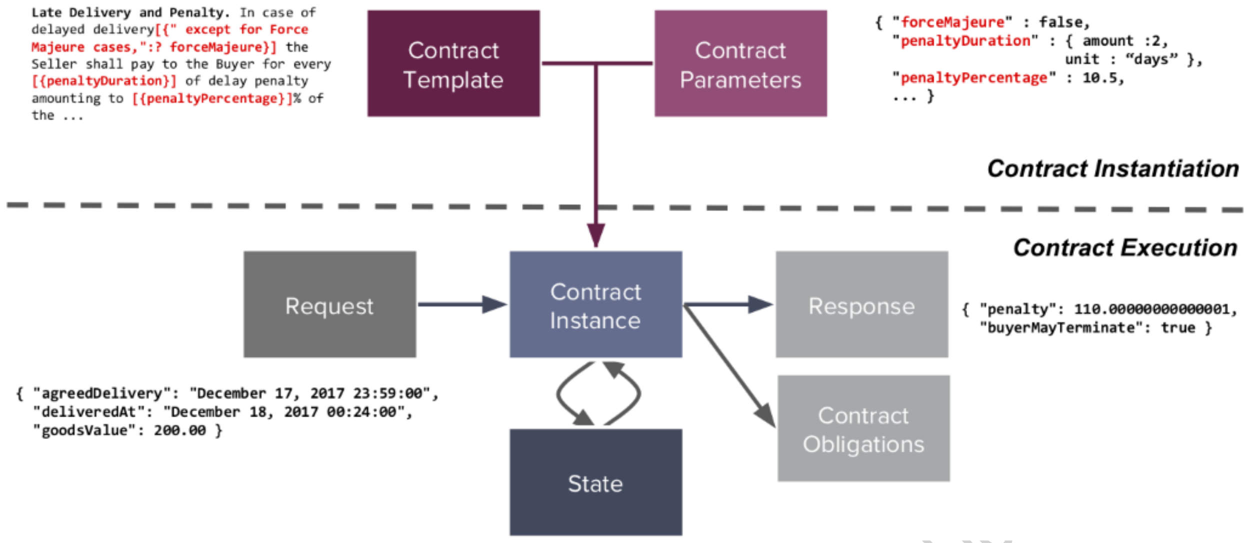

Late Delivery and Penalty. In case of delayed delivery[{" except for Force Majeure cases,":? forceMajeure}] the Seller shall pay to the Buyer for every [{penaltyDuration}] of delay penalty amounting to [{penaltyPercentage}]% of the ...
Contract Template
Contract Parameters
{ "forceMajeure" : false, "penaltyDuration" : { amount :2, unit : "days" }, "penaltyPercentage" : 10.5, ... }
Contract Instantiation
Contract Execution
Request
Contract Instance
Response
{ "penalty": 110.00000000000001, "buyerMayTerminate": true }
{ "agreedDelivery": "December 17, 2017 23:59:00", "deliveredAt": "December 18, 2017 00:24:00", "goodsValue": 200.00 }
State
Contract Obligations

Figure 3-18: Example Accord contract execution context

3.3.2.2 *Accord for 5GZORRO prose templates*

For 5GZORRO stakeholders to define templates that align with the chosen TM Forum Open API model specifications [44][45][13], a set of models will be developed to mirror the key relevant information models (see below) and packaged in a distributable NPM package. Using these, a template developer will be able to model agreement templates to meet their needs whilst remaining aligned with the underlying expectation that Agreements and SLAs conform to the TM Forum specification; naturally templates are subject to verification of both the technical implementation and legal text prior to approval in any case.

Accord will be used to encapsulate the following categories of prose, where concerto models mirror the noted TM Forum information models:

Prose Type	TMF Spec	Info Models
Agreement	TMF651	Agreement
SLA	TMF623	ServiceLevelAgreement, SLAViolation
License Terms	TMF620	productOfferingPrice

3.3.2.3 *Contract logic (Ergo)*

Ergo logic that encapsulates contract logic will be defined for SLAs such that a request can be posted to the SLA and should this correspond with a violation of the terms, an event (obligation) will be raised. It is intended that this logic is executed in a Trusted Execution Environment, in a serverless function fashion as part of the SLA monitoring/analysis pipeline. Subscribers for emitted SLA violations (contract obligation events) will be able to consume these events. For example, the Smart Contract Lifecycle Manager will subsequently post the violation to the DLT for recording.

It was decided that contract business logic should be executed off-chain (i.e., not on the DLT) for efficiency. Most notably the DLT only needs to be notified when a violation has occurred. Any inconsequential 'good performance' can be recorded in the Data Lake if necessary.

3.3.3 Specific and Relevant Workflows

The services offered by Legal Prose Management contemplate the following supported workflows:

3.3.3.1 *Publish Prose Template*

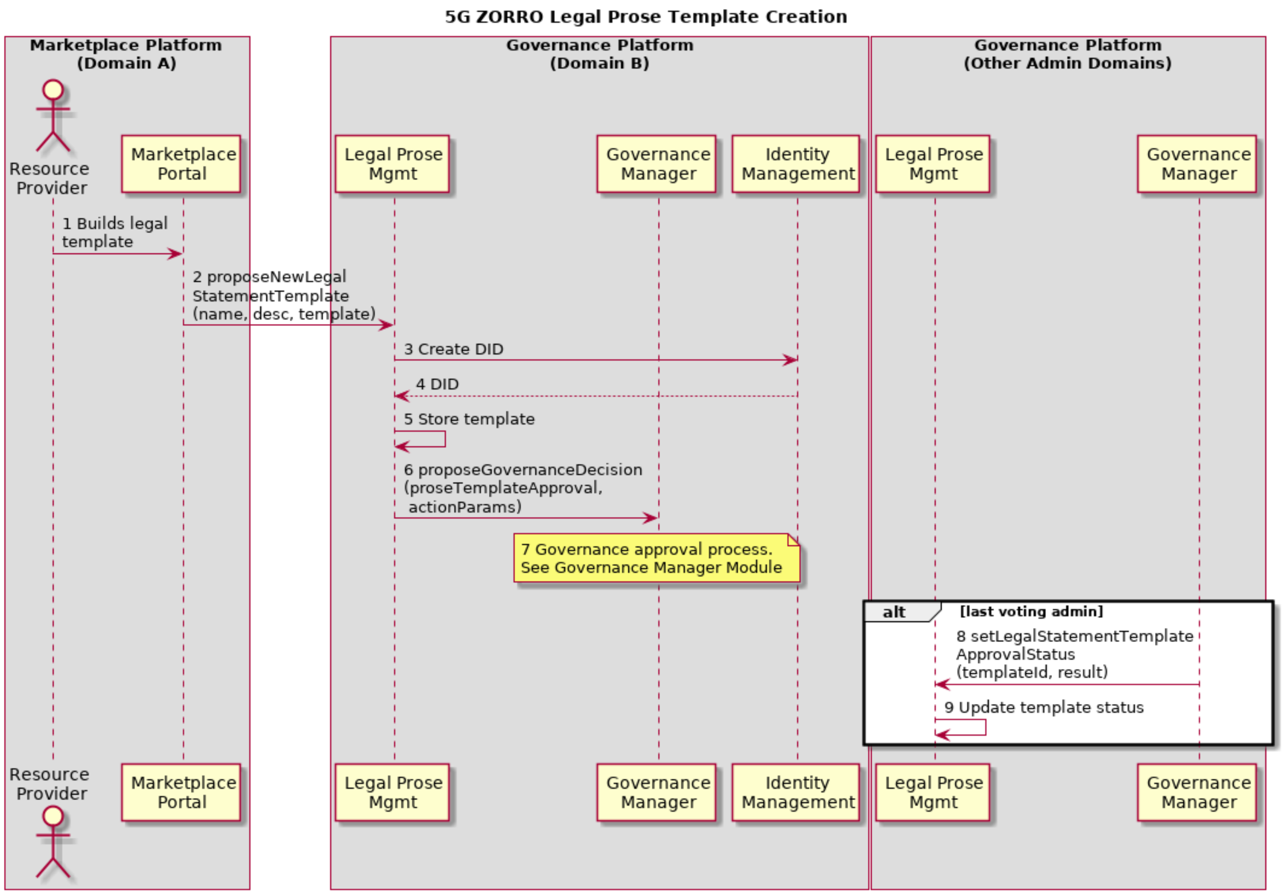

5G ZORRO Legal Prose Template Creation
Marketplace Platform (Domain A)
Governance Platform (Domain B)
Governance Platform (Other Admin Domains)
Resource Provider
Marketplace Portal
Legal Prose Mgmt
Governance Manager
Identity Management
Legal Prose Mgmt
Governance Manager
1 Builds legal template
2 proposeNewLegalStatementTemplate (name, desc, template)
3 Create DID
4 DID
5 Store template
6 proposeGovernanceDecision (proseTemplateApproval, actionParams)
7 Governance approval process. See Governance Manager Module
alt
[last voting admin]
8 setLegalStatementTemplateApprovalStatus (templateId, result)
9 Update template status
Resource Provider
Marketplace Portal
Legal Prose Mgmt
Governance Manager
Identity Management
Legal Prose Mgmt
Governance Manager

Figure 3-19: Publish Prose Template Creation

It is expected that prose template definition - whether for agreements, SLAs or license terms - will be a task that involves multiple user personas including business, legal and technical functions. The Governance Portal will provide the necessary UI to build and submit Accord prose templates.

Step 1-2: Stakeholder defines Accord template according to their requirements (prose, model and ergo logic) and submits the template to Legal Prose Management API endpoint exposed by a governance administrator 5GZORRO platform for approval.
Step 3-5: A DID is created for identification purposes and the template stored in distributed storage with a status of "PROPOSED".
Step 6-7: A new proposal request is made to the governance manager on which each governance admin will review and vote (via the Governance portal)[2]. See also Governance Manager for the details of this process.
Step 8-9: The template status will be updated accordingly and if approved become available for use in the marketplace by stakeholders.

3.3.3.2 *Archive Prose Template*

The process of archiving a template will be much the same to the publishing of one. If the decision is to reject the proposal to archive, then the template will continue to be available.

[2] Some legal terminology related to spectrum trading, cannot be subject to a vote as it has to be in line with enforceable legal frameworks

3.3.4 APIs

Operation name: **getLegalStatementTemplates**		
Description	API endpoint to retrieve a filtered list of legal prose templates to base an Agreement, SLA or Licensing Term on	
Input Parameters	**Type**	**Description**
Criteria	string	String to be used to filter the result set. E.g., “Spectrum” to retrieve templates relating to spectrum agreements
Output Parameters	**Type**	**Description**
templates	List<LegalStatementTemplate>	A list of legal statements for the requester to select from
Notes		

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

Operation name: **proposeNewLegalStatementTemplate**		
Description	API endpoint to create a new template.	
Input Parameters	**Type**	**Description**
name	String	The name to assign to the template
description	String	A description of what the legal template relates to
template	Blob	Accord archive file (.cto) that encapsulates the parameterised specification of a Ricardian contract
Output Parameters	**Type**	**Description**
id	String	The DID of the newly created template
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		

Operation name: **removeLegalStatementTemplate**		
Description	API endpoint to archive a template definition (soft delete)	
Input Parameters	**Type**	**Description**
id	String	DID of the template to be archived
Output Parameters	**Type**	**Description**
N/A		

Notes
The removed template would continue to be available until a governance process has approved its removal.

Operation name: **setLegalStatementTemplateApprovalStatus**			
Description		API endpoint to update the status of a given template definition	
Input Parameters		**Type**	**Description**
	id	String	DID of the template
	accept	Boolean	Flag indicating whether the template action should be approved or rejected
Output Parameters		**Type**	**Description**
Notes			
If the template is in "PROPOSED" state then it will move to either "ACTIVE" or "REJECTED", or if in "ARCHIVE_PROPOSED" state it will move to "ACTIVE" or "ARCHIVED" depending on the value of 'accept'.			

3.3.5 Information Models

3.3.5.1 *LegalStatementTemplate*

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

Parameter	Type	Description
id	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
template	Blob	Accord archive (.cto) containing the legal prose 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 ARCHIVE_PROPOSED | ARCHIVED
statusUpdated	DateTime	The date & time the template status was updated
archived	DateTime	The date & time the template was archived

3.3.5.2 *Accord project CTO Archive*

An accord archive comprises of the following elements that comprise a smart agreement template:

- **Cicero Template** - written in TemplateMark - a markdown format that extends CommonMark - to write contract & clause templates. It provides markdown extensions to facilitate parameterisation (and validation) of legal text based on the types defined in a Concerto model. The template contains the parameterised, human-readable text and is backed by the machine-readable concerto model
- **Concerto model** - Concerto is an object-orientated modelling language that allows you to define the model for a contract or clause, as well as any other required types, transactions and states.
- **Ergo logic** - Ergo is a domain-specific language to enable legal-tech developers to express contractual logic in legal contracts. It will be used to define the logic associated with calculating, for example, whether an SLA has been breached or a request is within the terms of a license agreement.

3.4 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 another subset of 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 model dictating the different steps to be validated before approval (business logic) is coded as smart contracts which are deployed on-chain. Thus, 5GZORRO solution automates the business processes that should take place after a final decision has been made, i.e., multiple voters (those with voting power) have collectively voted for or against the specific request. The governance model itself differs for each type of request: one may require 3 out of 4 positive votes, all weighting the same (same voting power), while others may, for instance, require at least 1 positive vote to go through – all of these are properly decided collectively and off-chain, before being deployed on-chain.

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 shall support the following user stories, whose actions should yield an expected behaviour (based on the interfaces and underlying functionality of associated software modules).

It is clear, thus, that at the Governance Portal, a web GUI component will mostly interact with three other major components: Identity and Permissions Manager (Section 3.2), Governance Manager (Section 3.1) and the Legal Prose Repository (Section 3.2.53.3).

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 Governance Board (list of Governance Admins)	**DLT Governance Manager**	*applyforMembership()*	stakeholderDID; stakeholderClaimCertificate; NotificationMethod (5GZORRO IM)	VC (Governance Admin onboarding)
R2	As a Stakeholder trying to join the Marketplace, I must be able to watch the status of my onboarding process	**DLT Governance Manager**	*checkMembershipStatus()*	stakeholderDID	*MembershipStatus* (5GZORRO IM)
R3	As a Stakeholder I must be able to revoke my access to the Marketplace	**DLT Governance Manager**	*revokeMembership*	stakeholderDID	
R4	As a Stakeholder, I must propose the revoking of another stakeholder's access to the Marketplace	**DLT Governance Manager**	*revokeMembership*	stakeholderDID	
R5	As a Stakeholder, I must be able to submit and propose new Governance Decisions	**DLT Governance Manager**	*proposeGovernanceDecision()*	actionType & actionParams (5GZORRO IM) (based on the selected action, a dropdown form will be displayed, with a different set of inputs)	*proposalIdentifierDID*
R6	As a Governance Admin, I must be able to vote on other Stakeholders' submitted Governance Decisions	**DLT Governance Manager**	*voteGovernanceDecision()*	DID Boolean	
R7	As a Stakeholder, I must be able to check the decision status of all previously submitted proposals	**DLT Governance Manager**	*getGovernanceDecision()*	DID	*GovernanceProposalStatus* (5GZORRO IM)

US id	User Story (US)	SW Module (Interface)	API endpoint / RPC	Input	Outputs
R8	As a Stakeholder, I must be able to query all previously submitted proposals	**DLT Governance Manager**	*getProposals()*		
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 a particular Smart Legal Template	**Legal Prose Repository**	*getLegalStatementTemplate()*	Template DID	*LegalStatementTemplate (5GZORRO IM)*
RL4	As a Governance Admin, I must be able to approve and reject previously submitted Smart Legal Templates	**Legal Prose Repository**	*setLegalStatementTemplateApprovalStatus()*	A checkbox to indicate the accept parameter of a particular Template ID (possibly listed as output of getLegalStatementTemplate())	
RL5	As a Governance Admin, I must be able to propose the archival of previously submitted Smart Legal Templates	**Legal Prose Repository**	*removeLegalStatementTemplate()*	Template ID	

4 Trustworthy Marketplace applications

The stringent requirements and dynamic adaptation introduced by next-generation networks impose the need to simplify the processes for procurement, deployment and management required for building 5G flexible networks. Motivated by this reality, the 5GZORRO Marketplace Platform aims precisely to foster multi-party collaboration for on-demand end-to-end service provisioning in dynamic 5G environments. This section describes the main functionalities and components of the 5GZORRO Marketplace Platform.

In general terms, the Marketplace Platform enables the creation and 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 this, 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.

Given the decentralized nature of the 5GZORRO ecosystem, the Marketplace Platform is the result of a mesh of distributed marketplace instances like the one depicted in Figure 4-1. Specifically, the Marketplace architecture needs to include the following components, whose main functionalities are next outlined, and a more in-depth description is provided for each one of them later in this section.

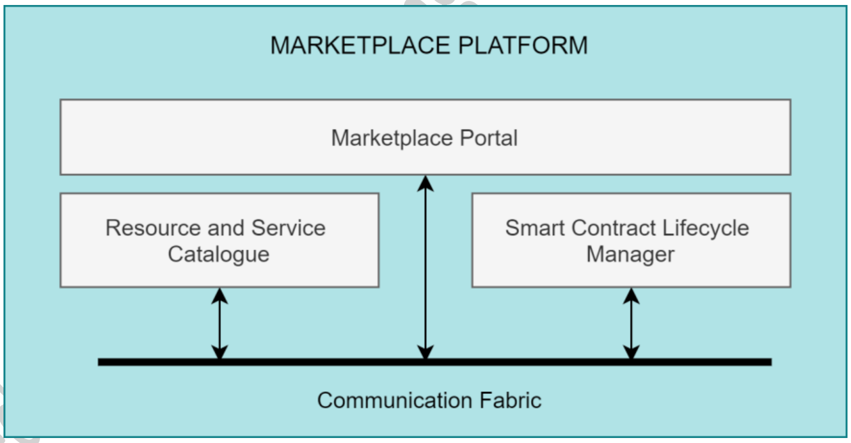

MARKETPLACE PLATFORM
Marketplace Portal
Resource and Service Catalogue
Smart Contract Lifecycle Manager
Communication Fabric

Figure 4-1: Marketplace Platform Architecture

- ***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 (Offer) 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 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 [7], this component facilitates the interaction among Marketplace services by playing both the roles of service consumer and service producer.

4.1 Resource and Service Offer Catalogue

As part of the 5GZORRO Marketplace Platform, the Resource and Service Offer Catalogue 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 asset market is mainly supported by three different catalogues:

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 candidates are then wrapped into commercial product offers, stored in the product catalogue and exposed to the market by means of smart contracts.

4.1.1 5GZORRO Specific enhancements

In 5GZORRO, the Resource and Service Offer Catalogue 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 Resource and Service Offer Catalogue 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 includes the associated commercial agreement between providers and consumers.

4.1.2 Design Details

The Resource and Service Offer 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 Catalogue, shown in Figure 4-2, comprises the following main entities:

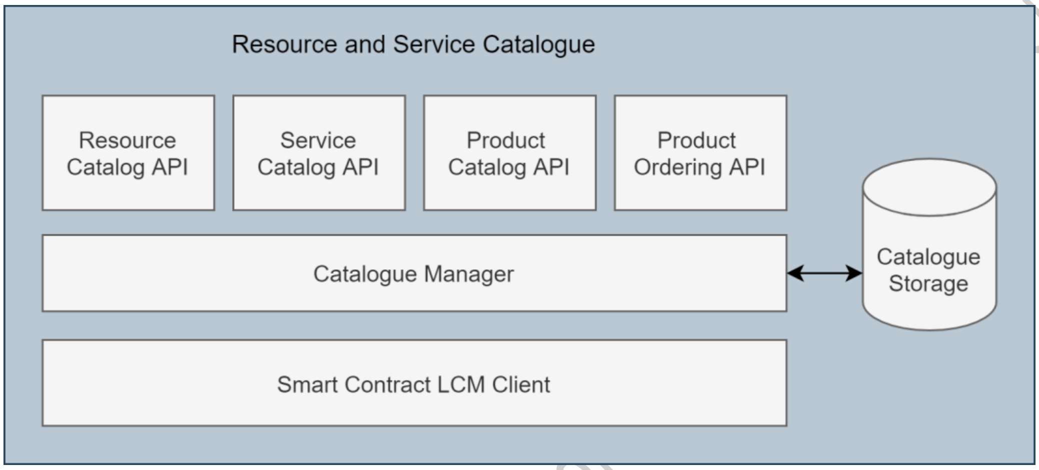


Figure 4-2: Resource and Service Catalogue Architecture

- Catalogue Northbound Interface (NBI): a set of TM Forum APIs is implemented to expose the module capabilities regarding the lifecycle management of offers and orders. More details about these APIs are depicted in the following subsections.
- Catalogue Manager: Functional block that implements the logic of the 5GZORRO Catalogue, triggering internal interaction (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 are stored. The information model adopted is compliant with the TM Forum SID and depicted in more details in following subsections.
- Smart Contract LCM Client: is the entity in charge of interacting with the Smart Contract Lifecycle Manager (LCM), triggering API calls to conduct the deployment of smart contracts related to a product offer that becomes available to be traded, as well as the update and removal of previously deployed smart contracts.

4.1.3 Specific and relevant workflows

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

4.1.3.1 *On-boarding of (Resource and Service) Assets*

Resource providers on-board and store in the Catalogue the technical description of available assets as shown in Figure 4-3 for the case of a resource asset.

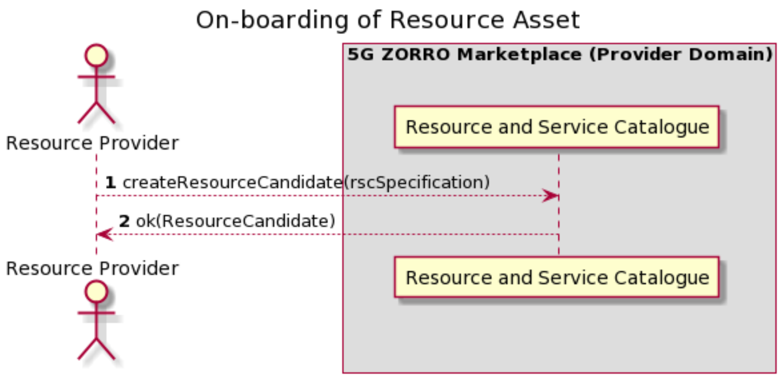

On-boarding of Resource Asset
Resource Provider
5G ZORRO Marketplace (Provider Domain)
Resource and Service Catalogue
1 createResourceCandidate(rscSpecification)
2 ok(ResourceCandidate)
Resource Provider
Resource and Service Catalogue

Figure 4-3: On-boarding of resource asset workflow

Given the diverse nature of considered resources, the provided asset specification (see step 1 in Figure 4-3) 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 (i.e., Radio/Virtual Resource Manager) 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.

4.1.3.2 *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 4-4.

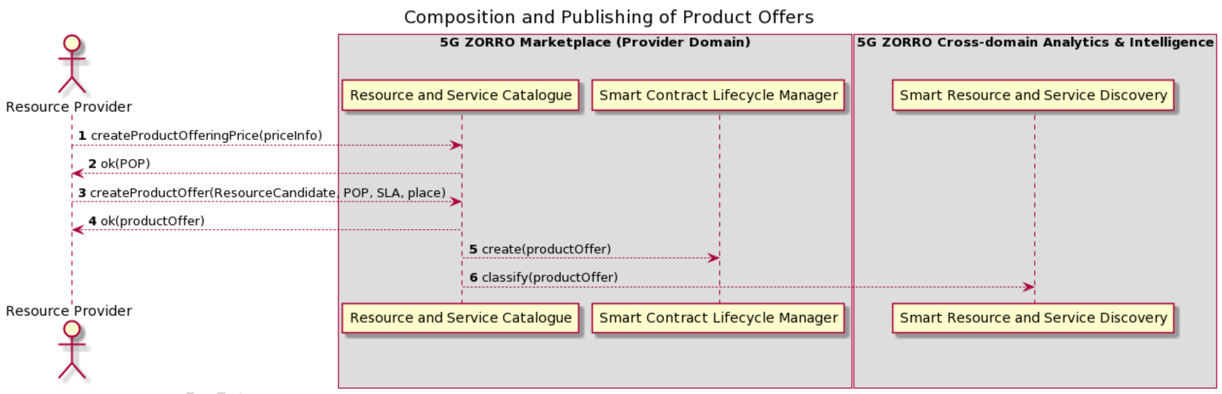

Composition and Publishing of Product Offers
Resource Provider
5G ZORRO Marketplace (Provider Domain)
5G ZORRO Cross-domain Analytics & Intelligence
Resource and Service Catalogue
Smart Contract Lifecycle Manager
Smart Resource and Service Discovery
1 createProductOfferingPrice(priceInfo)
2 ok(POP)
3 createProductOffer(ResourceCandidate, POP, SLA, place)
4 ok(productOffer)
5 create(productOffer)
6 classify(productOffer)
Resource Provider
Resource and Service Catalogue
Smart Contract Lifecycle Manager
Smart Resource and Service Discovery

Figure 4-4: Composition and publishing of product offers workflow

Step 1-2: In addition to the description of associated assets, pricing information is also required. Following a consistent and standardized component definition, involved dependencies such as the product-offering price (POP), can be created once and reused many times as part of new product offerings.

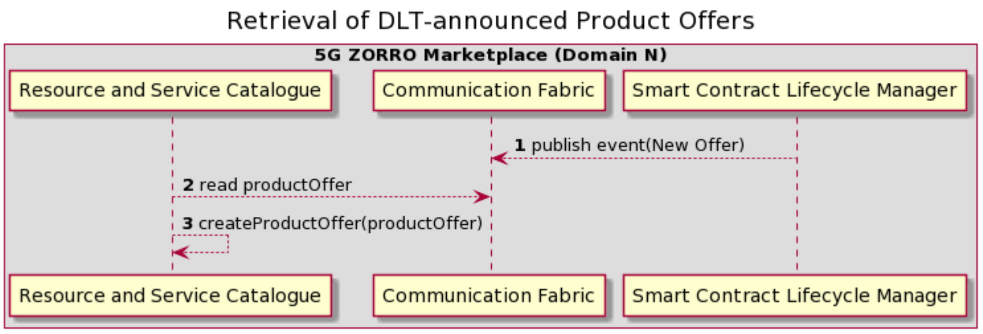
Step 3-4: During this stage, the SLA corresponding to the conceived product is also associated to the offer, which is retrieved from the Smart Contract Lifecycle Manager, where an SLA manager sub-module is in charge of conforming a variety of SLAs based on suitable contract templates.

Step 5: Once the product offer is assembled, the Catalogue interacts with the Smart Contract Lifecycle Manager for the deployment and commitment into the ledger of the smart contract related to the new product offering.

Step 6: Likewise, the newly created product offer is ingested into the Smart Resource and Service Discovery application for the execution of ML-based classification later explained in Section 5.5.
In the case of spectrum trading, the composition and publishing of product offer might need to be vetted offline and may be subject to the approval of the Regulator in particular to safeguard from competition distortion ex-ante.

4.1.3.3 *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 relating to offer registration/update/removal as depicted in Figure 4-5.



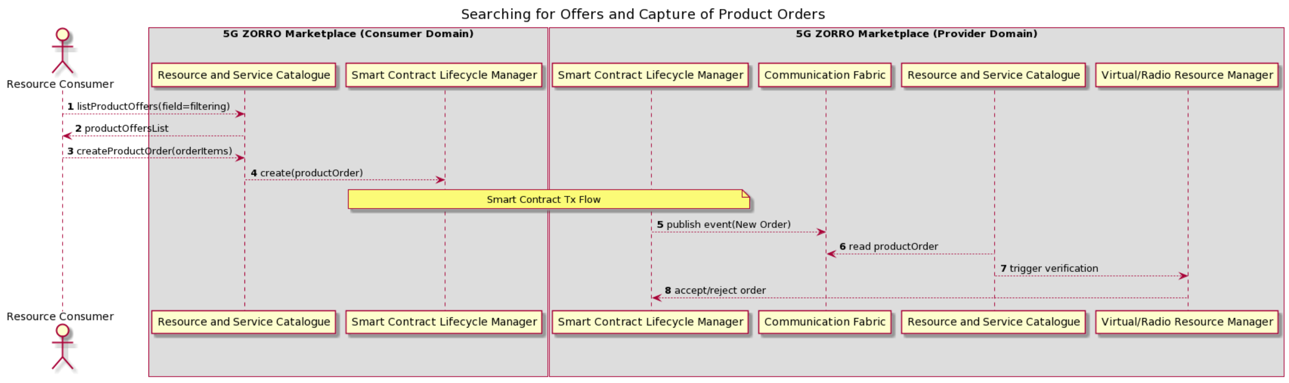
Retrieval of DLT-announced Product Offers
5G ZORRO Marketplace (Domain N)
Resource and Service Catalogue
Communication Fabric
Smart Contract Lifecycle Manager
1 publish event(New Offer)
2 read productOffer
3 createProductOffer(productOffer)
Resource and Service Catalogue
Communication Fabric
Smart Contract Lifecycle Manager

Figure 4-5: 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.

4.1.3.4 *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 4-6.



Searching for Offers and Capture of Product Orders
5G ZORRO Marketplace (Consumer Domain)
5G ZORRO Marketplace (Provider Domain)
Resource Consumer
Resource and Service Catalogue
Smart Contract Lifecycle Manager
Smart Contract Lifecycle Manager
Communication Fabric
Resource and Service Catalogue
Virtual/Radio Resource Manager
1 listProductOffers(field=filtering)
2 productOffersList
3 createProductOrder(orderItems)
4 create(productOrder)
Smart Contract Tx Flow
5 publish event(New Order)
6 read productOrder
7 trigger verification
8 accept/reject order
Resource Consumer
Resource and Service Catalogue
Smart Contract Lifecycle Manager
Smart Contract Lifecycle Manager
Communication Fabric
Resource and Service Catalogue
Virtual/Radio Resource Manager

Figure 4-6: Searching for offers and capture of product orders workflow

Step 1-2: Advertised product offers can be searched by filtering them based on several criteria such as category, provider, and geographic location, among others.
Step 3-4: After performing a selection, the customer places a request specifying the corresponding offer,

which is relayed to the Smart Contract Lifecycle Manager for its translation as smart contract and transmission towards the provider domain.

Step 5-7: This new record triggers a feasibility check at the Catalogue instance acting as offer provider, which is delegated to the associated resource management entities, to confirm whether the corresponding product (i.e., availability and sufficient capacity of the involved resources to support the product order) can still be delivered.

Step 8: The result is then propagated back to the customer, which, if feasible, proceeds with the required orchestration actions.

4.1.4 APIs

To facilitate the handling of digital assets, the Resource and Service Offer 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)*** [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 Resource catalogue.
- ***Service Catalog Management API (TMF633)*** [12]: 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)*** [13]: 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)*** [14]: 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.

4.2 Smart Contracts Lifecycle Manager

This module comprises a set of services for managing agreements, SLAs, licensing terms and products 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. Agreement and SLA encapsulate TM Forum API service 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.

4.2.1 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 and SLA template 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 remain 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.

Catalogue 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 assets (agreements, SLAs and License Terms) are published to the DLT such that relevant parties can come to agreement over the terms of the order, and subsequently enter into an immutable agreement. From there the lifecycle of the agreement, SLAs and License terms 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.

4.2.2 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 below comprises the following main entities:

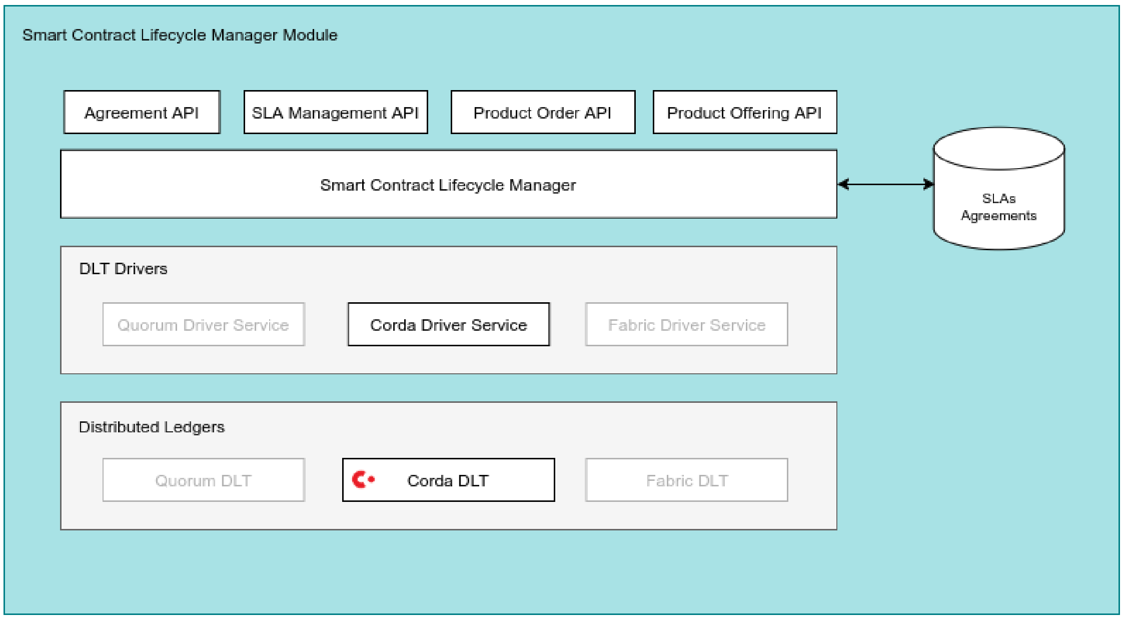


Figure 4-7: Smart Contract Lifecycle Manager Module Architecture

Agreement API: implementation of TM Forum APIs as defined in TMF651 – Agreement Management [44] for providing CRUD capabilities for Agreement definitions that can be associated with product orders. Agreements are created off-chain and only published to the DLT when a product order is made.

SLA Management API: implementation of TM Forum APIs as defined in TMF623 – SLA Management [45] 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.

Product Order API: partial interface implementation of TM Forum APIs as defined in TMF622 – Product Order API [14], 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.

Product Offering API: partial interface implementation of TM Forum APIs as defined in TMF620 – Product catalogue management [13], 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

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 Agreement and SLA entities are persisted and queried from to support their inclusion in product offering and product order 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.

4.2.3 Specific and relevant workflows

4.2.3.1 *Agreement definition*

Resource providers create Agreement definitions based on templates defined in the Legal Prose Management Repository such that they can then be incorporated into product offerings / orders and published in the catalogue. The creation of Agreements 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 Agreement is based upon, tying the technical definition to the human-readable prose equivalent.

4.2.3.2 *SLA definition*

Resource providers create SLA definitions based on templates defined in the Legal Prose Management Repository such that they can then be incorporated into product offerings / orders and published in the catalogue. The creation of SLAs 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 is based upon, tying the technical definition to the human-readable prose equivalent.

4.2.3.3 *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. Where necessary – such as when ensuring Spectrum rights – the appropriate verifiable credential is supplied for it to be verified by the DID Oracle; failure resulting in the transaction not being signed by the oracle and the ledger update rejected. The figure below illustrates the steps involved.

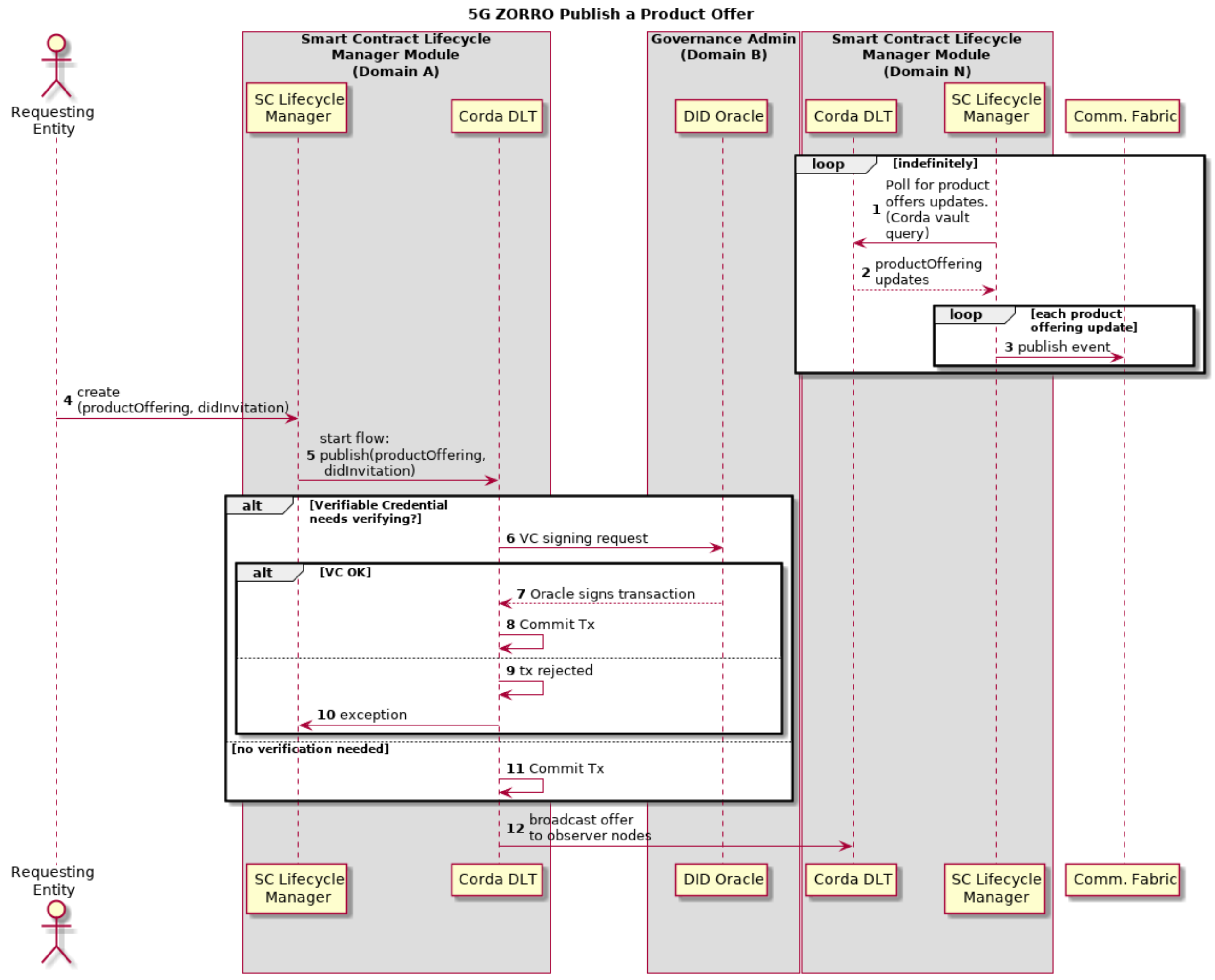

5G ZORRO Publish a Product Offer
Requesting Entity
Smart Contract Lifecycle Manager Module (Domain A)
SC Lifecycle Manager
Corda DLT
Governance Admin (Domain B)
DID Oracle
Smart Contract Lifecycle Manager Module (Domain N)
Corda DLT
SC Lifecycle Manager
Comm. Fabric
loop [indefinitely]
1 Poll for product offers updates. (Corda vault query)
2 productOffering updates
loop [each product offering update]
3 publish event
4 create (productOffering, didInvitation)
5 start flow: publish(productOffering, didInvitation)
alt [Verifiable Credential needs verifying?]
6 VC signing request
alt [VC OK]
7 Oracle signs transaction
8 Commit Tx
9 tx rejected
10 exception
[no verification needed]
11 Commit Tx
12 broadcast offer to observer nodes

Figure 4-8: Publish a Product Offer

4.2.3.4 *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, the order comprises of metadata, resource or service offers, SLAs and agreements. 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 should be 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. The workflow steps are illustrated in the figure below:

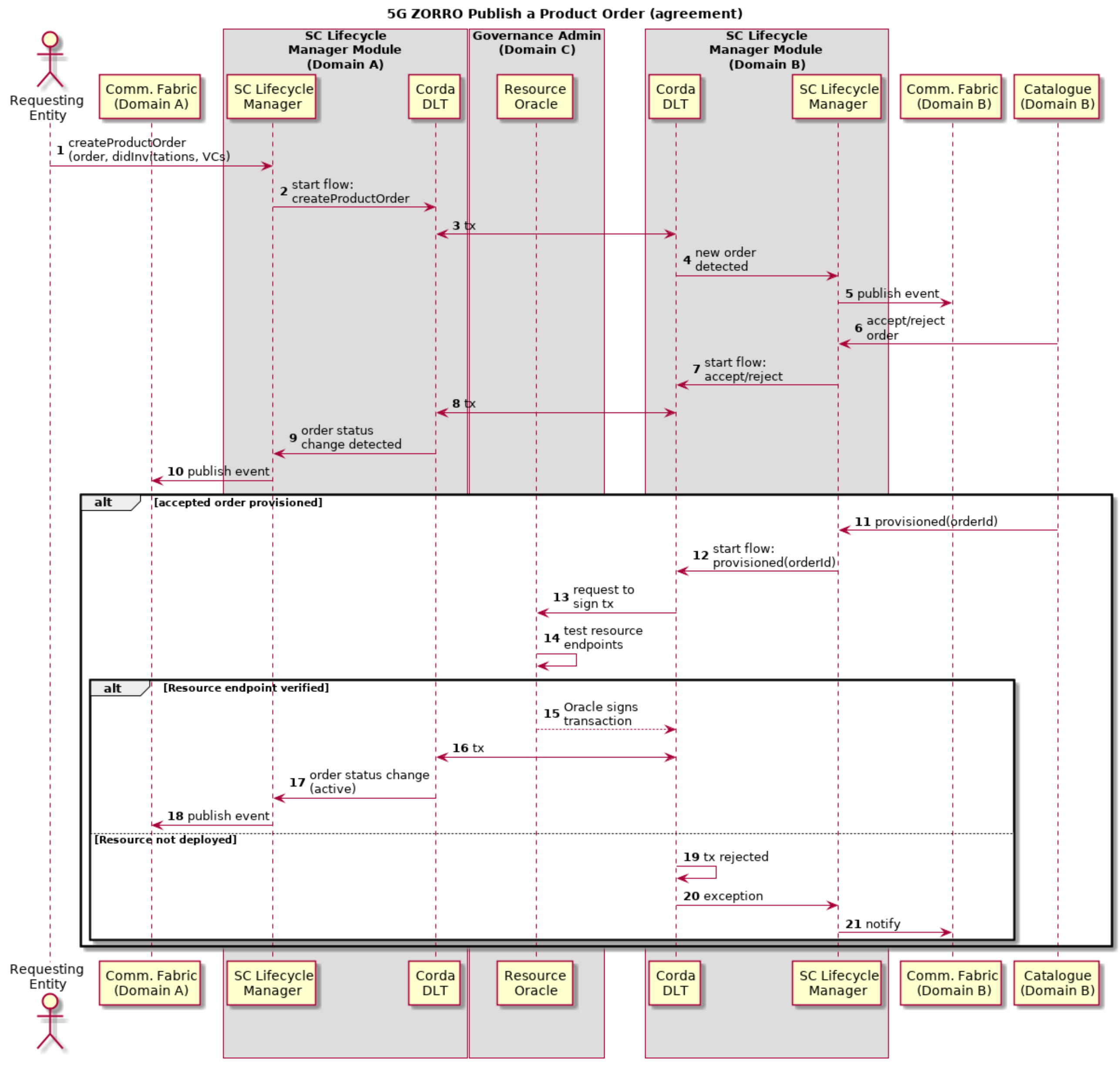

5G ZORRO Publish a Product Order (agreement)
SC Lifecycle Manager Module (Domain A)
Governance Admin (Domain C)
SC Lifecycle Manager Module (Domain B)
Requesting Entity
Comm. Fabric (Domain A)
SC Lifecycle Manager
Corda DLT
Resource Oracle
Corda DLT
SC Lifecycle Manager
Comm. Fabric (Domain B)
Catalogue (Domain B)
1 createProductOrder (order, didInvitations, VCs)
2 start flow: createProductOrder
3 tx
4 new order detected
5 publish event
6 accept/reject order
7 start flow: accept/reject
8 tx
9 order status change detected
10 publish event
alt [accepted order provisioned]
11 provisioned(orderId)
12 start flow: provisioned(orderId)
13 request to sign tx
14 test resource endpoints
alt [Resource endpoint verified]
15 Oracle signs transaction
16 tx
17 order status change (active)
18 publish event
[Resource not deployed]
19 tx rejected
20 exception
21 notify

Figure 4-9: 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 8-10: Having validated the order and their ability to service it, the provider initiates the accept or reject flow via the Smart Contract Lifecycle Manager.

Step 11-14: The status of the order is updated on the DLT and the consumer is notified.

Step 15-17: Having provisioned the appropriate resources for the order, either the provider or consumer can give acknowledgement to the Smart Contract Lifecycle Manager (via the Catalogue*) to initiate the provisioned flow.

Step 18-19: Resource proxy oracle tests the endpoints for each resource offer in the order and signs the transaction (tx) as successful.

Step 21-25: Successful oracle verification leads to submission of the status update to the ledger and the agreement becomes active. Consumer is notified.

Step 25-28: If endpoint testing fails, the transaction is rejected, and the provider is notified.

** The current thinking is that the Catalogue will act as a proxy between Core Platform services such as Intelligent Network Slice & Service Manager, subscribing to events raised by these components and initiating the appropriate interaction with the Smart Contract Lifecycle Manager.*

4.2.3.5 *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 below workflow depicts this monitoring process and the instantiation/teardown of monitoring pipelines by the monitoring manager.

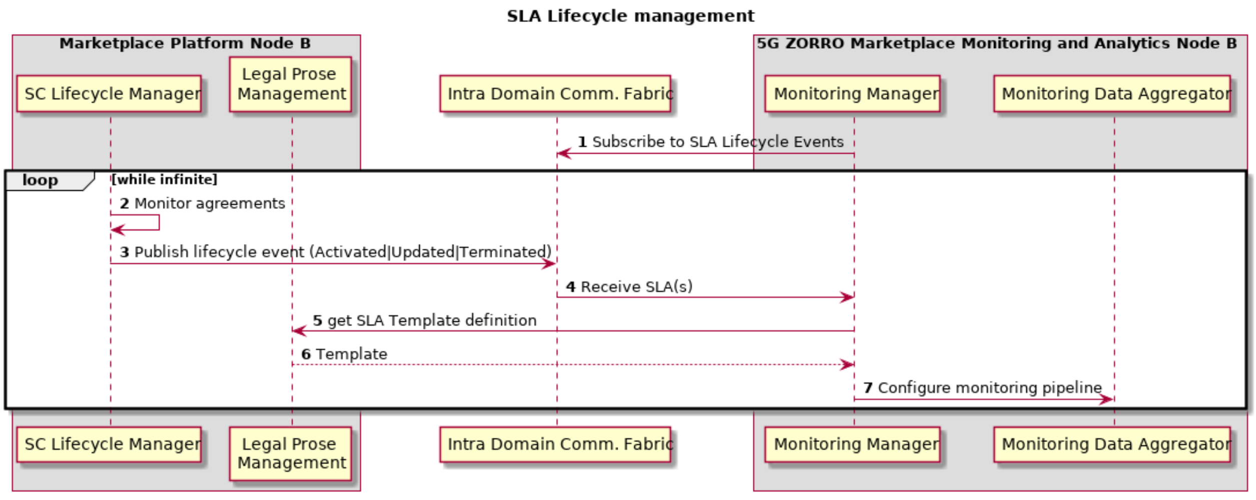

SLA Lifecycle management
Marketplace Platform Node B
SC Lifecycle Manager
Legal Prose Management
Intra Domain Comm. Fabric
5G ZORRO Marketplace Monitoring and Analytics Node B
Monitoring Manager
Monitoring Data Aggregator
1 Subscribe to SLA Lifecycle Events
loop
[while infinite]
2 Monitor agreements
3 Publish lifecycle event (Activated|Updated|Terminated)
4 Receive SLA(s)
5 get SLA Template definition
6 Template
7 Configure monitoring pipeline

Figure 4-10: SLA Lifecycle Management

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

Step 2: Lifecycle Manager monitors Smart Contract agreements for changes to agreements and related SLAs.

Step 3: Lifecycle Manager publishes related events as contract changes occur (e.g., new SLA, updated SLA).

Step 4: Monitoring Manager receives changes to SLAs that are to be monitored.

Step 5-6: Monitoring Manager retrieves associated SLA template, which contains the executable contract logic.

Step 7: Monitoring Manager configures the Monitoring Aggregator according to the SLAs it is required to monitor.

Note: Create/update should configure the Monitoring 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. The workflow below illustrates the steps involved and the temporal assessment of SLA performance.

Steps 1-2: Resource Provider monitors resources and publishes metrics to Service & Resource Monitoring.

Step 3-4: Monitoring data is aggregated and published to the Data Lake.

Step 5: Analysis is performed on the aggregated data temporally using a function derived from the Legal Prose Template relating to the SLA in order to detect violations.

Step 6: If an SLA violation is detected and event is emitted by the analysis function and the event is published to the communication fabric for subscribing apps to consume. Included in the payload is a hash of evidence & VC for the monitoring service (i.e., proof that it is permitted to submit a violation for the SLA).

Steps 7-9: Smart Contract Lifecycle Manager consumes the Violation Event and records the violation on the DLT with the Identity Oracle attesting to the fact that the VC is valid.

Step 10-14: If the VC is valid the oracle and provider sign the transaction and it is synced with the consumer as an observer of the transaction. The state change is emitted to the Smart Contract Lifecycle manager of both provider and consumer.

Step 15: If the VC is invalid, i.e., unauthorised to update the Smart Contract, then the transaction is rejected.

Steps 16-17: Smart Contract Lifecycle Manager publishes the DLT event for marketplace subscribers (both in the consumer and provider domains).

Steps 18-19: When the agreement is terminated, the Smart Contract Lifecycle Manager publishes the DLT event for marketplace subscribers (both in the consumer and provider domains).

Whilst the 5GZORRO exemplary implementation does not consider an impartial stakeholder to perform trusted monitoring, it is envisaged that this role would be identified and provide the monitoring capability rather than the provider depicted in the workflow above. The solution remains agnostic to this however and would only require the third party's monitoring service to be identified in the product order in place of the providers. Consequently, it would be the mutually trusted third party (having deployed 5GZORRO components) that would monitor, measure and record violations pertaining to SLAs. For the purposes of 5GZORRO, the issue of trust around SLA monitoring and measurement is somewhat mitigated by the execution of rules within a trusted execution environment, and through the governance model, whereby a consumer can raise a dispute should they feel that a provider has acted dishonestly.

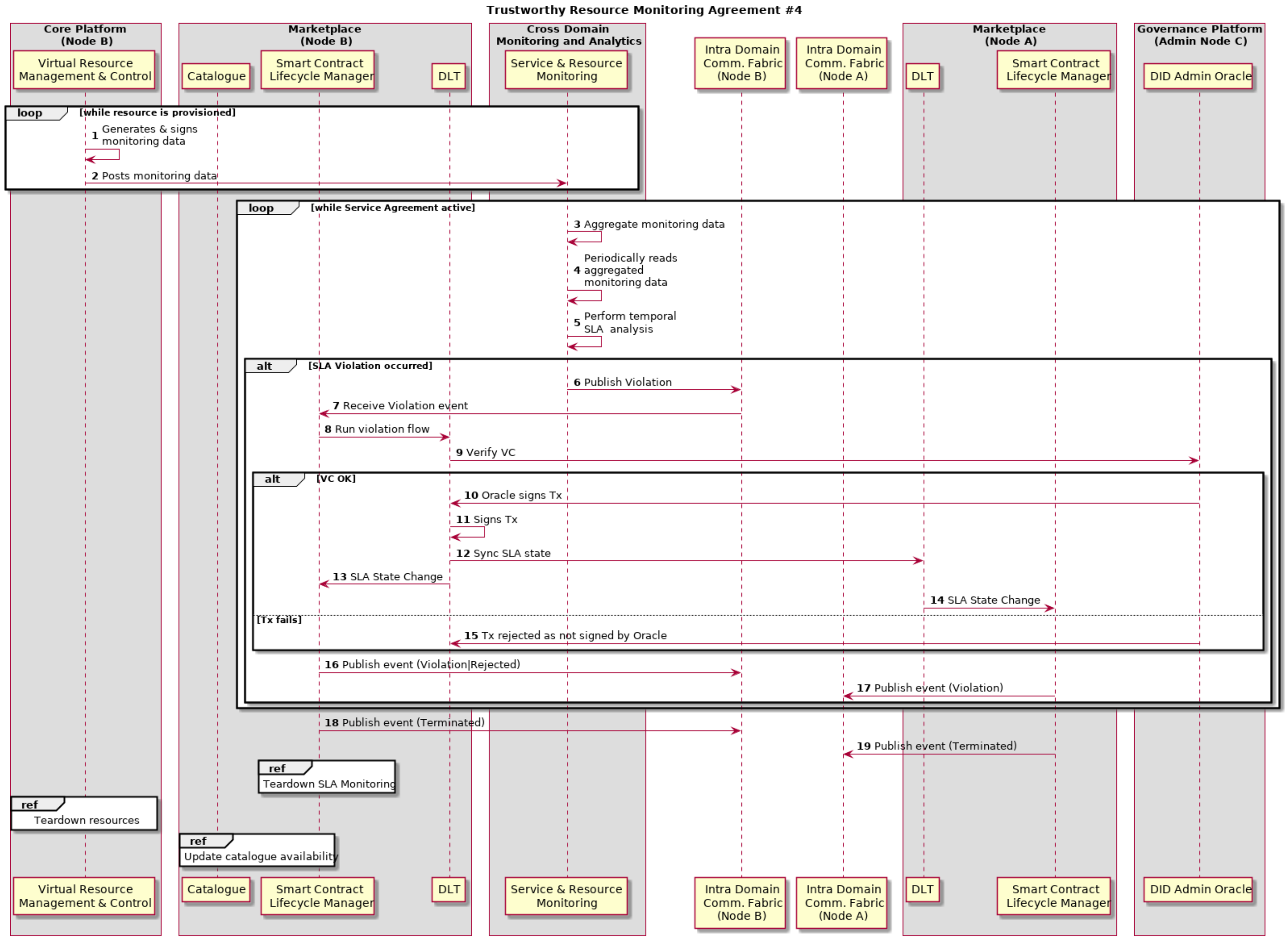

Trustworthy Resource Monitoring Agreement #4
Core Platform (Node B)
Virtual Resource Management & Control
Marketplace (Node B)
Catalogue
Smart Contract Lifecycle Manager
DLT
Cross Domain Monitoring and Analytics
Service & Resource Monitoring
Intra Domain Comm. Fabric (Node B)
Intra Domain Comm. Fabric (Node A)
Marketplace (Node A)
DLT
Smart Contract Lifecycle Manager
Governance Platform (Admin Node C)
DID Admin Oracle
loop [while resource is provisioned]
1 Generates & signs monitoring data
2 Posts monitoring data
loop [while Service Agreement active]
3 Aggregate monitoring data
4 Periodically reads aggregated monitoring data
5 Perform temporal SLA analysis
alt [SLA Violation occurred]
6 Publish Violation
7 Receive Violation event
8 Run violation flow
9 Verify VC
alt [VC OK]
10 Oracle signs Tx
11 Signs Tx
12 Sync SLA state
13 SLA State Change
14 SLA State Change
[Tx fails]
15 Tx rejected as not signed by Oracle
16 Publish event (Violation|Rejected)
17 Publish event (Violation)
18 Publish event (Terminated)
19 Publish event (Terminated)
ref Teardown SLA Monitoring
ref Teardown resources
ref Update catalogue availability

Figure 4-11: Monitoring and SLA Compliance

4.2.4 APIs

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

The following TM Forum APIs will be adopted for the implementation:

- **Product Catalog Management API (TMF620):** 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)**: 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
- **Agreement Management API (TMF651):** provides the models, dependencies, lifecycle management operations (i.e., create, update, delete, read and list) and event notification capabilities to manage Agreement entities
- **SLA Management API (TMF623):** 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

4.2.4.1 *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 will expose:

Product Offerings

Operation name: **publishProductOffer**		
Description	API Endpoint for a provider to publish a new product offer to the DLT	
Input Parameters	**Type**	**Description**
offer	ProductOffering	TM Forum model capturing the product offering specification
didInvitations	Map<string, Invitation>	Mapping of DID to Invitation Objects (see Identity Management) for any DIDs that may require credential verification
verifiableCredentials	Optional<VerifiableCredential>	Optionally include applicable verifiable credentials for the offer. E.g., Spectrum rights VC required to successfully publish a spectrum offer
Output Parameters	**Type**	**Description**
N/A		
Notes		

Operation name: **updateProductOffer**		
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**
offer	ProductOffering	TM Forum model capturing the updated product offering specification

Output Parameters	**Type**	**Description**
N/A		
Notes		

Operation name: **removeProductOffer**		
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**
Notes		

Product Orders

Operation name: **getProductOrder**		
Description	API Endpoint for consumer or provider to get the details of a specific product order	
Input Parameters	**Type**	**Description**
productOrderId	String	DID of the product order
Output Parameters	**Type**	**Description**
productOrder	ProductOrder	TM Forum ProductOrder object
Notes		

Operation name: **getProductOrders**		
Description	API Endpoint for a provider or consumer to obtain a list of their – active – product orders (agreements)	
Input Parameters	**Type**	**Description**
activeOnly	Boolean	
Output Parameters	**Type**	**Description**
productOrders	List<ProductOrder>	List of TM Forum ProductOrder objects
Notes		

Operation name: **createProductOrder**		
Description	API Endpoint for a consumer to create a new product order and begin the associated lifecycle	
Input Parameters	**Type**	**Description**
productOrder	ProductOrder	TM Forum model capturing the product order specification

didInvitations	Map<string,Invitation>	Mapping of DID to Invitation Objects (see Identity Management) for any DIDs that may require credential verification
Output Parameters	**Type**	**Description**
N/A		
Notes		

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

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

Operation name: **proposeUpdateProductOrder**		
Description	API Endpoint for a provider or consumer to submit a product order update proposal	
Input Parameters	**Type**	**Description**
productOrderId	String	DID of the product order
productOrder	ProductOrder	TM Forum model capturing the updated product order specification
Output Parameters	**Type**	**Description**
N/A		
Notes		

Operation name: **acceptProposedProductOrderChanges**		
Description	API Endpoint for a counterparty to accept a proposed change to a product order	
Input Parameters	**Type**	**Description**
productOrderId	String	DID of the product order to accept proposed changes on

Output Parameters	Type	Description
N/A		
Notes		

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

Operation name: **terminateProductOrder**		
Description	API Endpoint for a consumer or provider to terminate an existing product order	
Input Parameters	**Type**	**Description**
productOrderId	String	DID of the product order to terminate
Output Parameters	**Type**	**Description**
Notes		

Operation name: **markProductOrderProvisioned**		
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**
productOrderId	String	DID of the product order to update
Output Parameters	**Type**	**Description**
Notes		

SLA Violation

Operation name: **createSLAViolation**		
Description	API Endpoint for a consumer to post violations received from the approved monitoring service	
Input Parameters	**Type**	**Description**
productOrderId	String	DID of the product order the violation relates to

violation	SLAViolation	A SIGNED (by the generating monitoring service) TM Forum SLAViolation model to support provenance of the violation payload
monitoringServiceVC	VerifiableClaim	Claim presentation to allow a governance Oracle to verify that the monitoring service does indeed have permission to post this violation to the ledger
Output Parameters	**Type**	**Description**
N/A		
Notes		

License Terms

Operation name: **setProvisionedQty**		
Description	API Endpoint for a consumer to update the DLT with any scaling action such that they are reflected on ledger, but also verified as being within the terms of the contract	
Input Parameters	**Type**	**Description**
productOrderId	String	DID of the product order
productOfferingId	String	DID of the product offering
newQty	Number	The number of instances/users required
Output Parameters	**Type**	**Description**
N/A		
Notes		

4.3 Marketplace portal

This portal is the module that comprises the unique 5GZORRO's web GUI: Governance (Section 3.4) and Marketplace portal. As mentioned in D2.2, 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 of such products into the Catalogue, the GUI must also interface directly with both the Resource and Service Offer Catalogue (Section 4.1) and, in a later stage, the Smart Contract Lifecycle Manager (Section 4.2) for subsequent deployment of said offers to the DLT. Furthermore, this GUI interacts also directly with the Smart Resource and Service Discovery (Section 5.5), allowing stakeholders to intelligently query and retrieve catalogue data and metrics, abstracting the underlying complex ML-based filtering and ordering.

4.3.1 Design Details

Following the same approach as in Governance portal (Section 3.4), the table below indicates the User Stories that shall drive the implementation of this 5GZORRO module (Web GUI).

Table 4-1: Definition of Marketplace Portal's User Stories

US id	**User Story** (US)	**SW Module & Interface/Operation** (Interface)		**Inputs and notes**	**Outputs**
US 1	As a Resource Consumer (Communication Service Provider), I must be able to browse the most suitable available offerings, based on specific criteria	Module	*Smart Resource and Service Discovery application*	*Discovery Criteria* (category, location, price, preference for provider)	List of Offers
		Operation	*discoverOffers()*		
US 2	*As a Resource Provider, I must be able to on-board resources into the Catalogue*	Module	*Resource and Service Offer Catalogue (Catalogue Manager)*	Type of resources to be selected before passing correct parameters: •InformationResource, •SpectrumResource, •PhysicalResource, •LogicalResource, •Virtual Resource, •NetworkFunction The onboarding of these resources should follow a "wizard" type of interface, guiding the user through different forms depending on selected items. More details on the parameters to be filled can be found in Section 6.2.1.	
		Operation	Create Resource method exposed by the Catalogue Manager		
US3	*As a Service Provider, I must be able to on-board services into the Catalogue*	Module	Resource and Service Offer Catalogue (Catalogue Manager)	Same procedure as the onboarding of a resource (US 1), whose parameters (fields of a form) will follow the specs found in the Information Model of Section 6.2.2.	
		Operation	Create Service method exposed by the Catalogue Manager		

US id	**User Story** (US)	**SW Module & Interface/Operation** (Interface)		**Inputs and notes**	**Outputs**
US4	*As a Resource and Service Consumer I must be able to browse all active and inactive Product Orders*	Module	Resource and Service Offer Catalogue (Catalogue Manager)	Boolean (active or not active)	List of productOrder
		Operation	*getProductOrders()*		
US 5	*As a Resource and Service Consumer I must place an order on a particular Product Offering*	Module	Resource and Service Offer Catalogue (Catalogue Manager)	productOrder (5GZORRO IM) list of DIDs with mapping to Invitation Objects for all DIDs bundled into a Product (more info in Section 4.2.4.1) List of verifiableCredentials	
		Operation	*createProductOrder()*		
US6	*As a Resource and Service Provider I must compose the offering of a Product Offering and publish it to the Catalogue*	Module	Resource and Service Offer Catalogue (Catalogue Manager)	ProductOffering (5GZORRO IM) list of DIDs with mapping to Invitation Objects for all DIDs bundled into a Product (more info in Section 4.2.4.1)	
		Operation	*publishProductOffer()*		
US 7	*As a Resource and Service Provider, I must be able to attach the license terms (pricing), agreements and SLA to a particular Product Offer*	Module	SLAs and Agreement API (provided by SC Lifecyle Manager Module - see Section 4.2.2)	When creating a ProductOffer (US6), it will only be available for consumption in the Marketplace once all required parameters are specified (business-level). For this, it is expected that the user: (i) selects the product (ii) attaches and fills the Agreement and SLA templates previously registered through Governance Portal (section3.4) (iii) update productOffer	Updated *ProductOffer*
		Operation	Retrieval of Agreement and SLAs from respective APIs (see Section 4.2.2)		

US id	User Story (US)	SW Module & Interface/Operation (Interface)		Inputs and notes	Outputs
			updateProductOffer() method from the SC Lifecycle Manager module		
US 8	*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	SC Lifecycle Manager	Use the allowed parameters of this method to filter out ProductOrders referencing a ProductOffer which has spectrum as resources	List of ProductOrders
		Operation	getProductOrders()		
US9	*As a Resource and Service Provider, I must be able to update a previously published ProductOffer*	Module	SC Lifecycle Manager	Modified *ProductOffering*	Updated *ProductOffer*
		Operation	*updateProductOffer*()		
US10	*As a Regulator, I must be able to govern (allow or reject) the orders placed by Consumers, whose Product Offering (by Provider) contains the spectrum as an asset*	Module		If a Product Offer has a resource type of *spectrum,* the acceptance of a subsequent Product Order is subject to approval from not only the Provider, but also the Regulator.	
		Operation			
US11	*As a Provider, I must be able to reject a Product Order Request*	Module	Resource and Service Offer Catalogue (Catalogue Manager)	productOrderId, rejectionReason	
		Operation	*acceptProductOrder()* *or* *rejectProductOrder()*		

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).

4.3.2 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 TMForum's set of APIs for a Telecom Marketplace, by enhancing it and extending it to a user-friendly web application.

5 Cross-domain Analytics & Intelligence for AIOps

The AIOps paradigm revolves around a concept of operational data and involves smart and efficient data collection (capture, monitoring, telemetry), governed and intelligently stored over time, and advanced data analytics (statistics, machine learning, artificial intelligence) to provide valuable insights actionable in the context of a particular use case.

AIOps products and services can be divided into various categories. Gartner's [4] three aspects of AIOps platforms are:

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

We use the term *Data Lake* to include the Data Store and the surrounding tools to perform these operations.

5.1 Baseline Data Lake Platform

Many players in the Cloud arena provide AIOps and Data Lake services, including Amazon, Google, Microsoft, Oracle, Cloudera, Zaloni, Teradata, RedHat, and others. Many of these systems are proprietary. Open Data Hub (ODH), supported by RedHat, is an open-source project that provides open-source AI tools for running large and distributed AI workloads on OpenShift Container Platform (essentially Kubernetes). The Open Data Hub project provides open-source tools for data storage, distributed AI and Machine Learning (ML) workflows and a Notebook development environment.

The 5GZORRO Data Lake will be modelled upon the components of **Open Data Hub**, inheriting all the usual services.

We envision the Data Lake to provide the following basic services.

- Data Storage; e.g., S3-compliant Object Storage such as Ceph or Minio, databases, etc
- Messaging / streaming capability; e.g., Kafka
- Metadata Services;
- Analytics tools; e.g., Spark, TensorFlow, Kubeflow
- Runtime: Docker
  - Container orchestrator: Kubernetes
  - Serverless; e.g., Knative, FaaS, Tekton
- Data Transformation services; e.g., Hadoop, Spark
- Workflow management tools, e.g., Argo
- Event mechanisms, e.g., Argo Events, Knative Events

5.1.1 Relevant Entities and Modules

We start with the following basic components to support the workflows needed for 5GZORRO.

- Runtime (Docker containers, Kubernetes)
- Messaging / streaming capability (Kafka)
- Data Store with S3-compliant interface (e.g., Ceph, Minio)

- Pipelines (Argo)
- Data Transformation services (e.g., Hadoop, Spark)

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

Apache Kafka is a distributed streaming platform for publishing and subscribing records as well as storing and processing streams of records.

Ceph is 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.

Argo Workflows is an open-source container-native workflow engine for orchestrating parallel jobs on Kubernetes.

The **Apache Hadoop** software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.

Apache Spark™ is a unified analytics engine for large-scale data processing.

As needed, additional components will be added.

Messages to Data Lake must be signed to be able to detect authenticity and corruption of data. 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.

5.1.1.1 *Running Services in the Data Lake*

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

Using the services of a Data Lake usually includes several coordinating services, which can be thought of as a pipeline. The general cycle of using a Data Lake is:

1. gather data;
2. pre-process the data;
3. register the data in a catalogue;
4. perform some analysis on the data to obtain insights;
5. perform some action based on the result of the analysis.

As a specific example:

1. gather monitoring data of a computer cluster;
2. train an anomaly detection model to recognize abnormal behaviour;
3. deploy model at run-time;
4. detect anomalies during production;
5. perform action in response to anomaly.

The portion of the pipeline of detecting anomalies during production is depicted in Figure 5-1. 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.

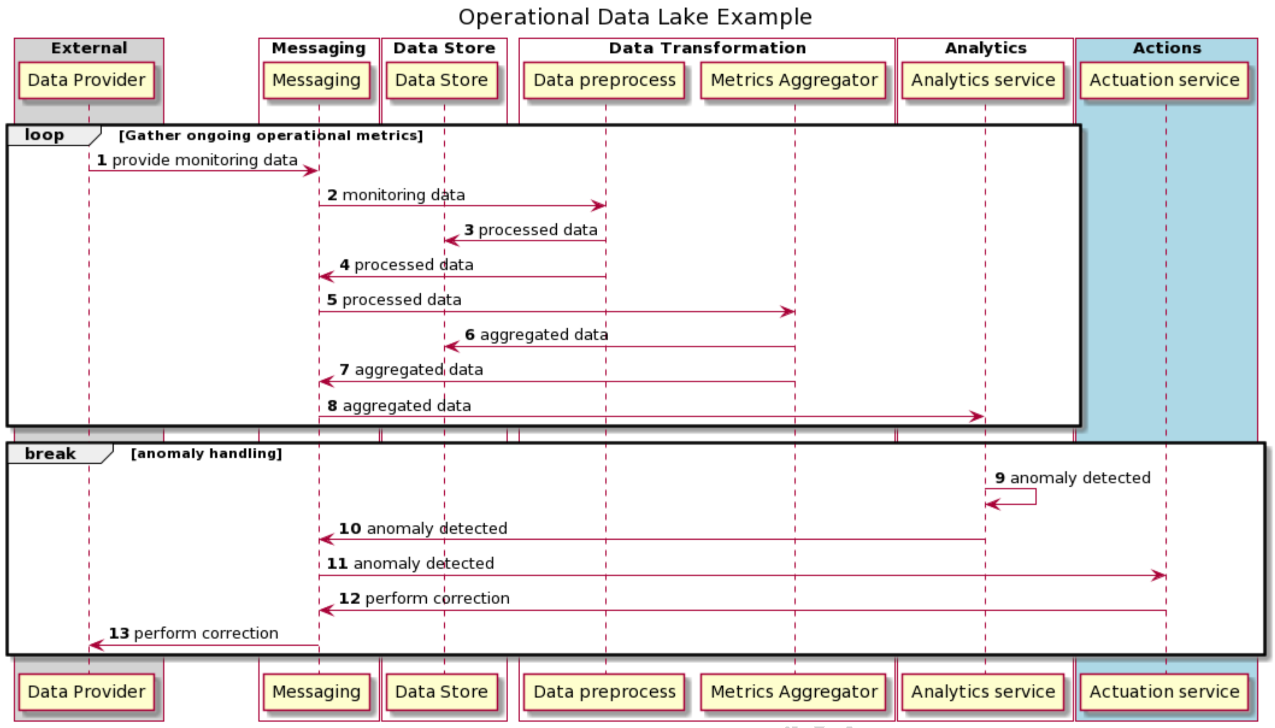


Figure 5-1: Operational Data Lake example

5.1.1.2 *Typical Data Lake workflow*

The implementation and deployment of the stages of the pipeline must be coordinated. The output from one stage often serves as the input for another stage. See Figure 5-2.

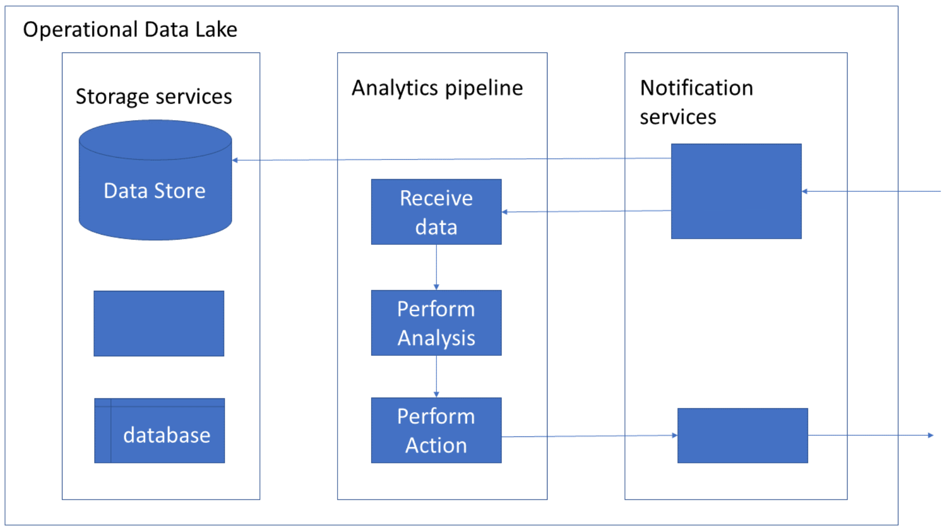


Figure 5-2: Typical Data Lake analytics pipeline setup

A Data Lake can increase its usefulness by providing services (or interfaces) to ease the deployment of interacting components in a pipeline. 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.

We propose a general Kubernetes (K8s) native framework to allow easy connectivity and coordination between services defined in a Data Lake. The design rationale is to use K8s to orchestrate services execution.

- 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 has a channel (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 as well as to a service that checks for anomalies.
- Output of data can constitute an event that will trigger another workflow (i.e., a service).
- Suppose the service wants to trigger some functionality on the client. This is encapsulated in a message delivered via the output channel (Kafka topic) of the service.

In addition to event-driven pipelines, there is also a need to support general request-response functions. This is supported using the underlying K8s mechanisms to run a long-running module that provides a REST interface to clients.

A combination of event-driven (FaaS) pipelines and general REST-interface request-response modules should be sufficient to provide efficient solutions for all the identified scenarios.

5.1.2 5GZORRO specific example

In 5GZORRO, we have the following specific example:

1. provide monitoring data;
2. data aggregator;
3. followed by some analytics to detect or predict violation of SLA;
4. followed by actions to be performed upon detection or prediction of SLA violation.

This workflow (depicted in Figure 5-3) is more complicated than the standard event-driven pipeline described above, and involves the interaction of several services. In this example, there is one workflow to collect and aggregate the resource monitoring information for each resource, and there is a separate workflow to monitor SLAs. The metrics of each resource need to be mapped to the SLAs upon which they impact. The SLA monitoring workflow then needs to obtain the relevant metrics to be able to perform its analytics to determine SLA breaches and expected breaches. This is accomplished with a combination of data-driven events (using Kafka topics) and other specialized (REST) interfaces, as needed.

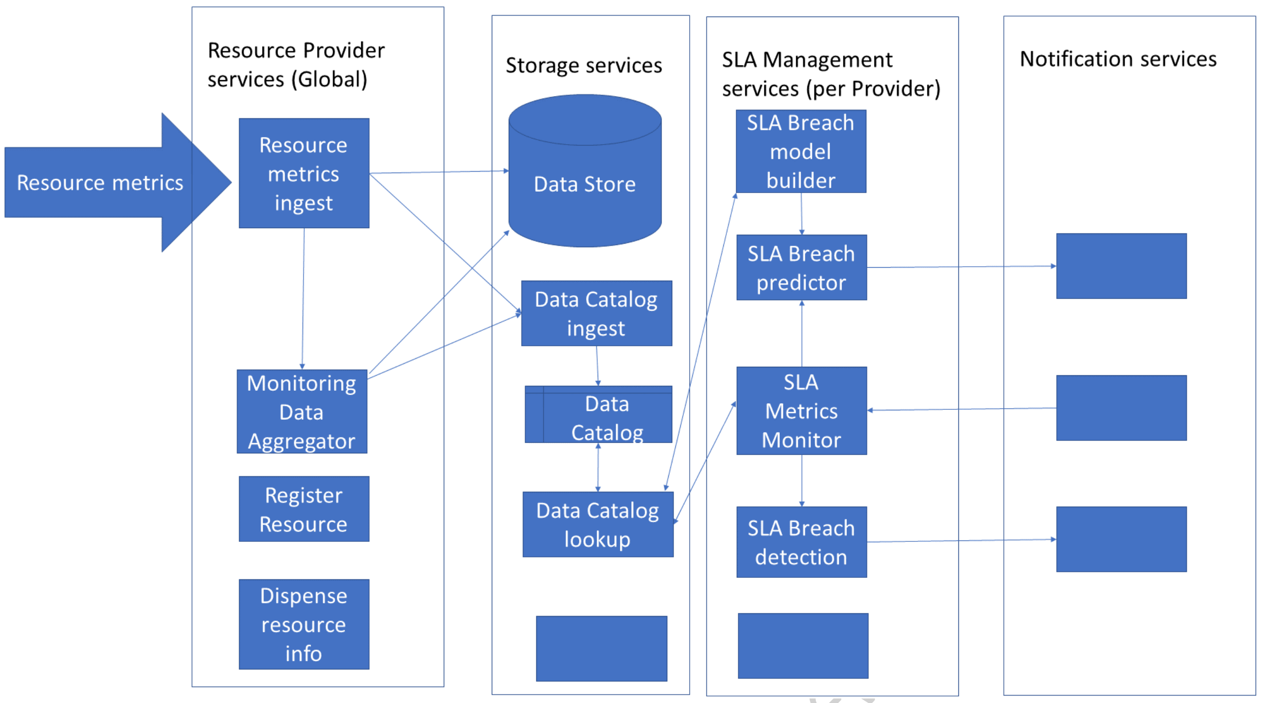


Figure 5-3: Resource Monitoring and SLA breach detection/prediction

5.1.3 Data Lake APIs

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

- Kubernetes
- S3-compatible Data Store API
- Apache Kafka API
- Argo API

In addition, the 5GZORRO Data Lake provides several APIs to facilitate the use of the Data Lake. In particular, the Data Lake provides interfaces to upload Argo pipelines with connecting input and output Kafka topics to trigger those pipelines using Function as a Service (FaaS). For other operations, the Data Lake exposes the URLs of the underlying services (Kubernetes, Kafka, Data Store, etc) to allow a user 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 Prov

iders) are examples of Operators.

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.
Notes		

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		
Notes		

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.
Notes		

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.
Notes		

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.
Notes		

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**
Notes		

5.1.4 Data Lake Information Models

5.1.4.1 *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 Operator. This includes a list of URLs to allow the Operator to access directly Kubernetes, Data Store, Kafka, etc, that are supported on the Data Lake, including 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 5-1: 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.

5.1.4.2 *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/.

5.1.4.3 *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/.

5.1.4.4 *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 5-2: 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.

5.1.4.5 *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/

5.1.4.6 *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.

5.2 Service & Resource Monitoring

Service or resource monitoring serves stakeholders to have a picture of what is the status of their own assets (network, computational, storage, etc.) with the objective of detecting possible faulty operations or overruns. In the context of multi-domain deployments, this kind of monitoring service becomes more critical but also more complex to implement.

5.2.1 5GZORRO specific enhancements

The Service & Resource Monitoring is a crucial component of 5GZORRO cross-domain functionality. On the one hand, it provides certainty to the service or resource providers that their assets are being used as agreed with the consumers. On the other hand, it also serves the resource or service consumers to be certain that the conditions signed with the provider are being fulfilled. Moreover, the Service & Resource Monitoring supports intelligent services in the 5GZORRO platform like, for instance, the Intelligent SLA Monitoring and Breach Predictor component.

In the 5GZORRO platform, the monitoring data is provided by the resource or service provider This information is pushed to the local Data Lake and copied to the cross-domain Data Lake, so the resource usage information is also available across domains. Authorised parties can obtain digested monitoring information coming from intelligent applications to, for instance, predict or 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. Every piece of information that is stored in the Data Lake is digitally signed by the source data provider. This way, the information source can be tracked and recognised as valid and trustworthy.

5.2.2 Design Details

The Service & Resource Monitoring provides two fundamental services, namely Post Monitoring Data and Aggregate Monitoring Data. The Post Monitoring Data service is called by the service or resource providers to store relevant monitoring data concerning the service, slice, and/or resource deployed into a specific data domain in the Data Lake. The data must be channelled to the Aggregate Monitoring Data module, from where the results are eventually channelled to an SLA monitoring component. See related Section 5.3.

5.2.3 Specific and relevant workflow(s)

See Figure 4-10 and Figure 5-6 for the role the Service & Resource Monitoring component plays in SLA detection and prediction.

5.2.4 APIs

Operation name: **getMonitoringKpiState**		
Description	The Service or Resource provider gets information on the status of its shared resources with the KPI values set in the SLA	
Input Parameters	**Type**	**Description**
OperatorID	string	Identity of Operator providing the resource
ResourceID	string	The ID of the resource in the 5GZORRO platform
Output Parameters	**Type**	**Description**
KPIvalues	json	List of monitoring KPIs and their values
Notes		

Operation name: **slaBreachNotification**		
Description	The Intelligent SLA monitoring and breach service sends a notification to the	
Input Parameters	**Type**	**Description**
N/A		
Output Parameters	**Type**	**Description**

	Reason	string	A description of the reason that triggered the SLA breach
	KPIvalues	json	List of monitoring KPIs and their values
Notes			

5.3 Monitoring Data Aggregator

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.

5.3.1 5GZORRO Specific and relevant workflow(s)

The main workflow we have in mind is in Figure 5-3, 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.

5.3.2 Design Details

The pipeline is broken into several components. One component simply ingests monitoring data that is provided by Operators and places the data in the Data Store. A separate component aggregates selected data over time, as needed by consumers of the data. This aggregated data is also stored in the Data Store. An interface is provided to register which data requires aggregation. In addition, a mechanism is needed to specify the list of resources whose monitoring data is required to satisfy a particular service.

A particular resource may be used for one service at one time and for another service at another time. Let's call each such time interval an 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.

5.3.3 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 5.3.4.
	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**
Notes			

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 5.3.4.
StorageLocation	url	Bucket in Data Store to place the aggregated data.
Output Parameters	**Type**	**Description**
N/A		
Notes		

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		
Notes		

Operation name: **lookupMetricsByReference**		
Description	Obtain list of aggregated metrics that pertain to specified referenceID.	
Input Parameters	**Type**	**Description**
referenceID	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 referenceID.
Notes		

5.3.4 Monitoring Data Information Models

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

Table 5-3: Resource Monitoring information

Parameter	Type	Description
resourceId	String	Unique ID of resource for which we are collecting monitoring data.
referenceId	String	Unique ID of entity (e.g., SLA) for whom 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 referenceID might be 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 referenceID, the following structure is used:

Table 5-4: 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.

5.4 Intelligent SLA Monitoring and Breach Prediction

The AIOps cycle that is followed towards the SLAs Closed-loop automation is depicted in Figure 5-4. A feedback loop starts from (1) "observation", i.e., the gathering of monitoring data about managed resources and services as provided by Virtual Resource Managers; (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.

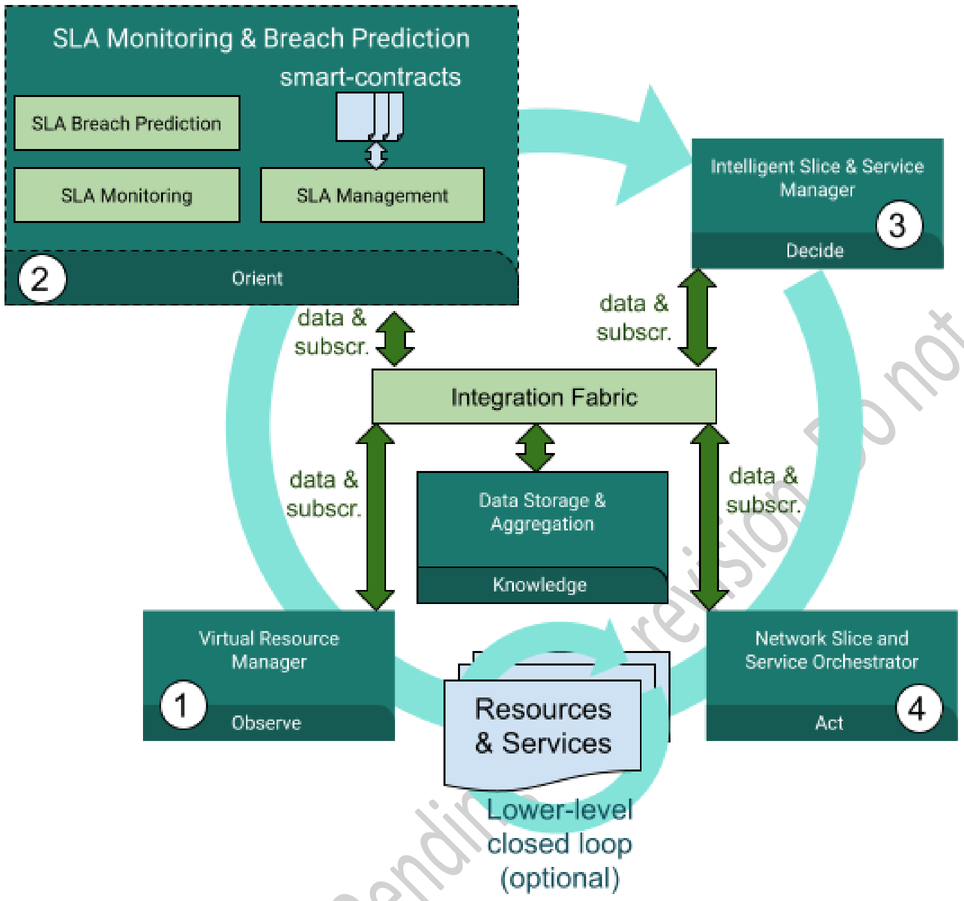


Figure 5-4: SLA Monitoring & Breach Prediction inter-communication Architecture

Delving deeper into the SLA Monitoring and Breach Prediction components, the SLA Breach Prediction collects and analyses resource monitoring data from the Data Lake and uses ML techniques in order to predict possible SLA violations. Upon SLA violations, countermeasures can be proactively taken to maintain the desired QoS for an offered service. The services that are offered by the SLA breach prediction module were described in deliverable D2.2. Each described service includes a list of interfaces in terms of input and output parameters that are explained below.

5.4.1 5GZORRO Specific enhancements

A high-level design proposed for both SLA monitoring as well as breach prediction services is shown in Figure 5-5. Currently, the design is based on the ETSI ZSM architecture and involves the Data Lake as well as the Smart Contracts Lifecycle Manager modules.

Figure 5-5 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*)

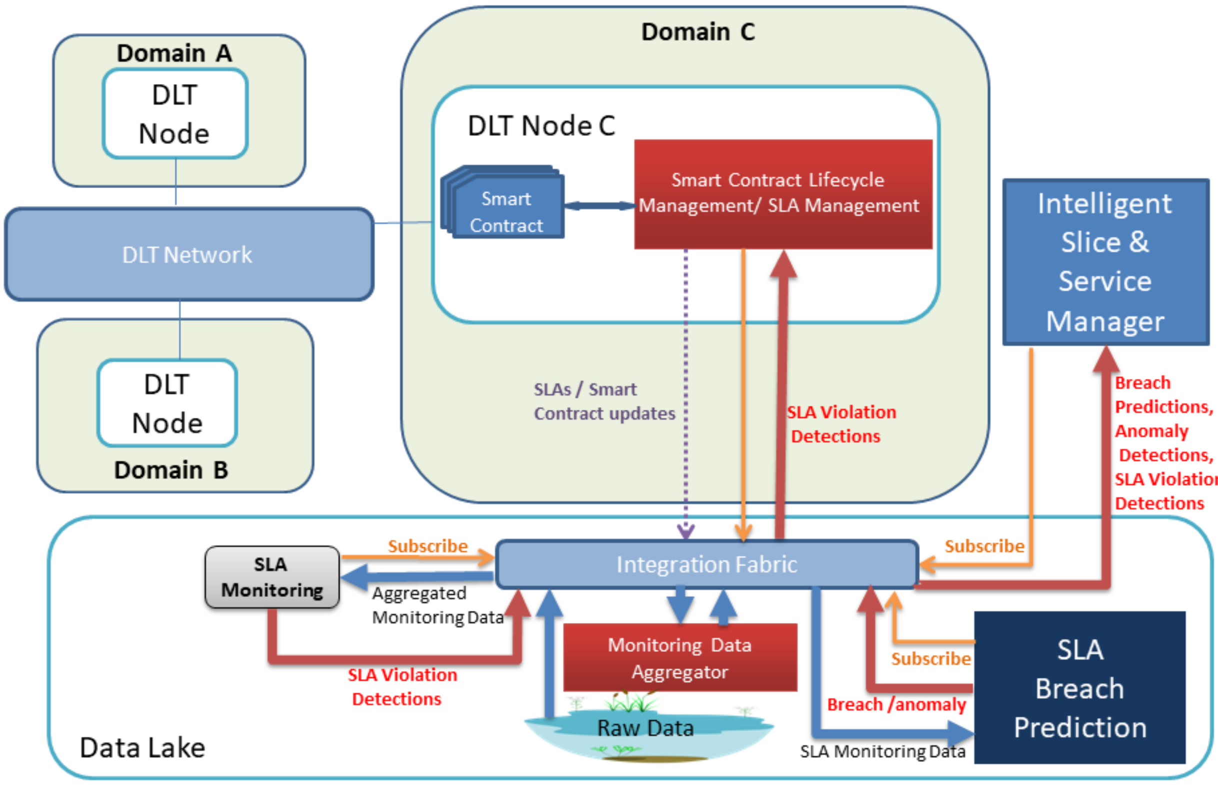


Figure 5-5 SLA monitoring and breach prediction architecture

5.4.2 Specific and relevant workflow(s)

The operational pattern that describes the sequence of actions that are linked to the Intelligent SLA Breach Prediction module is shown in the following Figure.

The workflow for SLA Breach Prediction consists of the following steps:

Step 1–2: The Resource Provider requests from the Cross-Domain Monitoring and Analytics (i.e., the Data Lake) to start the algorithms for the SLA Breach Prediction.

Steps 3–4: Monitoring data from the provider is recorded in the Data Lake, through the Service and Resource Monitoring.

Step 5: In turn, the Service and Resource Monitoring module analyses and aggregates the ingested data according to specifications defined in the Smart Contract.

Steps 6–7: Service and Resource Monitoring periodically publishes the aggregated monitoring data, which can then be retrieved by the Intelligent SLA Monitoring and Breach Prediction module in order to train the Machine Learning (ML) model.

Step 8: The ML model is executed at certain time intervals.

Steps 9–10: In case that an SLA Breach is predicted, the Resource Provider is informed accordingly and takes actions according to predefined rules.

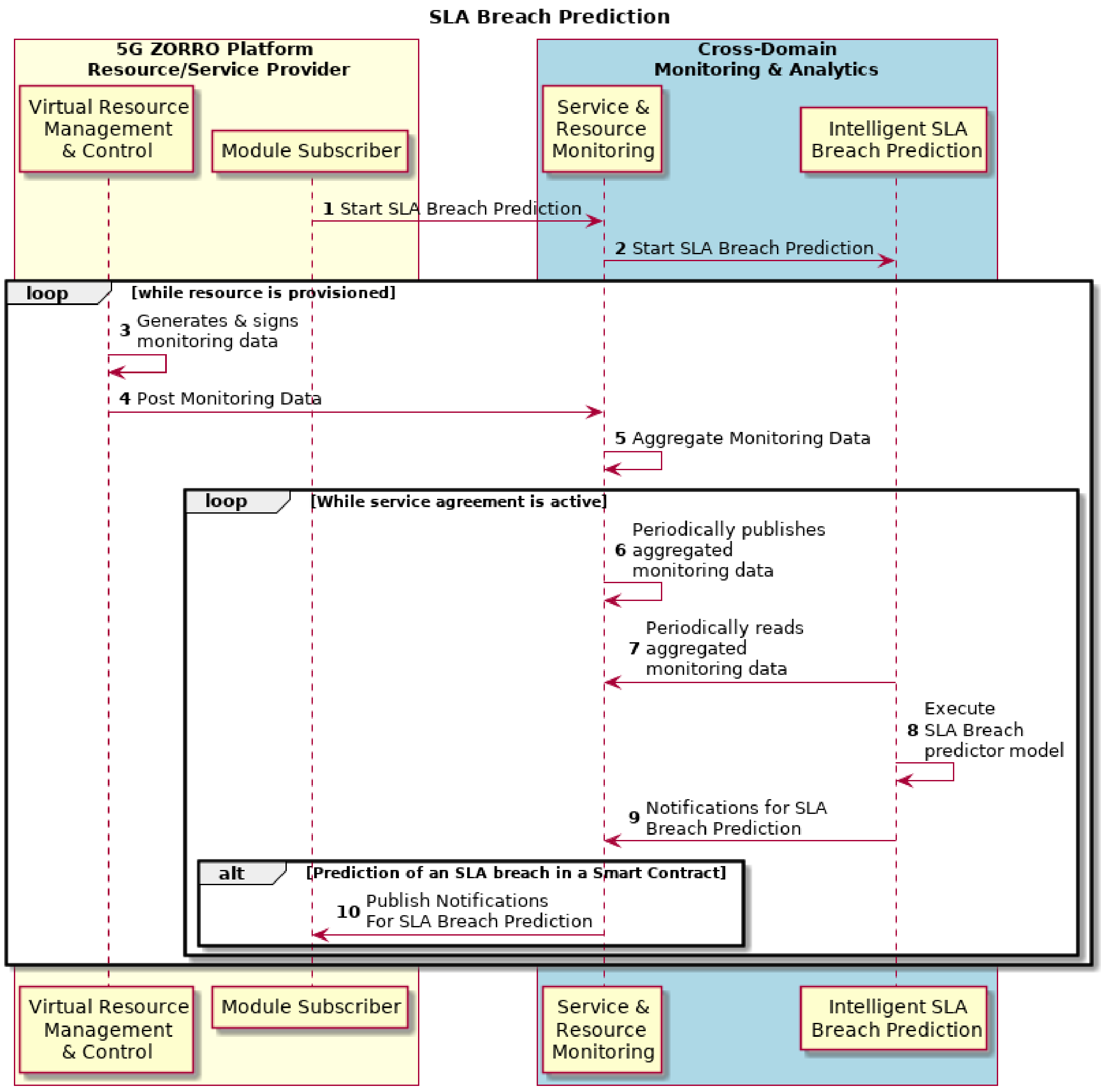

SLA Breach Prediction
5G ZORRO Platform Resource/Service Provider
Cross-Domain Monitoring & Analytics
Virtual Resource Management & Control
Module Subscriber
Service & Resource Monitoring
Intelligent SLA Breach Prediction
1 Start SLA Breach Prediction
2 Start SLA Breach Prediction
loop [while resource is provisioned]
3 Generates & signs monitoring data
4 Post Monitoring Data
5 Aggregate Monitoring Data
loop [While service agreement is active]
6 Periodically publishes aggregated monitoring data
7 Periodically reads aggregated monitoring data
8 Execute SLA Breach predictor model
9 Notifications for SLA Breach Prediction
alt [Prediction of an SLA breach in a Smart Contract]
10 Publish Notifications For SLA Breach Prediction

Figure 5-6: SLA Breach Prediction workflow

5.4.3 APIs

Table 5-5: Definition of Start SLA Breach service interface

Service name**: SLA Breach Prediction**		Type**: *Cross-domain***
Capabilities	**Support (O|M)**	**Description**
Start SLA Breach Prediction	**M**	The service starts to receive and analyse resources 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. SLA Breach Prediction process could be started by the marketplace after a new contractual agreement or by the *Intelligent Network Slice and Service Orchestration* service in order to estimate the future needs for resources.
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.
Subscribe to SLA Breach Prediction	**M**	This capability is used when a module should receive SLA Breach Predictions for specific SLAs/contracts.
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.
Unsubscribe from SLA Breach Prediction	**M**	This capability is used to stop receiving notifications for SLA breach predictions for specific SLAs/contracts.
Configure Machine Learning Algorithms	**M**	This capability is used to select and configure the machine learning algorithms based on the types of SLA metrics to be monitored and the associated requirements.
Notes		
none		

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.
Notes		

Operation name: **updateSLABreachPrediction**		
Description	This operation 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.	
Input Parameters	**Type**	**Description**
SLABreachPredictionID	String	The Identifier of the SLA Breach Prediction process that will be modified.
ContractIDs	String	The identifiers of the contracts for which predictions will be made for violations of their SLAs.
SLAMetrics	String	List of specific SLA for the case that the predictions will not be related to all SLAs of the contracts.
StartTime	DateTime	The SLA Breach Prediction could start immediate 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**
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.
Notes		

Operation name: **terminateSLABreachPrediction**		
Description	This operation is used to stop receiving and analysing monitoring data for specific contract SLAs.	
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.
Notes		

Operation name: **getActiveSLABreachPredictions**		
Description	This operation is used to create a list of active SLA Breach Predictions, their descriptions, their breach predictions and their subscribers.	
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.
Notes		

Operation name: **subscribeToSLABreachPrediction**		
Description	This operation is used when a module should receive SLA Breach Predictions for specific SLAs/contracts.	
Input Parameters	**Type**	**Description**
SubscriberID	String	The unique identifier of the module that will become a subscriber.
SLABreachPredictionID	String	The Identifier of the SLA Breach Prediction process in which the module will become a subscriber.
ContractIDs	String	The identifiers of the contracts for which the module will monitor SLAs breach predictions.
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 (e.g., subscription denied for security reasons) it returns the Identifier of the error message.
ErrorDescription	String	Optionally it returns specific details for the error.
Notes		

Operation name: **publishSLA Breach PredictionsNotifications**		
Description	The module sends to the subscribed modules predictions for SLA violations. 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
Notes		

Operation name: **unsubscribeFromSLABreachPrediction**		
Description	This operation is used to stop receiving notifications for SLA breach predictions.	
Input Parameters	**Type**	**Description**
SLABreachPredictionID	String	The Identifier of the SLA Breach Prediction process from which the module will unsubscribe.
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.
Notes		

Operation name: **configureMachineLearningAlgorithms**		
Description	This operation allows to select and configure the machine learning algorithms based on the types of SLA metrics to be monitored and the associated requirements.	
Input Parameters	**Type**	**Description**
ConfigurationParameters	json	A list of the algorithm specific configuration parameters that are used for prediction of SLA violations (i.e., unsupervised, supervised).
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.
Notes		

5.5 Smart Resource and Service Discovery application

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

As previously stated, stakeholders acting as assets consumers interact with the Marketplace, and in particular with the services provided by the 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.

5.5.1 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:

- **Marketplace Catalogue – Automated Discovery:** 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.
- **Smart Discovery Application – Intent-based Discovery:** based on (high-level) intent-based priorities (e.g., performance, proximity, cost, slice segment, and trade-offs). Response provides the most suitable offers according to the particular priority the consumer of the application has requested.

The architecture of the Smart Resource and Service Discovery application module is illustrated in Figure 5-7. In this architecture the request for this module is originating from the Intelligent Slice and Service Manager (ISSM) that is introduced in 5GZORRO Deliverable 4.1 150[3]. As a result of the request the Smart Resource and Service Discovery returns to ISSM a list of resource offers. The method that is used to retrieve these offers is described in the following part of this section.

In addition, to respond 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

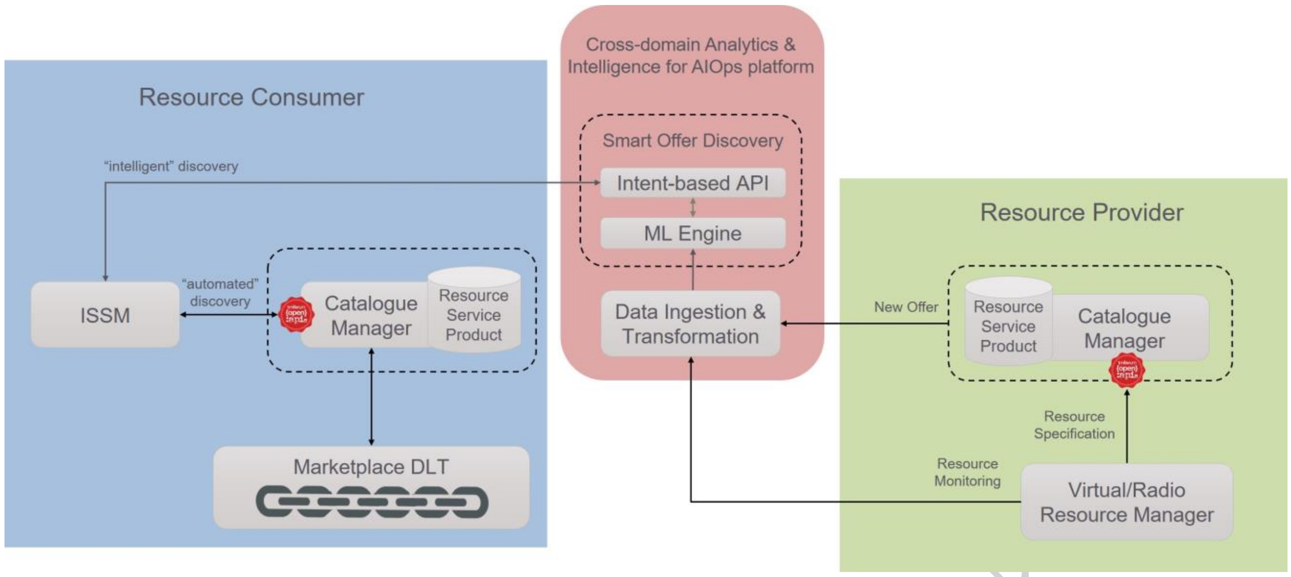


Figure 5-7: Smart Resource and Service Discovery application architecture

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 [8], 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 Catalogue), just a prescription is given, which in this case may correspond to a slice segment and/or to an aspect to be prioritized. 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 resource offers. The topics that are relevant to group resources with similar features are linked to type (compute, storage, network, RAN); format (physical, virtual); slice segment (core, RAN, transport, edge); location; offered price and capabilities; among others.

5.5.2 Specific and relevant workflows

The workflow associated with the services offered by the Smart Resource and Service Discovery application is shown in Figure 5-8.

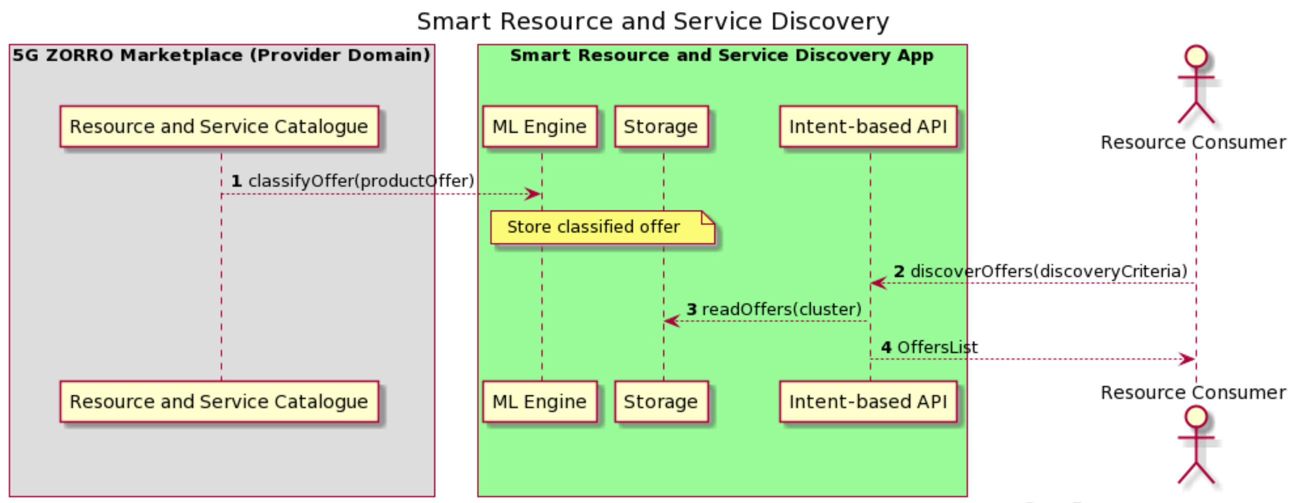

Smart Resource and Service Discovery
5G ZORRO Marketplace (Provider Domain)
Smart Resource and Service Discovery App
Resource and Service Catalogue
ML Engine
Storage
Intent-based API
Resource Consumer
1 classifyOffer(productOffer)
Store classified offer
2 discoverOffers(discoveryCriteria)
3 readOffers(cluster)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

Figure 5-8: 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 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 assessment).

5.5.3 APIs

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

Operation name: **classifyOffer**			
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 6.3.1)
Output Parameters		**Type**	**Description**
	-	-	-
Notes			

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
	-	-	-
Notes			

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. (see Section 6.3.1)
Notes			

Operation name: **clusteringSelection**			
Description		Endpoint for selecting the features that will be enabled in the Off-line clustering algorithm.	
Input Parameters		**Type**	**Description**
	Feature List	List	The default list of features used for the clustering algorithm training. This list can be updated from the API and impacts the retrieved List of Offers.
Output Parameters		**Type**	**Description**
	-	-	-
Notes			

6 Information Elements

6.1 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, 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, DIDs associate a *DID Subject*, the DID's owner, with a DID Document, the important metadata related to a decentralised identifier that can be used to authenticate it or to verify its relationship with the DID. Among metadata registered in a DID Document, 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). 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 assuming 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 the Verifiable Registry (i.e., 5GZORRO Governance DLT) 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.

6.1.1 High Level View

6.1.1.1 *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 Platform	No	The software system implementing 5GZORRO functionalities e.g., the Marketplace node, that is deployed and operated by a 5GZORRO stakeholder.
5GZORRO Resource	No	The available resource or set of resources managed by a Resource Provider through the 5GZORRO Framework which is/are deployed and enabled to be traded in the Marketplace.
5GZORRO Service	No	A 5G Service or set of 5G Services ready for utilizing in the Marketplace and which is/are delivered by a 5G Service Provider.
5GZORRO Product	No	A 5G Product offer comprised by 5GZORRO resources and / or services, that is published in 5GZORRO Catalogue.
5GZORRO Business Agreement	No	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.
5GZORRO Marketplace Governance Board	Yes	This entity represents the governance board of 5GZORRO Marketplace where all Stakeholders with permissions to take governance decisions are included. All Governance actions (including voting) and decisions are associated to the Marketplace Governance Board.

6.1.1.2 *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 below.

Table 6-1: 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
platformVC	1.1	Claims about the different 5GZORRO operational software packages including its endpoints.	5GZORRO Platform	DID User-Agent	Marketplace Governance
governanceActionVC	1.1, 1.7	Claim about a certain governance action performed by a Marketplace Admin e.g., vote for decision			
governanceDecisionVC	1.1, 1.7	Claim about a certain governance decision including. ▪ to accept or reject a new 5GZORRO member ▪ resolutions about SLAs disputes	Admin DID Agent Trading DID Agent	Admin DID Agent	Admin DID Agent Trading DID Agent
resourceVC	1.2	Claim about rights over a certain 5GZORRO Resource or set of 5GZORRO Resources. Claim that Resource is deployed and ready to be consumed by consumers and a certain behaviour is expected. Claim that data generated by the Resource namely monitoring data, is trustworthy	Trading Provider DID Agent	Admin DID Agent	Trader Consumer DID Agent

serviceVC	3.1, 3.2, 3.3	Claim that service is deployed and ready to be consumed by service consumers and a certain behaviour is expected. Claim that service can be extended to other domains, by defining descriptors to be used and VNF registries endpoints to be used from where images can be downloaded and deployed in the new domain.	Trading Provider DID Agent	Admin DID Agent	Trading Consumer DID Agent
agreementVC		Claim that identify 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

6.1.2 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.

6.1.2.1 *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 context of the VC, the id is an identifier that others must use in order to express statements covered by this verifiable credential, the type means the kind of information represented in the VC, the credentialSubject represent 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, the expirationDate depicts when VC will not be considered useful, the credentialStatus is utilized for knowing information about the current status of a VC and finally, the proof contains all details necessary to assess this VC is authentic.

More details are provided in the table below:

Parameter	**Type**	**Description**
id	URI	Identifier that others must use in order to express statements covered by this verifiable credential
type	String	Kind of information represented in the VC. The following 5GZORRO VC types are considered: "StakeholderCredential","ResourceCredential", "ServiceCredential", "ProductCredential", "AgreementCredential", "GovernanceBoardCredential".

credentialSubject		
id	DID	The Subject DID
claims	List of Claims	Set of claims about the Verifiable Credential Subject. See Subject Claims Information Model below
issuer	List of objects	Person or entity that allocates this VC
id	DID	Identifier of the Issuer
name	String	Issuer name
issuanceDate	String (of an [*RFC3339*])	Indicates when VC was emitted and validated
expirationDate	String (of an [*RFC3339*])	Depicts when VC will not be considered useful
credentialStatus	List of objects	Utilized for knowing information about the current status of a VC
id	String	Identifier of the Credential Status used, must be a URL
type	String	CredentialStatus type that should provide enough information to determine the current status of the credential including "active", "suspended", "revoked"
proof	List of Proof objects	Contains all details necessary to assess whether this VC is authentic. See Proof Information Model below

6.1.2.2 *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
resourceClaim	ResourceClaim Object	A Claim object to be used in Credentials of ResourceCredential type. Compliant with Catalogue ResourceCandidate Information Model
serviceClaim	ServiceClaim Object	A Claim object to be used in Credentials of ServiceCredential type. Compliant with Catalogue ServiceCandidate Information Model
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 ProductCandidate Information Model
governanceClaim	GovernanceClaim Object	A Claim object to be used in Credentials of GovernanceCredential type
platformClaim	PlatformClaim Object	A Claim object to be used in Credentials of StakeholderCredential type

Stakeholder Claim Object

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

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, …)

Agreement Claim Object

Parameter	**Type**	**Description**
stakeholderDID	DID	This is the stakeholder DID
stakeholderRole	String	Role played by the stakeholder
productDID	DID	Product DID being traded
authorityService	String	Entity that performs the SLA data monitoring aggregation and executes the violation check/analysis.

Governance Claim Object

Parameter	**Type**	**Description**
actionType	String	Type of Governance Actions including "GovernanceVote", "GovernanceMembershipDecision", "GovernanceSlaViolationDecision"
actionTarget	DID	The DID of the Subject on which the decision is taken
actionExecutor	DID	The DID of the Board member that is taking the action
actionDate	String (of RFC3339)	When the action was taken
actionValue	String	The value of the action that depends on the action type
description*	String	A human readable description about the action

Platform Claim Object

Parameter	**Type**	**Description**
platformType	String	Type of 5GZORRO platforms including "Cross-domain AIOps", "Marketplace", "Governance", "ZSM&O"
deploymentDate	String (of RFC3339)	When platform was deployed
version	String	Version of the deployed platform
deploymentStatus	String	Information about the platform availability, including unavailable, available, outstage, disruption

endpoints	List of Objects	list of key-value Objects where key is the service name and value the service endpoint url

6.1.2.3 *Verifiable Credential Presentation Information Model*

Verifiable Credential Presentation is Data derived from one or more verifiable credentials, issued by one or more issuers, that is shared with a specific verifier. A verifiable presentation is a tamper-evident presentation encoded in such a way that authorship of the data can be trusted after a process of cryptographic verification. Certain types of verifiable presentations might contain data that is synthesized from, but do not contain, the original verifiable credentials (for example, zero-knowledge proofs).

The Verifiable Credential Presentation parameters are detailed in the table below:

Parameter	**Type**	**Description**
id*	String	Provide a unique identifier for the presentation
type	List of objects	Is required and expresses the type of presentation, such as VerifiablePresentation.
verifiableCredential*	List of objects	Must be constructed from one or more verifiable credentials, or of data derived from verifiable credentials in a cryptographically verifiable format.
holder*	DID	Is expected to be a DID for the entity that is generating the presentation.
proof*	List of Proof objects	The value of the proof property ensures that the presentation is verifiable

6.1.2.4 *Verifiable Credential Proof Information Model*

Contains all details necessary to assess whether this VC is authentic. The set of attributes to be used in a proof will vary according to the representation language and the technology used.

Parameter	**Type**	**Description**	**Example**
type	String	Expresses the cryptographic signature suite that was used to generate the signature	"RsaSignature2018"
created	String (of an [*RFC3339*])	Indicates when proof was created	"2021-06-18T21:19:10Z"
proofPurpose	String	Expresses the purpose for the proof and ensures this information is protected by the signature	"assertionMethod"
verificationMethod	String	The verificationMethod property specifies the public key that can be used to verify the digital signature	"https://example.com/jdoe/keys/1"

jws	String	If a JWS is present, the digital signature either refers to the issuer of the verifiable credential, or in the case of a verifiable presentation, the holder of the verifiable credential.	"eyJhbGc..DJBMvv"

6.2 Resource and service modelling

Heterogeneity of resources and incorporation of edge resources to the available infrastructure mandate the appropriate modelling of resources. Modelling should be in such an abstracted level that fully describes the resources' capabilities, but at the same time allows for their homogeneous management. In this section, the proposed resource and service model is analysed in the form of class diagrams. This approach serves for maintaining resource information (including usage) at different levels in our system as well as for exchanging information about resources between different system components.

We assume that there are two general types of offers that can be supported by the Marketplace, namely the Resource offers and the Service offers, similarly to the TM Forum ZOOM Information Model [5].

6.2.1 Resource Model

To begin with the Resource Model, a high-level representation of its class diagram is shown in Figure 6-1. The Resource is defined as an abstract entity of interest to the managed environment that may or may not be manageable by electronic means. It may or may not form part of a Product offer [5]. According to the location of a resource, it can be characterised as an edge or as a core resource. The edge of the network is the part that's closer to the end user, while the core is the centralized hub of the network that processes data. Thus, the edge resources usually reside at the endpoints and at the first hop from the endpoints into the core of the network. For example, in enterprise, the endpoints are Personal Computers (PCs), including their associated adapter, and modems for connecting to carriers, and various connected devices. Other edge resources are WiFi access points as well as desktop and wiring closet switches.

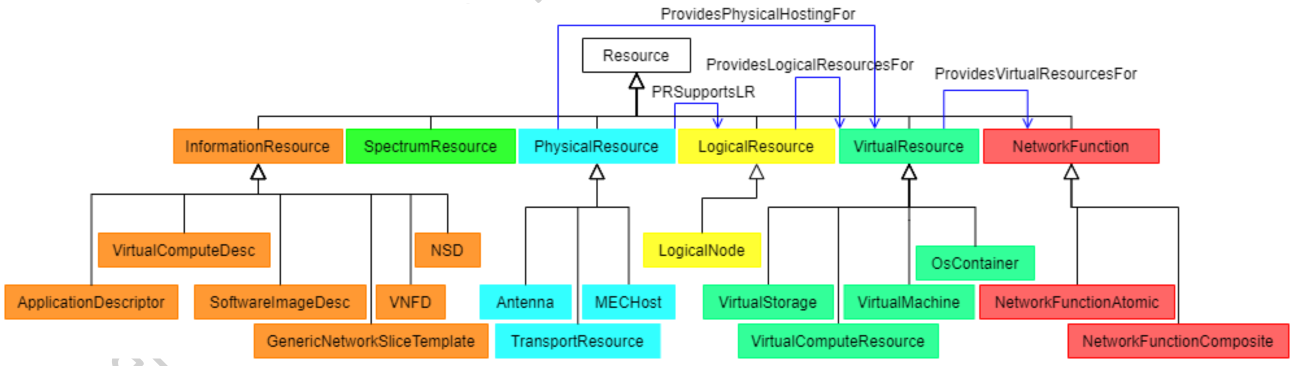


Figure 6-1: Resource Model Class Diagram

Table 6-2 depicts the Information Model of a Resource candidate. The last column shows an example of a Resource offer, by giving values to the respective fields.

Table 6-2: Resource candidate Information Model [9]

Parameter	Type	Description	Example
id	String	Unique identifier for the resource in the catalogue	"7479"

didRef	DID	Distributed identifier of the resource	
href	String	Hyperlink reference to this resource	"https://host:port/catalogManagement/resourceOffer/7479"
name	String	Name of the resource	"Virtual Storage Medium"
description	String	Description of this resource	"This resource offer ..."
type	String	Class type of this resource	"ResourceOffer"
schemaLocation	String	This field provides a link to the schema describing this resource	https://host:port/catalogManagement/schema/ResourceOffer.yml
baseType	String	The (immediate) base class type of this resource	
version	String	The version of resource offer	"3.2"
validFor	TimePeriod	The period for which this resource is valid	"startDateTime": "2020-08-12T00:00", "endDateTime": "2021-03-07T00:00"
lastUpdate	DateTime	Date and time of the last update of this resource	"2020-08-09T00:00"
lifecycleStatus	String	Used to indicate the current lifecycle status of the resource offer	"Active"
category	A list of category references (ResourceCategoryRef)(Table 6-3)	The category resource is used to group resource offerings in logical containers. Categories can contain other categories and/or resource offerings.	See Table 6-3
resourceSpecification	ResourceSpecificationRef (Table 6-4)	A resource specification reference	See Table 6-4
resourceOwnerDID	DID	Distributed Identifier of the resource owner	
resourcePhysicalCapabilities	List of objects	A list of operation band values	See Table 6-6
resourceVirtualCapabilities	List of objects	A list of operation band values	See Table 6-7

Table 6-3: ResourceCategoryRef Information Model [9]

Parameter	Type	Description	Example
id	String	Unique identifier of the category	"5355"
href	String	Reference of the category	"https://host:port/catalogManagement/category/5355"
version	String	Category version	"1.1"
name	String	Name of the category	"Cloud Resources"
type	String	Class type of this resource	"ResourceCategory"

Table 6-4: ResourceSpecificationRef Information Model [9]

Parameter	Type	Description	Example
id	String	Unique identifier of the resource specification	"42"
href	String	Reference of the resource specification	"http://hostname:port/catalogManagement/resourceSpecification/42"
version	String	Resource specification version	"3.2"
name	String	Name of the required ResourceSpecification	"VirtualStorageMedium"
type	String	Class type of this resource	"VirtualDeviceSpec"

Another class relevant to the Information Model of Resource, is the ResourceSpecification class (Table 6-5), where the resource specification is held.

Table 6-5: ResourceSpecification Information model

Parameter	Type	Description
href	DID	Distributed identifier of the resource specification
name	String	The name of the resource specification
category	String	Category of the target resource.
resourceSpecCharacteristic	List of ResourceSpecCharacteristic [9]	A list of resource spec characteristics (ResourceSpecCharacteristic [*]). This class defines the characteristic features of a resource specification. (see Table 6-6 and Table 6-7)

ResourceSpecCharacteristic list has two types of elements, resourcePhysicalCapabilities and resourceVirtualCapabilities, analysed in Tables Table 6-6 and Table 6-7, respectively. This level of description allows to model the virtual resources and the hardware capabilities that can be exposed. With different combinations of two simple models, it is possible to represent: A full cluster offering with any PaaS capability, simple virtualized instance offers, physical storage node offers or full bare metal offers.

Table 6-6: resourcePhysicalCapabilities Information Model

Parameter	Type	Description	Example
name	String	Name of the Physical Capability	"hardwareCapabilities"
description	String	Description of the capability	"Physical capabilities offered in the resource "
cloudId	String	Unique identifier of the cloud	"cloudId1"
datacenterId	String	Unique identifier of the datacenter	"datacenter1"
nodeId	String	Unique identifier of the node	"node1"
hardwareCapabilities	List of objects	Descriptor of HW capabilities (GPU, SDD storage, FPGA, ASIC, NIC, high performance network uplinks, etc.)	
hardwareCapKey	String	Type of capability	storage
hardwareCapValue	Float	Value of capability	200
hardwareCapUnit	String	Unit of measure (Gb, MHz, etc.)	GB
hardwareQuota	Float	Quota of offered resource	
feature	List of objects	Describes the PaaS of the capability	
type	String	Openstack/K8s/Openshift etc	
href	String	Hyperlink reference to the target specification	
isBundle	String	A flag indicating if the node is master (true) or not (false)	

Table 6-7: resourceVirtualCapabilities Information Model

Parameter	Type	Description	Example
name	String	Name of the capability	"virtualCapabilities"
description	String	Capability description	"Virtualized capabilities offered in the resource"

cloudId	String	Unique identifier of the cloud	"cloudId1"
datacenterId	String	Unique identifier of the datacenter	"datacenter2"
nodeId	String	Unique identifier of the node	"node3"
isMaster	Boolean	Identifies the master or worker node	
type	String	Type of the Resource ("openstack", "kubernetes", "openshift")	
href	String	Hyperlink reference to the target specification	
virtualCapabilities	List of Objects	A list of objects that represents the virtual capabilities	
*VirtualCap**ValueType***	String	CPU, storage, RAM, etc	
*VirtualCap**Value***	Object	Values of the value type	
*VirtualCap**Unit***	String	Unit of measure: Gb, MHz, SO version, etc	

A resource can belong to any of the following subtypes (also depicted in Figure 6-1.):

- InformationResource,
- SpectrumResource,
- PhysicalResource,
- LogicalResource,
- VirtualResource,
- NetworkFunction.

The InformationResource is basically a holder of information [5]. Such information may be an Application Descriptor, a Virtual Compute Descriptor, a Software Image Descriptor, a Network Slice Template, a VNF Descriptor (VNFD) or a Network Service Descriptor (NSD), as represented by the respective subclasses in Figure 6-1. An Application Descriptor (ApplicationDescriptor class) is a part of an application package and it is usually provided by the application provider. The ApplicationDescriptor expresses the application requirements and rules of an application [6]. In regard with the Virtual Compute Descriptor (VirtualComputeDesc class), it supports specification of requirements related to virtual compute resources, such as virtual memory, CPU and disk requirements. For example, the virtual compute resources defined by such a descriptor could be used by a VNF when each of the VNF Component (VNFC) instances of the VNF is intended to be deployed in a single VM [7]. Similarly, to VirtualComputeDesc, the Software Image Descriptor (SoftwareImageDesc class) defines descriptors of software images to be used, for instance by a VNF for a VM-based VDU, an Operating System (OS) Container or a virtual storage resource [7]. A Generic Network Slice Template (GST) (GenericNetworkSliceTemplate class) contains a set of attributes that can characterize a type of network slice. One or more Network Slice Instances (NSIs) can be created out of the same type based on the requirements. GST is generic and is not tied to any specific network deployment [10]. The information model of the Network Slice is explicitly detailed in Section 8.5. The VNFD [7] and the NSD classes describe specific deployment versions of a VNF and an NS, while the information models for VNFs and NSs are presented in Sections 8.4 and 8.5, respectively.

The SpectrumResource Information Model is analysed in Section 8.1.

The PhysicalResource is an abstract base class defining different types of hardware that may or may not be packaged as part of a Product. By definition, a PhysicalResource is a representation of a physical object, such as Chassis, Rack, Cable Duct, etc. Generally speaking, hardware resources do not have the ability to communicate by electronic means. However, they can be used to hold Logical Resources (LogicalResource class) and/or Virtual Resources (VirtualResource class) that communicate with other entities. In other words, a PhysicalResource provides a means for containing a LogicalResource that does give status and other information that is associated with the PhysicalResource. For example, a Rack could have a door mounted with an electronic sensor that detects when it is open or closed [5]. The Antenna, the MEC Host (MECHost class) and the Transport Resource (TransportResource class) are subclasses of the PhysicalResource. The complete class diagram of the PhysicalResource is illustrated in Figure 6-2.

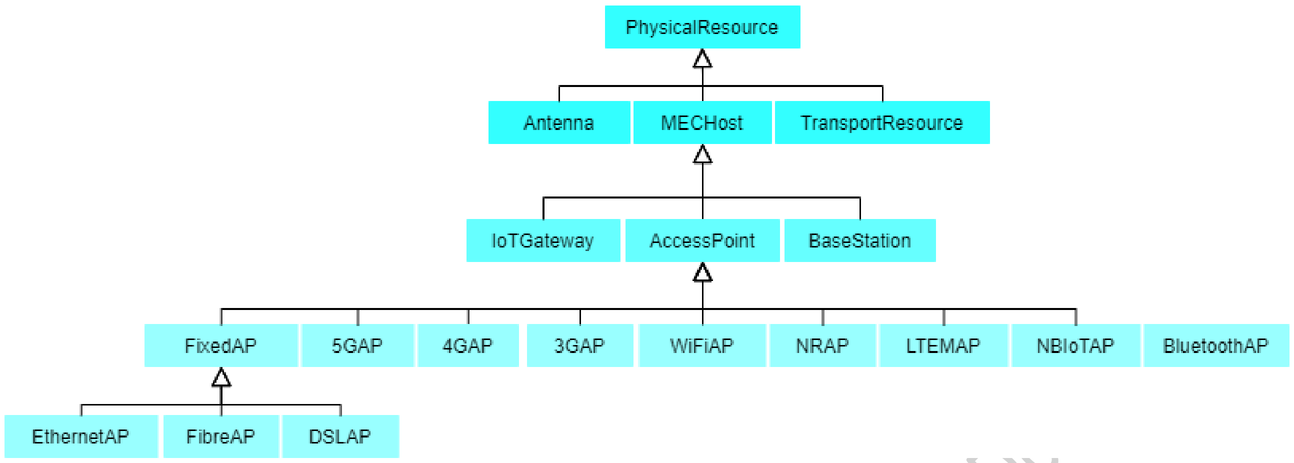


Figure 6-2: PhysicalResource Class Diagram

MECHost describers a MEC-enabled networking edge node. MEC hosts are located at the network edge and simultaneously provide networking, storage and computational services, such as Internet access, video content delivery, caching etc. Each edge node is equipped with a set of networking, storage and computational capabilities, usually measured in terms of number of Resource Blocks (RBs), megabytes, and billions of instructions per second (GIPS), respectively [11]. Examples of MEC-enabled networking edge nodes are the Internet of Things (IoT) Gateways, the Access Points (APs) and the Base Stations. These are shown as subclasses of MECHost in Figure 6-2. Focusing on Access Points (AccessPoint class), this refers to the access technologies that can be supported, for instance, by a Network Slice. Some examples are the access points for fixed access (such as Ethernet, Fibre and DSL APs), 3G, 4G, 5G, Wi-Fi, New Radio (NR), Long Term Evolution for Machines (LTE-M), Narrowband IoT (NB-IoT) and Bluetooth [10].

The LogicalResource is an abstract class for representing resources that are of interest to the managed environment and are not virtualized. Basically, it describes logical aspects of devices (e.g., services). A LogicalResource can (but does not have to) be packaged as part of a Product [5]. A LogicalNode is a subclass of LogicalResource and describes compute, memory and input/output (I/O) requirements that are to be associated with the logical node of infrastructure. The logical node requirements are a subcomponent of the Virtual Deployment Unit (VDU) level requirements [7].

A VirtualResource is the result of applying one or more Software Processes to a Resource to create one or more new Resources whose assets (e.g., memory) and services (e.g., networking) can be shared among multiple, possibly heterogenous, consumers. In other words, a VirtualResource is an abstraction that decouples physical manifestation & delivery of a Resource from its logical operation. In particular, it could be used in conjunction with a virtualization process to create virtual compute, virtual network and virtual storage. A virtualized CPU is an example of VirtualResource. As shown in Figure 6-1, a VirtualResource must be hosted by a PhysicalResource and can (but does not have to) be packaged as part of a Product [5]. Figure 6-3 further analyses the subclasses of Virtual Resource.

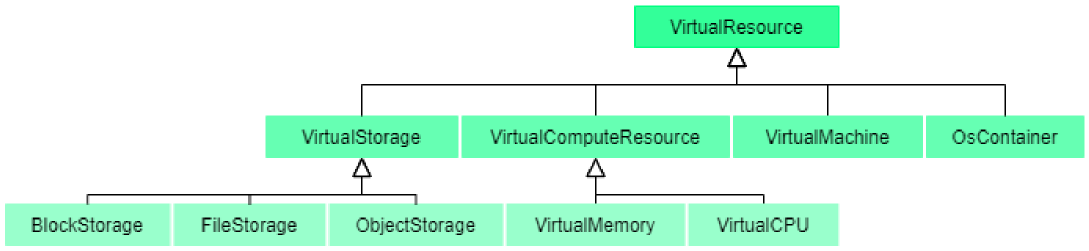


Figure 6-3: VirtualResource Class Diagram

The VirtualStorage class specifies the requirements related to persistent virtual storage resources. The BlockStorage, FileStorage and ObjectStorage classes specify the details of a block storage resource, a file storage resource and an object storage resource, respectively [7]. Ephemeral virtual storage is specified in VirtualComputeResource class [7]. The VirtualComputeResource supports the specification of requirements related to virtual compute resources [7]. A Virtual Compute Resource can be characterised by a Virtual Memory (VirtualMemory class) and a Virtual CPU (VirtualCPU class). The VirtualMachine class describes the properties of a VM, while an OsContainer class describes the members properties of a set of co-located container compute resources when these are realizing a VDU [7].

The NetworkFunction class represents the concept of a Functional Block. It can represent either stand-alone (NetworkFunctionAtomic instances) and/or composite (NetworkFunctionComposite instances) objects. It is related to other model elements (specifically with the LogicalResource, VirtualResource, and InformationResource classes). A NetworkFunction is an element within a network with well-defined external interfaces and functional behaviour, such as a DHCP, a Firewall, a SAE Gateway, an MME or an eNodeB [5].

6.2.2 Service Model

A high-level representation of the Service model class diagram is shown in Figure 6-4, while its Information Model is analysed in Table 6-8.

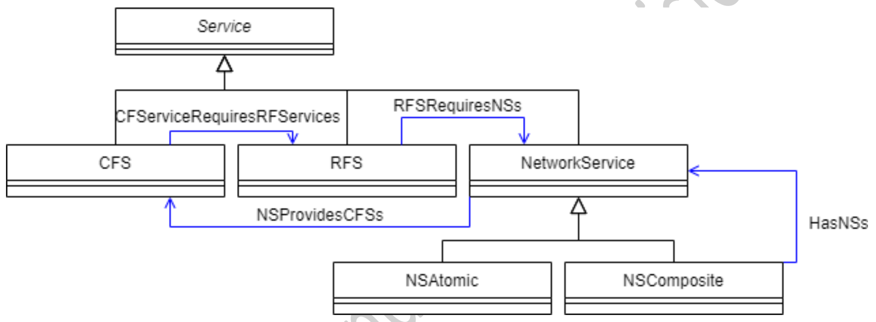


Figure 6-4: Service Model Class Diagram

Table 6-8: Service candidate Information Model [12]

Parameter	Type	Description	Example
id	String	Unique identifier	"4994"
didRef	DID	Distributed identifier of the Service	
category	List of service category references (ServiceCategoryRef)	List of categories for this service offer.	See Table 6-9
href	String	Hyperlink reference to this service	"https://hostname:port/serviceCatalogManagement/v3/serviceOffer/4994"
description	String	Description of this service	"This service offer allows provision of TV service"
lastUpdate	DateTime	Date and time of the last update of this service	"2020-08-27T00:00"
lifecycleStatus	String	Used to indicate the current lifecycle status of the service	"Active"
name	String	Name of the service	"TVServiceOffer"
serviceSpecification	A service specification reference (ServiceSpecificationRef)	The service specification implied by this offer	See Table 6-10
version	String	The version of service	"2.1"

validFor	TimePeriod	The period for which this service offer is valid	"startDateTime": "2020-08-23T00:00", "endDateTime": "2021-03-25T00:00"
baseType	String	When sub-classing, this defines the super-class	
schemaLocation	Uri	A URI to a JSON-Schema file that defines additional attributes and relationships	"https://hostname:port/schema/Service/ServiceOffer.schema.json"
type	String	When sub-classing, this defines the sub-class entity name	"ServiceOffer"
serviceOwnerDID	DID	DID of the service owner	

Table 6-9 shows the Information Model for the ServiceCategoryRef. The service category resource is used to group service offers in logical containers. Categories can contain other categories [12].

Table 6-9: ServiceCategoryRef Information Model [12]

Parameter	Type	Description	Example
id	String	Unique identifier of category	"5980"
href	String	Hyperlink reference to the category	"https://hostname:port/serviceCatalogManagement/v3/serviceCategory/5980"
name	String	Name of the category	"TV"
baseType	String	When sub-classing, this defines the super-class	"Category"
schemaLocation	Uri	A URI to a JSON-Schema file that defines additional attributes and relationships	"https://hostname:port/schema/Service/ServiceCategory.schema.json"
type	String	When sub-classing, this defines the sub-class entity name	"ServiceCategory"
referredType	String	The actual type of the target instance when needed for disambiguation	"ServiceCategory"

Tables Table 6-10 and Table 6-11 show the Information Models for the ServiceSpecificationRef and the associated TargetServiceSchema, respectively.

Table 6-10: ServiceSpecificationRef Information Model [12]

Parameter	Type	Description	Example
id	String	Unique identifier of the service specification	"9600"
href	String	Reference of the serviceSpecification	"https://hostname:port/serviceCatalogManagement/v3/serviceSpecification/9600"
name	String	Name of the requiredServiceSpecification	"CFSS_TV"
targetServiceSchema	TargetServiceSchema (Table 6-11))	The reference object to the schema and type of target service which is described by service specification	See Table 6-11
version	String	Service specification version	"2.1"
baseType	String	When sub-classing, this defines the super-class	"ServiceSpecification"

schemaLocation	Uri	A URI to a JSON-Schema file that defines additional attributes and relationships	"https://hostname:port/schema/Service/CustomerFacingServiceSpecification.schema.json"
type	String	When sub-classing, this defines the sub-class entity name	"CustomerFacingServiceSpecification"
referredType	String	The actual type of the target instance when needed for disambiguation	"ServiceSpecification"

Table 6-11: TargetServiceSchema Information Model [12]

Parameter	Type	Description	Example
baseType	String	When sub-classing, this defines the super-class	
schemaLocation	Uri	This field provides a link to the schema describing the target service.	"https://hostname:portschema/Service/RFS.schema.json"
type	String	Class type of the target service	"RFS"

Figure 6-4 distinguishes three types of services, the Customer Facing Services (CFS), the Resource Facing Services (RFS) and Network Services.

A CFS is a service that is obtained as a Product by a Customer. Therefore, the Customer may have specific control over this Service via its associated Product. CFSs are associated with resource technology neutral services. This means that they describe general capabilities and have attributes that are general across many technologies. Examples of such attributes are throughput, latency, loss rate and availability [15]. A CFS is supported by one or more RFSs, as indicated by the *CFServiceRequiresRFServices* aggregator in Figure 6-4 [5].

However, the Customer never knows explicitly which RFSs are being used to support a CFS. More importantly, the Customer shouldn't have to know which RFSs are being used, since the Customer hasn't explicitly obtained them. RFSs are associated with resource technology specific services, which indicates that they have attributes that predominately relate to a specific technology [15]. The *RFRequiresNSs* aggregator defines the NetworkServices that a given RFS has [5].

A NetworkService is defined using the composite pattern. Therefore, NSAtomic and NSComposite are the leaf and composite parts of this pattern, respectively [5]. Generally, a NetworkService should be able to represent CFSs, as indicated by the *NSProvidesCFS* aggregator in Figure 6-4 [5].

6.3 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.

6.3.1 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 and the associated flows (and constraints) for managing its lifecycle.

Table 6-12: 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
places	List<PlaceModel>	Geographical places of significance to the offer
productOfferTerms	List<ProductOfferTerms>	Terms such as licensing associated with the offer
prices	List<ProductPriceOfferings>	Prices associated with the offering
serviceCandidate	ServiceCandidate	The specification of the service
serviceLevelAgreements	List<SLA>	List of SLAs defined for the offer
owner	Party	Corda Party is a Node identity

Below is a sequence diagram that identifies the actions (flows) and participants for product offerings. 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.

6.3.2 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 6-13: 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
didInvitations	Map<DID, Invitation>	A map of invitations for each product in the order to allow consumers to establish DID agent connections

Below is a sequence diagram that identifies the actions (flows) and participants for product orders. 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.

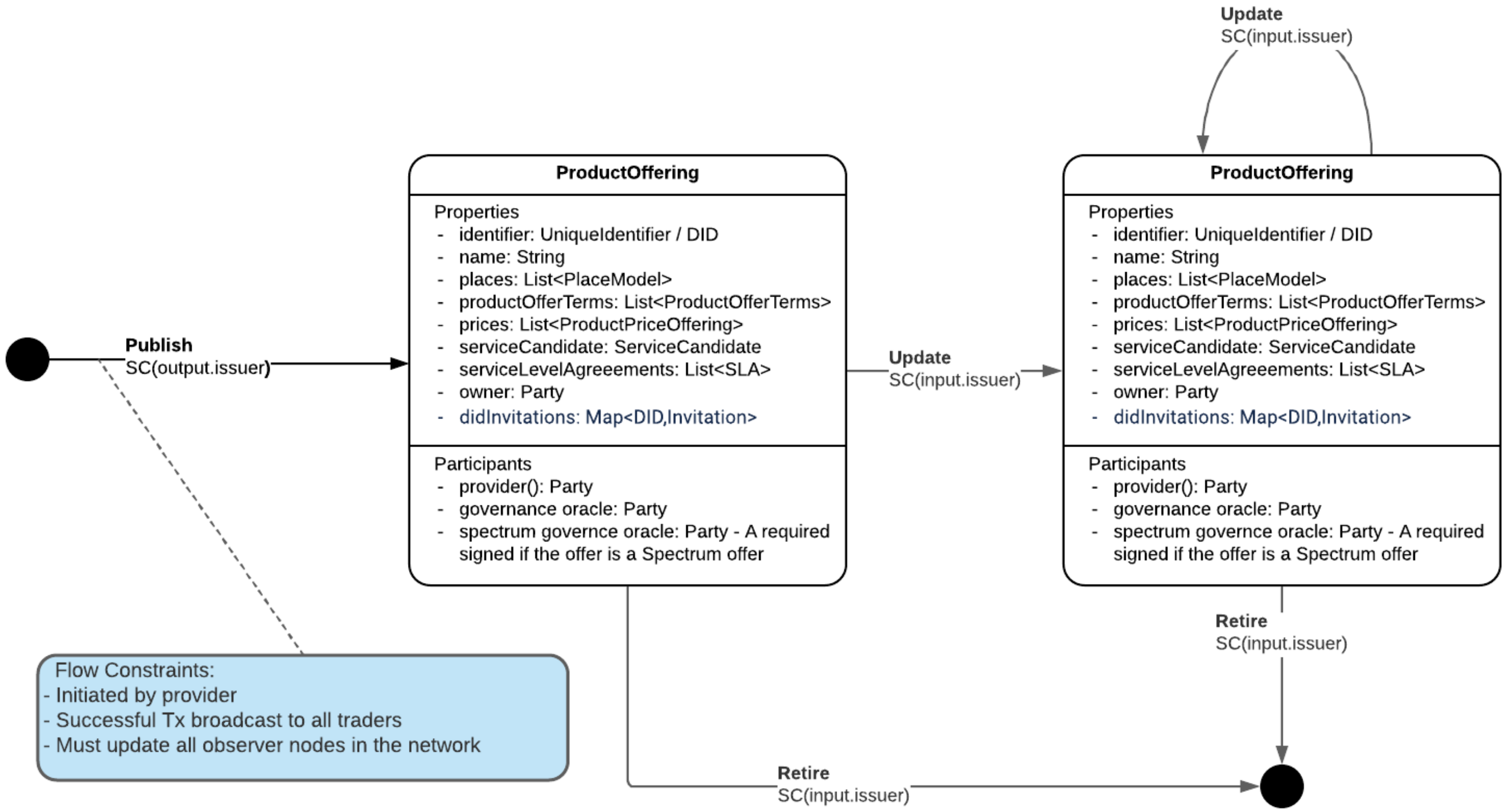

Publish
SC(output.issuer)
ProductOffering
Properties
- identifier: UniqueIdentifier / DID
- name: String
- places: List<PlaceModel>
- productOfferTerms: List<ProductOfferTerms>
- prices: List<ProductPriceOffering>
- serviceCandidate: ServiceCandidate
- serviceLevelAgreeements: List<SLA>
- owner: Party
- didInvitations: Map<DID,Invitation>
Participants
- provider(): Party
- governance oracle: Party
- spectrum governce oracle: Party - A required signed if the offer is a Spectrum offer
Update
SC(input.issuer)
ProductOffering
Properties
- identifier: UniqueIdentifier / DID
- name: String
- places: List<PlaceModel>
- productOfferTerms: List<ProductOfferTerms>
- prices: List<ProductPriceOffering>
- serviceCandidate: ServiceCandidate
- serviceLevelAgreeements: List<SLA>
- owner: Party
- didInvitations: Map<DID,Invitation>
Participants
- provider(): Party
- governance oracle: Party
- spectrum governce oracle: Party - A required signed if the offer is a Spectrum offer
Update
SC(input.issuer)
Retire
SC(input.issuer)
Retire
SC(input.issuer)
Flow Constraints:
- Initiated by provider
- Successful Tx broadcast to all traders
- Must update all observer nodes in the network

Figure 6-5: Product Offering Object

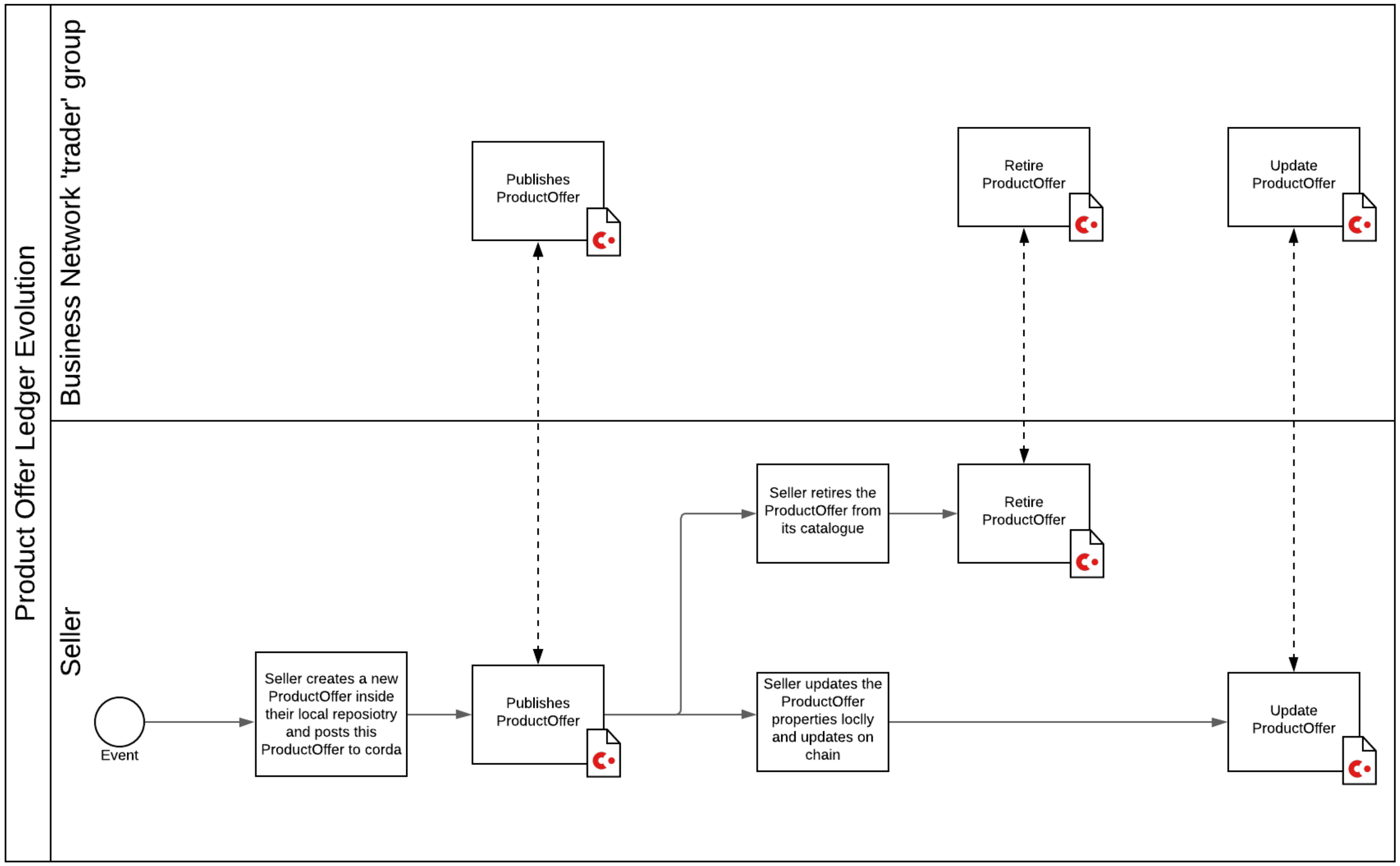

Product Offer Ledger Evolution
Business Network 'trader' group
Seller
Publishes ProductOffer
Retire ProductOffer
Update ProductOffer
Event
Seller creates a new ProductOffer inside their local reposiotry and posts this ProductOffer to corda
Publishes ProductOffer
Seller retires the ProductOffer from its catalogue
Retire ProductOffer
Seller updates the ProductOffer properties loclly and updates on chain
Update ProductOffer

Figure 6-6: Product Offerings Actions

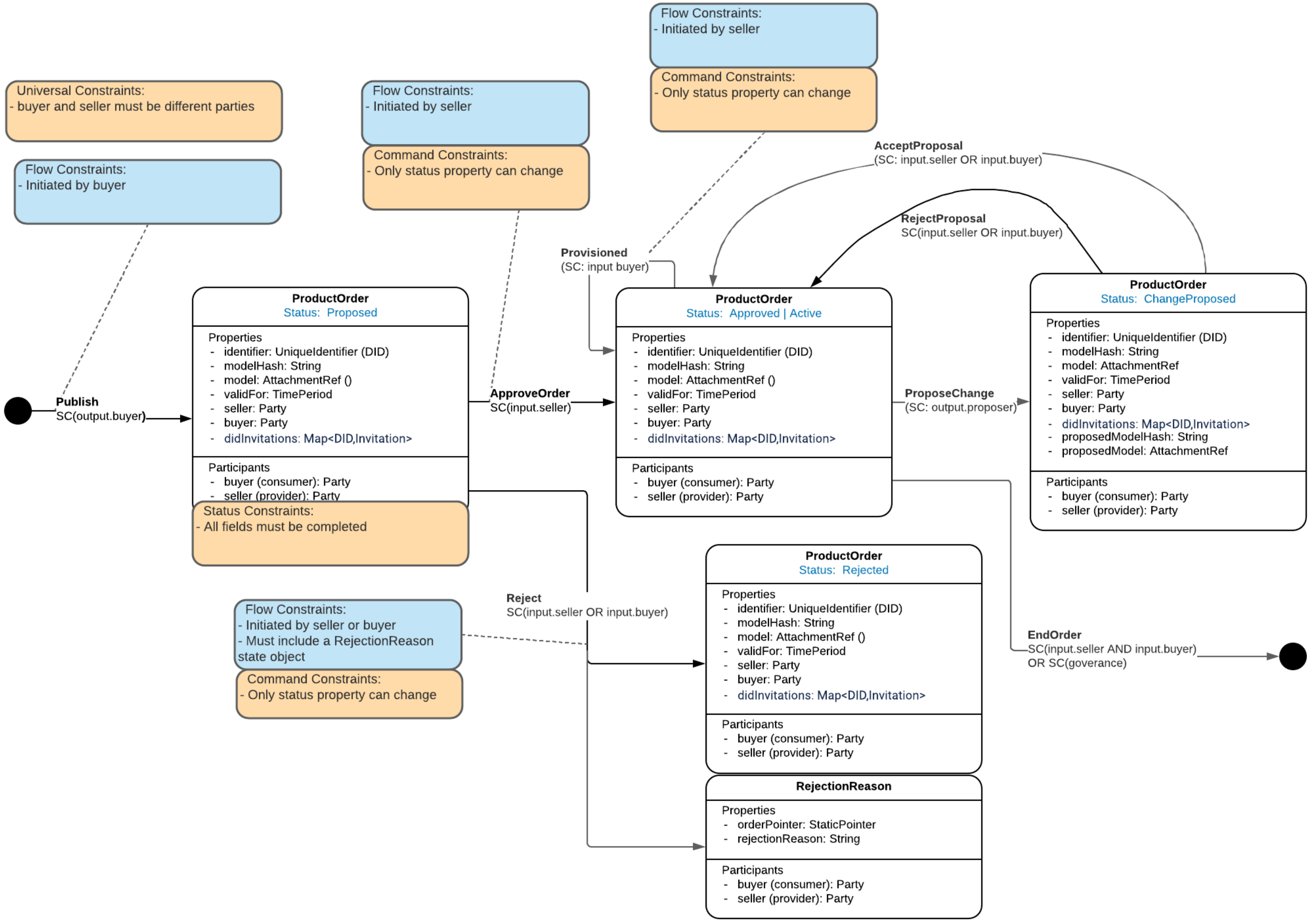

Universal Constraints:
- buyer and seller must be different parties
Flow Constraints:
- Initiated by buyer
Flow Constraints:
- Initiated by seller
Command Constraints:
- Only status property can change
Flow Constraints:
- Initiated by seller
Command Constraints:
- Only status property can change
Publish
SC(output.buyer)
ProductOrder
Status: Proposed
Properties
- identifier: UniqueIdentifier (DID)
- modelHash: String
- model: AttachmentRef ()
- validFor: TimePeriod
- seller: Party
- buyer: Party
- didInvitations: Map<DID,Invitation>
Participants
- buyer (consumer): Party
- seller (provider): Party
Status Constraints:
- All fields must be completed
ApproveOrder
SC(input.seller)
Provisioned
(SC: input buyer)
ProductOrder
Status: Approved | Active
Properties
- identifier: UniqueIdentifier (DID)
- modelHash: String
- model: AttachmentRef ()
- validFor: TimePeriod
- seller: Party
- buyer: Party
- didInvitations: Map<DID,Invitation>
Participants
- buyer (consumer): Party
- seller (provider): Party
AcceptProposal
(SC: input.seller OR input.buyer)
RejectProposal
SC(input.seller OR input.buyer)
ProposeChange
(SC: output.proposer)
ProductOrder
Status: ChangeProposed
Properties
- identifier: UniqueIdentifier (DID)
- modelHash: String
- model: AttachmentRef
- validFor: TimePeriod
- seller: Party
- buyer: Party
- didInvitations: Map<DID,Invitation>
- proposedModelHash: String
- proposedModel: AttachmentRef
Participants
- buyer (consumer): Party
- seller (provider): Party
Reject
SC(input.seller OR input.buyer)
Flow Constraints:
- Initiated by seller or buyer
- Must include a RejectionReason state object
Command Constraints:
- Only status property can change
ProductOrder
Status: Rejected
Properties
- identifier: UniqueIdentifier (DID)
- modelHash: String
- model: AttachmentRef ()
- validFor: TimePeriod
- seller: Party
- buyer: Party
- didInvitations: Map<DID,Invitation>
Participants
- buyer (consumer): Party
- seller (provider): Party
RejectionReason
Properties
- orderPointer: StaticPointer
- rejectionReason: String
Participants
- buyer (consumer): Party
- seller (provider): Party
EndOrder
SC(input.seller AND input.buyer)
OR SC(goverance)

Figure 6-7: Product Order Object

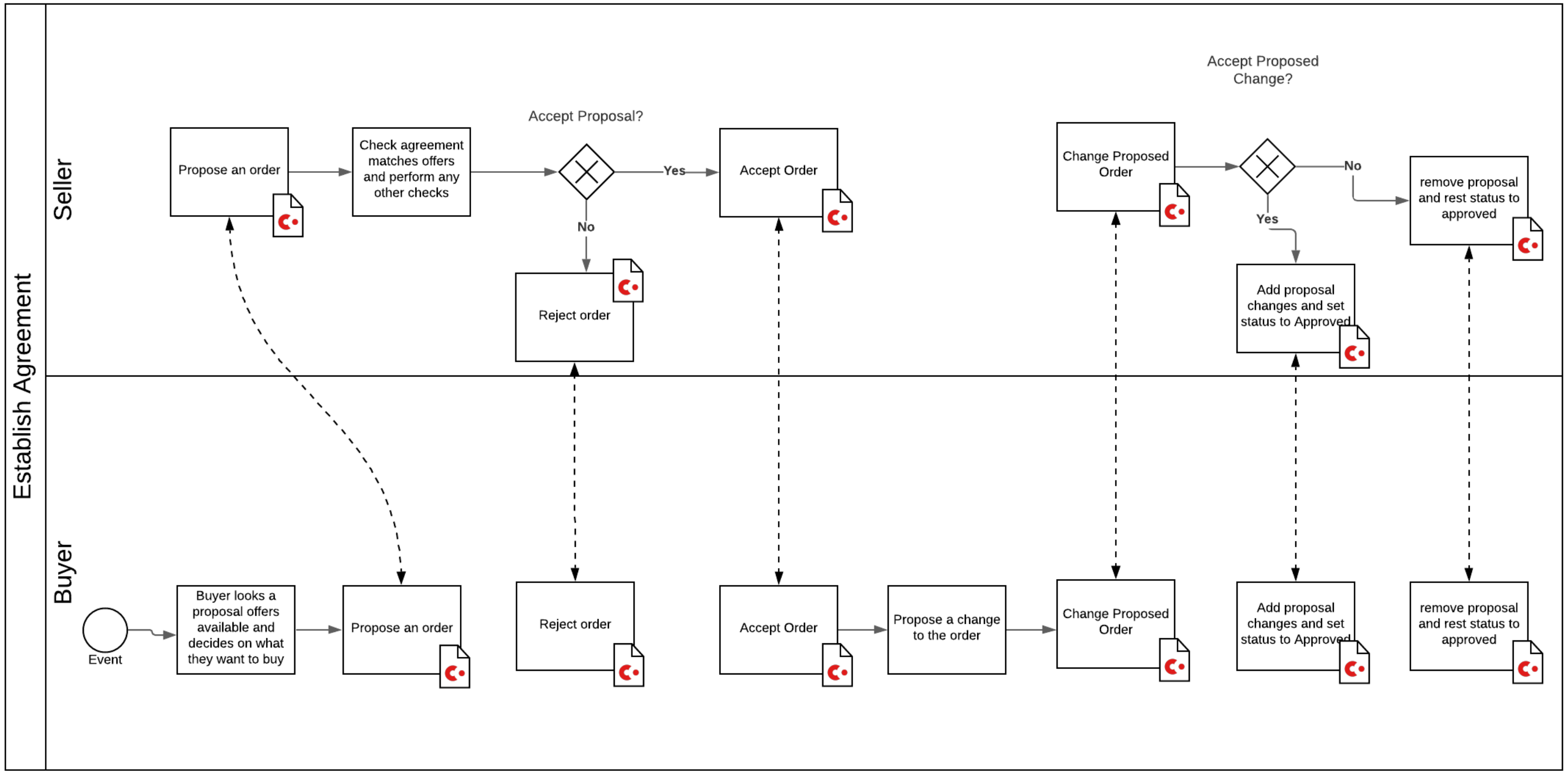

Establish Agreement
Seller
Buyer
Propose an order
Check agreement matches offers and perform any other checks
Accept Proposal?
Yes
No
Accept Order
Reject order
Change Proposed Order
Accept Proposed Change?
No
Yes
remove proposal and rest status to approved
Add proposal changes and set status to Approved
Event
Buyer looks a proposal offers available and decides on what they want to buy
Propose an order
Reject order
Accept Order
Propose a change to the order
Change Proposed Order
Add proposal changes and set status to Approved
remove proposal and rest status to approved

Figure 6-8: Product Order Actions

6.3.3 SLA Violation

When an SLA violation has been detected, this is recorded on the DLT. It references the product order, SLA and specific rule that was violated as well as the recorded – violating – value. Key to the acceptance of the transaction will be the signing by a DID Agent Oracle, whose role is to enforce that the violation state has been generated by the permitted monitoring service. Should the oracle determine that the generating monitoring entity is not permitted, then the transaction will be rejected.

Table 6-14: SLA violation Information Model

Parameter	Type	Description
productOrderRef	StaticPointer	A pointer to the ledger state of the product order to which the violation relates
monitoringServiceId	DID	The identifier of the monitoring service recording the violation
actualValue	String	The value recorded
slaId	String	DID of the SLA in question
ruleId	String	DID of the rule breached

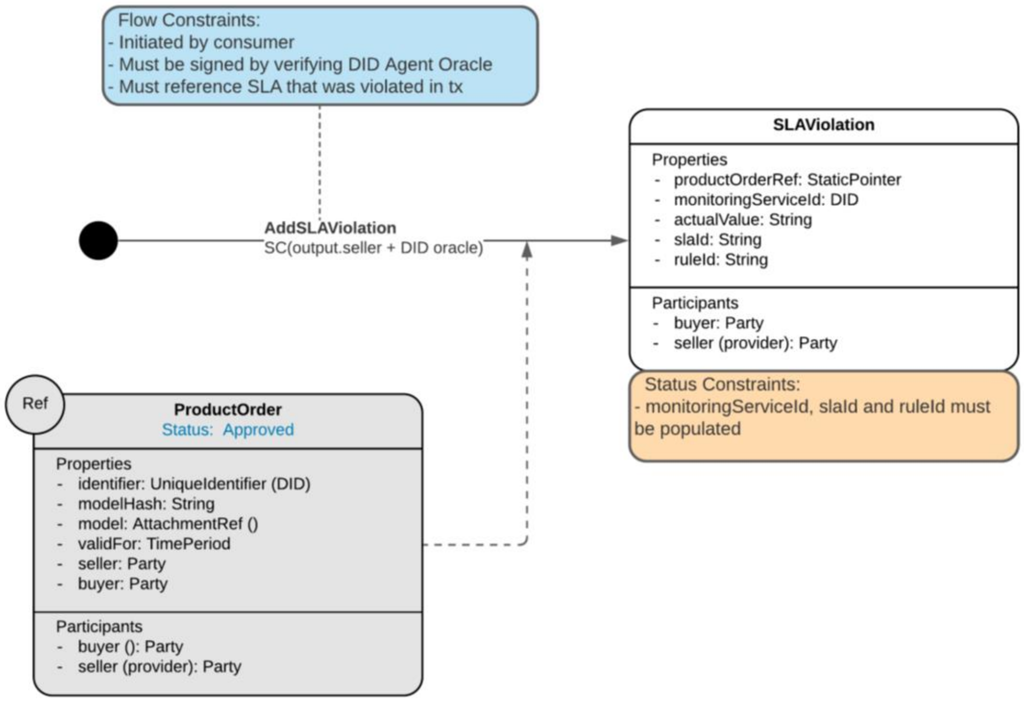

Flow Constraints:
- Initiated by consumer
- Must be signed by verifying DID Agent Oracle
- Must reference SLA that was violated in tx
AddSLAViolation
SC(output.seller + DID oracle)
SLAViolation
Properties
- productOrderRef: StaticPointer
- monitoringServiceId: DID
- actualValue: String
- slaId: String
- ruleId: String
Participants
- buyer: Party
- seller (provider): Party
Status Constraints:
- monitoringServiceId, slaId and ruleId must be populated
Ref
ProductOrder
Status: Approved
Properties
- identifier: UniqueIdentifier (DID)
- modelHash: String
- model: AttachmentRef ()
- validFor: TimePeriod
- seller: Party
- buyer: Party
Participants
- buyer (): Party
- seller (provider): Party

Figure 6-9: SLA Violation Object

The below sequence diagram depicts the expected flow of events on the DLT when publishing an SLA Violation to the ledger.

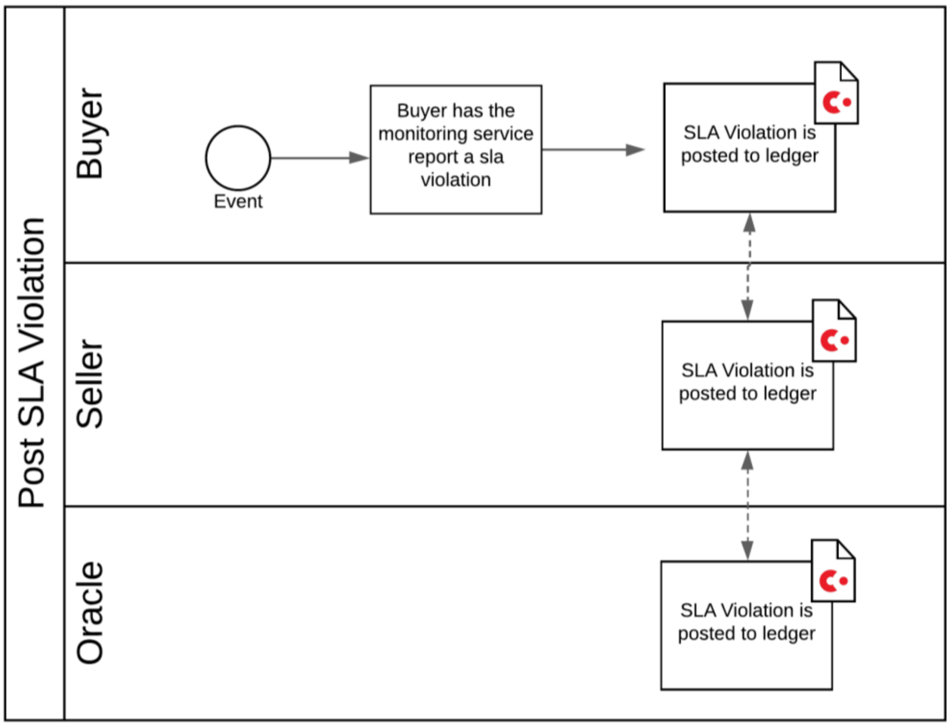

Post SLA Violation
Buyer
Seller
Oracle
Event
Buyer has the monitoring service report a sla violation
SLA Violation is posted to ledger
SLA Violation is posted to ledger
SLA Violation is posted to ledger

Figure 6-10: SLA Violation Flow

6.3.4 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. The below figure illustrates the contract state object and associated flows for state transitions:

Table 6-15: License Terms Information Model

Parameter	Type	Description
identifier	Unique Identifier	A Corda identifier made up of a Corda specific UUID and an "external id" -> the DID of the offer
productOrderRef	Static Pointer	A pointer to the ledger state of the product order to which the violation relates
productOfferId	String	DID of the offer to which these license terms relate
type	Enum	The type of license. Example: 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
lastUsageCalculation	Duration	Period usage was last calculated over
amountTypeEnum	Enum	The type of unit to which the current and amountLimit fields relate Example: USERS | INSTANCES

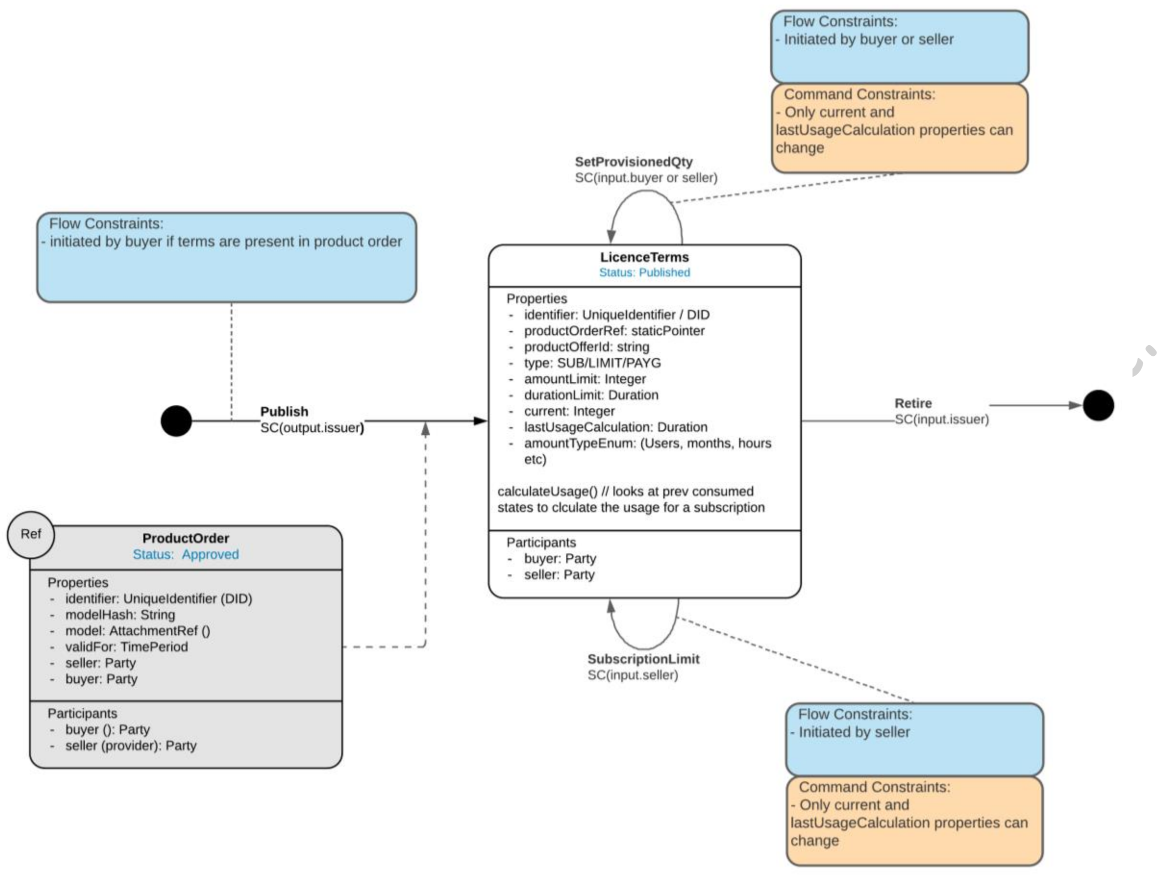

Flow Constraints:
- Initiated by buyer or seller
Command Constraints:
- Only current and lastUsageCalculation properties can change
SetProvisionedQty
SC(input.buyer or seller)
Flow Constraints:
- initiated by buyer if terms are present in product order
LicenceTerms
Status: Published
Properties
- identifier: UniqueIdentifier / DID
- productOrderRef: staticPointer
- productOfferId: string
- type: SUB/LIMIT/PAYG
- amountLimit: Integer
- durationLimit: Duration
- current: Integer
- lastUsageCalculation: Duration
- amountTypeEnum: (Users, months, hours etc)
calculateUsage() // looks at prev consumed states to clculate the usage for a subscription
Participants
- buyer: Party
- seller: Party
Publish
SC(output.issuer)
Retire
SC(input.issuer)
Ref
ProductOrder
Status: Approved
Properties
- identifier: UniqueIdentifier (DID)
- modelHash: String
- model: AttachmentRef ()
- validFor: TimePeriod
- seller: Party
- buyer: Party
Participants
- buyer (): Party
- seller (provider): Party
SubscriptionLimit
SC(input.seller)
Flow Constraints:
- Initiated by seller
Command Constraints:
- Only current and lastUsageCalculation properties can change

Figure 6-11: contract state object and associated flows

7 Conclusions

This report provides a detailed architecture for the 5GZORRO data-driven solution for DLT and distributed intelligent resource discovery and management. Specifically, this deliverable describes the Marketplace, the Governance and the Cross-domain Analytics & Intelligence for AIOps. This report describes the key interactions of their constituent modules to implement the envisaged Zero-Touch SLA Smart Contract, resource discovery, allocation and provisioning services offered by 5GZORRO platform.

The architecture presented in this document is based on deliverable D2.2, where the 5GZORRO high-level reference architecture was introduced. In this document we go to the next level of detail, beyond D2.2, to specify the 5GZORRO components.

The outputs of this deliverable will serve as input for the implementation work that will be carried out in deliverables D3.2 (*Prototypes of evolved 5G Service layer solutions*) and D4.3 (*Final prototype of Zero Touch Service Management with Security and Trust*), as well as WP5 (*Validation through Use Cases*).

7.1 Deliverable contribution to 5GZORRO objectives and KPIs

We summarize in this section how our various objectives are met by the design discussed in this document. Each objective is expanded out into its sub-objectives, and we point to the sections in the document that address the issue in the sub-objective.

Table 7-1: D3.1 contribution to 5GZORRO objectives and KPIs.

OBJECTIVE	Target KPIs	Applicable Design Artifact
OBJ-1. Define a system level architecture combining zero-touch automation solutions and distributed ledger technologies to enable a secure, flexible and multi-stakeholder combination and composition of resources and services in 5G networks.	• *Support actual distributed multi-party service and business configurations (KPI target: more than 3 providers/operators of virtualized resources or services for spectrum, radio/edge/core compute & network).*	3.3 Legal Prose Management, 5.4 Intelligent SLA Monitoring and Breach
	• *Inject and process operational service data (configurations and runtime monitoring and logging) into a multi-party 5G Operational Data Lake (KPI target: at least 10 heterogeneous and diverse operational data sets streamed into 5G Operational Data Lake from various data sources, at least one per provider/operator).*	5.1 Baseline Data Lake Platform, 5.2 Service & Resource Monitoring, 5.3 Monitoring Data Aggregator, 5.4 Intelligent SLA Monitoring and Breach
	• *Expose open APIs to application layer for processing operational data for analytical processes, which discover and "inventorize" various types of resources (KPI target: all external 5GZORRO APIs are exposed via open and public specifications).*	5.5 Smart Resource and Service Discovery application
	• *Automate the overall service lifecycle management with seamless use of heterogeneous virtualization platforms (i.e., VMs and containers, interconnected with various levels and forms of service meshes) across different providers (KPI target: completion of end-to-end provisioning in less than 5 mins, service deletion in less than 1 min).*	D4.1
	• *Support a real-time market for dynamic spectrum allocation allowing business agents to trade on spectrum allocations in space and time (KPI target: Time from transaction to spectrum availability in less than 10 minutes; support of 5GNR, LTE and WiFi technologies).*	3.2 Identity and Permissions Manager, 4.1 Resource and Service Offer Catalogue
OBJ-2. Design and prototype a security and trust framework, integrated with 5G service management platforms, to demonstrate Zero-Day trust establishment in distributed multi-stakeholder environments and automated security management to ensure trusted and secure execution	• *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).*	2.1 DLT for Smart Contracts and Resource offering, 02.2 DLT for distributed identities, 3.1 DLT Governance Manager, 3.2 Identity and Permissions Manager
	• *Enhance a 5G service management framework enabling the detection of security vulnerabilities and compromises and the provision of a set of potential countermeasures to mitigate them using a zero-touch approach (KPI target: identifying 6 different types of common attacks to software*	D4.1

OBJECTIVE	Target KPIs	Applicable Design Artifact
of offloaded workloads across domains in 5G networks	*infrastructures and provide a complete set of countermeasures -filter traffic, divert it to a honeynet, send an alert to the system admin, etc.- for each of them).*	
	• *Support the integration of zero trust hardware platforms (TEE - Trusted Execution Environments) as a root of trust for the monitoring of information and the establishment of end-to-end secure communications enabling critical workloads to go across different tenants and different stakeholders (KPI target: research on the integration evolution of three TEE platforms --one provided by a project partner-- and two other commercial ones to support a fast and secure establishment of end-to-end cross-slice communications for critical workloads).*	D4.1
OBJ-3. Define a Smart Contract ecosystem anchored on a native distributed ledger to allow commercial and technical data provided by 3rd-party users to be standardised and mapped into Smart Contracts, which can be initiated “at will” between multiple untrusted parties.	• *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).*	2.1 DLT for Smart Contracts and Resource offering, 3.3 Legal Prose Management, 4.1 Resource and Service Offer Catalogue, 4.2 Smart Contracts Lifecycle Manager, 5.4 Intelligent SLA Monitoring and Breach
	• *Availability of an Oracle data layer to enable external data sources, processing and results to be requested by SLA smart contracts (KPI target: Oracle data layer accessed by 3 or more parties).*	2.1.2 Oracles
	• *Enable off-chain processing of transactions through payment channels using smart contract in order to enable faster and cheaper transactions compared to on-chain (KPI target: Twice the number of transactions performed over on-chain).*	Part of the Smart Contract DLT capabilities 2.1.2
OBJ-4. Define solutions for secure, automated and intelligent resource discovery, brokerage and selection, operation with SLA to facilitate workload offloading to 3rd-party resources supporting pervasive computing across multiple 5G domains.	• *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).*	4.1 Resource and Service Offer Catalogue, 4.2 Smart Contracts Lifecycle Manager, 5.5 Smart Resource and Service Discovery application
	• *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).*	5.4 Intelligent SLA Monitoring and Breach

OBJECTIVE	Target KPIs	Applicable Design Artifact
	• *Support intent-based API to guide the AI-driven resource discovery system (KPI target: open 5GZORRO API specification for resource discovery).*	5.5 Smart Resource and Service Discovery application
OBJ-5. Define and prototype a secure shared spectrum market to enable real-time trading of spectrum allocations between parties that do not have a pre-established trust relationship.	• *Time to process and enforce new spectrum transactions (i.e., from the moment the transaction is settled until the spectrum becomes available) (KPI target: complete new spectrum transactions in less than 10 minutes).*	2.1N/A
	• *Number of transactions per second handled by the market, which will determine the volume of spectrum transactions processed by the market (KPI target: 20 transactions/second).*	N/A
	• *The authenticity of the market agents, preventing double spending that would allow an agent to trade spectrum rights that it does not own (no explicit KPI target: verification of the built-in property of Blockchains).*	3.1 DLT Governance Manager, 3.2 Identity and Permissions Manager, 4.1 Resource and Service Offer Catalogue
	• *Linkability between market agents and their associated radio access points, which will allow to provide the appropriate spectrum rights to each access point (KPI target: <10M cell towers should be linkable by the system, which is a reasonable EU nation-wide deployment).*	N/A
	• *Ability to enforce the settled spectrum rights and obligations, which will build on lightweight Trusted Execution Environments (TEE) embedded in the radio access points to ensure that the reported spectrum measurements are faithful, and the spectrum allocations settled in the market are enforced (KPI target: Be able to detect spoofing attacks where a base station uses an allocation not authorized by the market).*	D4.1
	• *Agnostic support of various radio technologies, to ensure that the market will work regardless of the considered radio technology (KPI target: 5GNR, LTE and WiFi will be supported).*	N/A
OBJ-6. Realize a cloud-friendly network software licensing framework for location independent network appliances execution.	• *Enable the creation of license agreement templates associated to VNF/NS instances (KPI target: create templates attached to eContract detailing name, context, license conditions, negotiation goal and constraints).*	3.3 Legal Prose Management
	• *Generate vendor independent license token to manage location independent VNFs from 3rd party edge to core datacenter (KPI target: license service creates generic tokens to latter run any vendor VNF across at least 2 network segments).*	D4.1

OBJECTIVE	Target KPIs	Applicable Design Artifact
	• *Instantiate Network Services with VNFs from diverse providers (KPI target: use eContract to include VNF licensed by at least 3 different providers).*	D4.1
OBJ-7. Validate the 5GZORRO zero-touch automation, security and trust in relevant use cases for the implementation of Smart Contracts for Ubiquitous Computing/Connectivity, Dynamic Spectrum Allocation, and Pervasive virtual CDN services over 3rd-party edge resources.	*No specific target to be covered by architecture design*	N/A
OBJ-8. Ensure the long-term success of the project through standardization and dissemination in scientific, industrial, and commercial fora, and by contributing to relevant open source communities & SDOs also exploring synergies with other EU initiatives and projects.	*No specific target to be covered by architecture design*	

8 Appendix: Examples of offer types Information Elements

This section introduces some examples of Information Models for specific types of product offers. The product offers analysed in the next paragraphs are offers for spectrum, cloud, RAN and VNF/CNF resources as well as for network slice and services. These Information Models are based on the general resource and service models illustrated in Section 6.2 and the product offer model explained in Section 6.3.1.

8.1 Information model for Spectrum product offers

The spectrum offer information model is used to indicate the characteristics, such as physical frequency and geographical location, of a licensed spectrum product offering.

Table 8-1: Spectrum Product Offering Information Model

Parameter		Type	Description	Example
id		String	Unique identifier for the spectrum product offering in the Catalogue	“BCNxx77”
href		DID	Distributed identifier of the spectrum product offering	
name		String	A name given to the spectrum product offering	“BCNxx77”
description		String	A string to describe the spectrum product offering	“A xx MHz spectrum resource in band 77 in Barcelona”
validFor		Time Period [9]		
	startDateTime	String	The time instant the spectrum offering starts	"2020-08-23T00:00"
	endDateTime	String	The time instant the spectrum offering expires	"2021-07-22T23:59"
lastUpdate		String	The last time the spectrum offering was updated	"2020-08-22T10:00"
lifecycleStatus		String	Used to indicate the current lifecycle status (e.g., Active, inactive)	“Active”
place		List of GeographicLocation objects [16]	A list of place references to GeographicLocation objects indicating where the spectrum resource is being sold	
	id	String	An id for the geographical location	
	href	String	Reference to the GeographicLocation	
	name	String	A human-readable description of the GeographicLocation	“Barcelona”
serviceLevelAgreement		Object	A reference to the SLA descriptor for Spectrum Products (see Table 8-3)	
	id	String	An id for the SLA	
	href	String	Reference to the SLA	

	name		String	A human-reading description of the SLA	“Spectrum SLA by Telefónica of Spain”
resourceCandidate			Object	Reference to the spectrum resource candidate in the catalogue (the spectrum resource in TM Forum format)	
	id		String	The id of the spectrum resource candidate	
	href		String	DID of the spectrum resource candidate	
	name		String	A human-readable description of the spectrum resource candidate	
category					
	name		String	The type of resource	“Spectrum”
productOfferingTerm			List of objects	A condition under which a ProductOffering is made available to Customers. For instance, a productOffering can be offered with multiple commitment periods	
	name		String	A human-readable name for the spectrum product offering term	
	description		String	A description of the spectrum product offering term	
	duration		Quantity [9] (one object)		
		amount	Integer	The number of units of the time duration	1
		units	String	Time units, e.g., Days, Weeks, Months, Years, etc	“Month”
	validFor		Time Period [9]		
		startDateTime	String	The time instant this spectrum offering term starts	"2020-08-23T00:00"
		endDateTime	String	The time instant this spectrum offering term expires	"2021-07-22T23:59"
productOfferingPrice			List of objects	The price that is asked for the spectrum product offering	
	id		String	The id of the spectrum product offering price	
	href		String	The reference for the spectrum product offering price	
	name		String	A human-readable name for the spectrum product offering price	

The spectrum offer is modelled for a specific number of geographic locations. Based on [16], Table 8-2shows the information model used to describe a geographic location. A Geographic Location is a point, a surface or a volume defined by a group of geographic point(s). These points must be associated with a certain level of accuracy and a spatial reference.

Table 8-2: GeographicLocation Information Model

Parameter	Type	Description	Example
id	String	Unique identifier of the geographic location in the Catalogue	“8980”
href	DID	Distributed identifier of the spectrum product offering	
name	String	A name given to the geographical location	“Barcelona”

geometryType	String	Type of the geographic location - one of: point, line, graph, ring polygon	“Polygon”
accuracy	String	Accuracy of the coordinate specified	“”
spatialRef	String	Geocoding referential	“WGS84”
geometry	List of GeoPoints	List of Geographical Points or GeoPoints. A GeoPoint defines a geographic point through coordinates	
x	String	x coordinate (usually latitude)	“1.430937”
y	String	y coordinate (usually longitude)	“43.597208”
z	String	z coordinate (usually elevation)	“”

The approved SLA is described in terms of SLA rules which contains the metrics, their related values or range, thresholds, valid period or date, consequences in case of violation of any clause of the SLA. It is also assumed that all the metrics are the existing ones which are stored in the 5GZORRO’s “Metrics Library” with their attached references. Besides, each metric is attached to a given Product in the Catalogue with a dedicated reference (DID).

Table 8-3: Spectrum Product Service Level Agreement Information Model

Parameter	Type	Description	Example
id	String	Unique identifier of the SLA	
name	String	Name of the SLA	“Spectrum SLA”
description	String	Description of the SLA	“The spectrum SLA provided by Telefónica of Spain and the Spanish National Regulator”
validityPeriod	Time Period	The period when the clauses of the SLA are applicable	
startTime	String	Date/Time of the beginning of the validityPeriod	"2020-08-23T00:00"
endTime	String	Date/Time of the end of the validityPeriod	"2021-07-22T23:59"
template		SLA template characteristics	
name	String	Name of the template	
href	String	Reference of the template	
description	String	Description of the template	
relatedParty		Parties engaged in the SLA (Regulator, Spectrum Resource Provider)	
role	String	Role attached to each party	“Regulator”
href	String	DID of the party	
state	String	SLA state	
approved	Boolean	Indicates if SLA is approved (true) or not (false)	true
rule		Common pattern or Template of SLA parameter	
id	String	Unique identifier of the metric	
metric	String	Reference of metric stored in the Service Provider “metrics library,” which might include, but not limited to, transmission power, interference level, radio coverage, 5G security,	“Transmission power”
unit	String	Unit of measure of metric	“dBm”
referenceValue	String	Reference value of the metric	“23”
operator	String	Operator used when calculating the rule	“”
tolerance	String	Allowable variation of the metric	+0%

	consequence	String	Defines the action to take as a result of a threshold crossing	Hlink to the contract specifying the action (SLA Violation rule?)

Table 8-4: Spectrum Product Offering Price Information Model

Parameter			Type	Description	Example
id			String	Id of the spectrum product offering price	
name			String	A name given to the spectrum product offering price	SpectrumOfferingPrice1
description			String	A description of the spectrum product offering price	“This pricing describes the recurring charging for a spectrum resource”
validFor			Time Period [9]		
	startDateTime		String	The time instant the spectrum product offering price starts	"2020-08-23T00:00"
	endDateTime		String	The time instant the spectrum product offering price expires	"2021-07-22T23:59"
priceType			String	Describes the price: recurring, discount, allowance…	“Recurring”
recurringChargePeriodType			String	The period of the recurring charge (e.g., 1)	12
recurringChargePeriodLength			Integer	The period to repeat the charging of the price. Possible values are Day, Week, Month, Year, Hour, Minute, Never, etc.	“Month”
percentage			Integer	Discount on the price value	0
price			Quantity		
	unit		String	The price currency, e.g., “EUR.” (ISO4217 norm uses 3 letters to define the currency).	“EUR”
	amount		Float	A positive value determining the amount of money	“10,000”
productOfferingTerm			List of objects	A list of conditions under which a ProductOfferingPrice is made available	
	name		String	Name of the productOfferingTerm	
	description		String	Description of the productOfferingTerm	“The Spectrum Offering price will be available until ***duration***”
	duration		Quantity	Duration of the productOfferingTerm	
		amount	Integer	The number of units of the duration of the productOfferingTerm	“1”
		units	String	Time units, e.g., Days, Weeks, Months, Years, etc	“Week”
	validFor		Time Period [9]	The period for which the productOfferingTerm is valid	
		startDateTime	String	The time instant this spectrum product offering price starts	"2020-08-23T00:00"

endDateTime	String	The time instant this spectrum product offering price expires	"2021-07-22T23:59"
pricingLogicAlgorithm			
id	String	Unique id of the PricingLogicAlgorithm	
href	String	Hyperlink reference of this PricingLogicAlgorithm	
name	String	Name given to the PricingLogicAlgorithm	
description	String	Description of the PricingLogicAlgorithm	
plaSpecId	String	Id of corresponding PricingLogicAlgorithm specification	
validFor	Time Period [9]	The period for which the PricingLogicAlgorithm is valid.	
startDateTime	String	The time instant the PricingLogicAlgorithm starts	
endDateTime	String	The time instant the PricingLogicAlgorithm expires	
tax			
taxAmount	Quantity	Amount of tax expressed in the given currency	
unit	String	The tax currency. ISO4217 norm uses 3 letters (EUR)	"EUR"
value	Float	Numeric value	1,000
taxCategory	String	Tax category. E.g., VAT	"VAT"
taxRate	Integer	Applied rate of the tax	10

Table 8-5: Spectrum resource ResourceCandidate Information Model

Parameter	Type	Description
id	String	Unique identifier in the catalogue
href	DID	Distributed identifier of the spectrum resource candidate
name	String	A name given to the spectrum resource candidate
resourceSpecification	A list of ResourceSpecificationRef [9] elements	A reference to the spectrum resource ResourceSpecification (see Table 8-6)

Table 8-6: Spectrum resource ResourceSpecification Information Model

Parameter	Type	Description
href	DID	Distributed identifier of the spectrum resource specification
name	String	The name of the spectrum resource specification
resourceSpecCharacteristic	List of ResourceSpecCharacteristic [9]	Description of the spectrum resource, starting with the operation mode of the spectrum (TDD or FDD) in Table 8-7, then followed by the start (Table 8-8) and end (Table 8-9) downlink (DL) frequencies, and, finally, the start (Table 8-10) and end (Table 8-11) uplink (UL) frequencies. If the spectrum resource is in TDD mode, DL and UL start

		frequency values must be the same, and the same reasoning applies to the end frequency values. If FDD operation, DL and UL bands cannot overlap.

Table 8-7: Spectrum operation mode ResourceSpecCharacterstic Information Model

Parameter	Type	Description
name	String	operationMode
description	String	The operation mode of the spectrum. The accepted values are "TDD" or "FDD"
configurable	Boolean	False
validFor	Time Period [9]	Date of expiration of the current description
isUnique	Boolean	True
resourceSpecCharacteristicValue	1 object	Spectrum operation modality object
valueType	String	String
value	String	"TDD" or "FDD"

Table 8-8: Start DL frequency ResourceSpecCharacterstic Information Model

Parameter	Type	Description
name	String	startFreqDl
description	String	The start DL frequency of the spectrum resource
configurable	Boolean	False
validFor	Time Period [9]	Date of expiration of the current description
isUnique	Boolean	True
resourceSpecCharacteristicValue	1 object	
valueType	String	Numeric
value	Float	A central frequency value
unitOfMeasure	String	Megahertz [MHz]

Table 8-9: End DL frequency ResourceSpecCharacterstic Information Model

Parameter	Type	Description
name	String	endFreqDl
description	String	The end DL frequency of the spectrum resource
configurable	Boolean	False
validFor	Time Period [9]	Date of expiration of the current description
isUnique	Boolean	True
resourceSpecCharacteristicValue	1 object	
valueType	String	Numeric
value	Float	A central frequency value
unitOfMeasure	String	Megahertz [MHz]

Table 8-10: Start UL frequency ResourceSpecCharacterstic Information Model

Parameter	Type	Description
name	String	startFreqUl
description	String	The start UL frequency of the spectrum resource
configurable	Boolean	False
validFor	Time Period [9]	Date of expiration of the current description
isUnique	Boolean	True
resourceSpecCharacteristicValue	1 object	

valueType	String	Numeric
value	Float	A central frequency value
unitOfMeasure	String	Megahertz [MHz]

Table 8-11: End UL frequency ResourceSpecCharacterstic Information Model

Parameter	Type	Description
name	String	endFreqUl
description	String	The end UL frequency of the spectrum resource
configurable	Boolean	False
validFor	Time Period [9]	Date of expiration of the current description
isUnique	Boolean	True
resourceSpecCharacteristicValue	1 object	
valueType	String	Numeric
value	Float	A central frequency value
unitOfMeasure	String	Megahertz [MHz]

8.2 Information model for Cloud product offers

This section analyses the cloud product offer information model that is used in the 5GZORRO Marketplace to describe the elements that compose the product to trade with. In Table 8-12 the fields that compose the cloud product are depicted.

Table 8-12: Cloud Product Offering Information Model

Parameter	Type	Description	Example
id	String	Unique identifier for the cloud product offering in the Catalogue	
href	DID	Distributed identifier of the cloud product offering	
name	String	A name given to the cloud product offering	
description	String	A name to describe the cloud product offering	Cloud resources description
validFor	Time Period [9]		
startDateTime	String	The time instant the cloud offering starts	"2020-08-23T00:00"
endDateTime	String	The time instant the cloud offering expires	"2021-07-22T23:59"
resourceSpecification	ResourceSpecificationRef [9]	A reference to the cloud resource ResourceSpecification (see Table 6-5)	

8.3 Information model for RAN product offers

The RAN product offer information element is used to indicate the RAN infrastructure elements the RAN infrastructure provider is sharing in the 5GZORRO architecture.

Table 8-13: RAN Product Offering Information Model

Parameter	Type	Description	Example
id	String	Unique identifier for the RAN product offering in the Catalogue	
href	DID	Distributed identifier of the RAN product offering	
name	String	A name given to the RAN product offering	
description	String	A name to describe the RAN product offering	"A RAN product offering "
validFor	Time Period [9]		
startDateTime	String	The time instant the RAN offering starts	"2020-08-23T00:00"
endDateTime	String	The time instant the RAN offering expires	"2021-07-22T23:59"
lastUpdate	String	The last time the RAN offering was updated	
lifecycleStatus	String	Used to indicate the current lifecycle status (e.g., Active, inactive)	“Active”
place	List of GeographicalAddress objects [17]	A list of place references to GeographicAddress objects indicating where the RAN resource is being sold	
id	String	An id for the geographical location	
href	String	Reference to the GeographicAddress	
name	String	A human-readable description of the GeographicalAddress	“Barcelona”
serviceLevelAgreement	Object	A reference to the SLA descriptor. The SLA information model for RAN products is given in Table 8-15	
id	String	An id for the SLA	
href	String	Reference to the SLA	
name	String	A human-reading description of the SLA	“Telefónica SLA”
category			
name	String	The type of resource	“RAN”
productOfferingTerm	List of objects	A condition under which a ProductOffering is made available to Customers. For instance, a productOffering can be offered with multiple commitment periods	
name	String	A human-readable name for the RAN product offering term	“RAN-BCN”
description	String	A description of the RAN product offering term	“Telefónica’s RAN resources in Barcelona”
duration	Object		

amount	Integer	The number of units of the time duration	“1”
units	String	Time units, e.g., Days, Weeks, Months, Years, etc	“Year”
validFor	Time Period [9]		
startDateTime	String	The time instant this RAN offering term starts	"2017-08-23T00:00"
endDateTime	String	The time instant this RAN offering term expires	"2018-08-23T00:00"
productOfferingPrice	List of objects	The price that is asked for the RAN product offering	
id	String	The id of the RAN product offering price	
href	String	The reference for the RAN product offering price	
name	String	A human-readable name for the RAN product offering price	“RAN offering in Barcelona”

Typically, the RAN infrastructure is composed of physical (non-virtual) resources that are deployed somewhere. For this reason, the RAN offer shall include information describing the precise location of each RAN resource of the RAN offer. This location information becomes essential when selecting the right RAN resources for a service aiming at covering a specific geographic area. The RAN resource location information consists of an address and, optionally, the geographic coordinates, as specified by the TMForum in its Address Information Model [17]. Table 8-14 contains the different fields in the Address Information Model.

Table 8-14: Address Information Model [17]

Parameter	Type	Description	Example
id	String	Unique identifier of the geographic location in the Catalogue	“8980”
streetNr	Integer	Number identifying a specific property on a public street. It may be combined with streetNrLast for ranged addresses	71
streetNrSuffix	String	The first street number suffix	“”
streetNrLast	Integer	Last number in a range of street numbers allocated to a property	“”
streetNrLastSuffix	String	Last street number suffix for a ranged address	“”
streetName	String	Name of the street or other street type	“Rambla del Poblenou”
streetType	String	Alley, avenue, boulevard, brae, crescent, drive, highway, lane, terrace, parade, place, tarn, way, wharf?	“”
streetSuffix	String	A modifier denoting a relative direction	“”
locality	String	"An area of defined or undefined boundaries within a local authority or other legislatively defined area, usually rural or semi rural in nature." [ANZLIC-STREET], or a suburb "a bounded locality within a city, town	“Poblenou”

		or shire principally of urban character " [ANZLIC-STREET]	
city	String	City that the address is in	“Barcelona”
postcode	String	A descriptor for a postal delivery area, used to speed and simplify the delivery of mail	“08005”
stateOrProvince	String	The State or Province that the address is in	“Catalonia”
country	String	Country that the address is in	“Spain”
geoCode	Geographic coordinates	Geographic coordinates to point to the address	
latitude	String	Latitude	“1.430937”
longitude	String	Longitude	“43.597208”
geographicDatum	String	Geocoding referencial	“WGS84”

Table 8-15: RAN Product Service Level Agreement Information Model

Parameter	Type	Description	Example
id	String	Unique identifier of the SLA	
name	String	Name of the SLA	
description	String	Description of the SLA	“SLA for Telefonica’s RAN”
validityPeriod	String	The period when the clauses of the SLA are applicable	
startTime	String	Date/Time of the beginning of the validityPeriod	"2020-08-23T00:00"
endTime	String	Date/Time of the end of the validityPeriod	"2021-07-22T23:59"
template		SLA template characteristics	
name	String	Name of the template	
href	String	Reference of the template	
description	String	Description of the template	
relatedParty		Parties engaged in the SLA (RAN Resource Provider, RAN Resource Consumer)	
role	String	Role attached to each party	“RAN Resource Provider”
href	String	DID of the party	
state	String	SLA state	
approved	Boolean	Indicates if SLA is approved (true) or not (false)	true
rule		Common pattern or Template of SLA parameter	
id	String	Unique identifier of the metric	
metric	String	Reference of metric stored in the Service Provider “metrics library”	“Transmission power”
unit	String	Unit of measure of metric	“dBm”
referenceValue	String	Reference value of the metric	“23”
operator	String	Operator used when calculating the rule	“”
tolerance	String	Allowable variation of the metric	+0%
consequence	String	Defines the action to take as a result of a threshold crossing	Hlink to the contract specifying the action (SLA Violation rule?)

Table 8-16: RAN Product Offering Price Information Model

Parameter	Type	Description	Example
id	String	Id of the RAN product offering price	
href	String	Distributed identifier of the RAN product offering price	
name	String	A name given to the RAN product offering price	
description	String	A description of the RAN product offering price	
validFor	Time Period [9]		
startDateTime	String	The time instant the RAN product offering price starts	"2017-08-23T00:00"
endDateTime	String	The time instant the RAN product offering price expires	"2017-08-29T00:00"
priceType	String	Describes the price: recurring, discount, allowance…	"Recurring"
recurringChargePeriodType	Integer	The period of the recurring charge (e.g., 1)	"1"
recurringChargePeriodLength	String	The period to repeat the charging of the price. Possible values are Day, Week, Month, Year, Hour, Minute, Never, etc.	"Month"
lastUpdate	String	The last time the RAN product offering price was updated	"2017-08-23T00:00"
lifecycleStatus	String	Used to indicate the current lifecycle status (e.g., Active, Inactive)	"Active"
isBundle	Boolean	It denotes if the RAN product offering represents a single productOffering (false), or a bundle of productOfferings (true)	false
percentage	Integer	Discount on the price value	"0"
price	Quantity		
unit	String	The price currency (ISO4217 norm uses 3 letters to define the currency)	"EUR"
amount	Float	A positive value determining the amount of money	100.0
productOfferingTerm	List of objects	A list of conditions under which a ProductOfferingPrice is made available	
name	String	Name of the productOfferingTerm	
description	String	Description of the productOfferingTerm	
duration	Quantity	Duration of the productOfferingTerm	
amount	Integer	The number of units of the duration of the productOfferingTerm	1
units	String	Time units, e.g., Days, Weeks, Months, Years, etc	"Year"
validFor	Time Period [9]	The period for which the productOfferingTerm is valid	
startDateTime	String	The time instant this RAN product offering price starts	"2017-08-23T00:00"

endDateTime	String	The time instant this RAN product offering price expires	"2018-08-23T00:00"
pricingLogicAlgorithm			
id	String	Unique id of the PricingLogicAlgorithm	
href	String	Hyperlink reference of this PricingLogicAlgorithm	
name	String	Name given to the PricingLogicAlgorithm	
description	String	Description of the PricingLogicAlgorithm	
plaSpecId	String	Id of corresponding PricingLogicAlgorithm specification	
validFor	Time Period [9]	The period for which the PricingLogicAlgorithm is valid.	
startDateTime	String	The time instant the PricingLogicAlgorithm starts	
endDateTime	String	The time instant the PricingLogicAlgorithm expires	
tax			
taxAmount	Quantity	Amount of tax expressed in the given currency	
unit	String	The tax currency. ISO4217 norm uses 3 letters	"EUR"
value	Float	Numeric value	10.0
taxCategory	String	Tax category. E.g., VAT	"VAT"
taxRate	Integer	Applied rate of the tax	10

Table 8-17: RAN Product Specification Information Model

Parameter	Type	Description	Example
id	String	Unique identifier for the RAN product specification in the catalogue	
href	String	Distributed identifier of the RAN product specification	
name	String	A name given to the RAN product specification	"RAN product xx"
description	String	A name to describe the RAN product specification	"A RAN Product Specification"
isBundle	Boolean	It determines whether the RAN productSpecification represents a single productSpecification (false), or a bundle of productSpecifications (true)	false
lastUpdate	String	The last time the RAN product specification was updated	"2017-08-23T00:00"
lifecycleStatus	String	Used to indicate the current lifecycle status (e.g., Active, inactive)	"Active"
validFor	Time Period [9]		
startDateTime	String	The time instant the RAN product specification starts	"2017-08-23T00:00"
endDateTime	String	The time instant the RAN product specification expires	"2018-08-23T00:00"
relatedParty	Object		

	id	String	Id of the related party	
	href	DID	DID of the related party	
	role	String	The role of the offering related party	“RAN provider”
	name	String	The name of the stakeholder providing the RAN offer	“Telefónica of Spain”
resourceSpecification		A list of ResourceSpecificationRef elements [9]	A reference to the RAN resource ResourceSpecification references (see Table 8-18)	
productSpecCharacteristic		List of objects	A list of objects defining the RAN. Will match the RAN resourceCandidateSpecCharacteristics in Table 8-18	

Table 8-18: RAN resource ResourceSpecification Information Model

Parameter	Type	Description	Example
href	DID	Distributed identifier of the RAN resource specification	
name	String	The name of the RAN resource specification	“A Telefonica RAN Resource”
resourceSpecCharacteristic	List of ResourceSpecCharacteristic [9]	Description of the RAN resource key features that are relevant for the definition of the applicable RAN range, i.e., pairs of central frequency and bandwidth Information Models (see Table 8-19 and Table 8-20)	

Table 8-19: Operation band ResourceSpecCharacteristic Information Model

Parameter		Type	Description	Example
name		String	operationBand	“n78”
description		String	In case of a cellular base station or Wi-Fi access point, the supported operation band (3GPP) or channel (Wi-Fi). Will contain as many values as supported	Operation in NR band 78
configurable		Boolean	Denotes whether the operation band is configurable or not. By default, the modification of the operation band is not permitted	false
validFor		Time period [9]	Date of expiration of the current description	
isUnique		Boolean	Denotes whether multiple Operation band ResourceSpecCharacterstic referred to the same RAN resource can have the same value (false) or not (true)	true
resourceSpecCharacteristicValue		List of objects	A list of operation band values	
	valueType	String	The value type	“Numeric”
	value	Integer	The band/channel number	“78”
	unitOfMeasure	String	Integer	“”

Table 8-20: Quota ResourceSpecCharacterstic Information Model

Parameter		Type	Description	Example
name		String	quota	
description		String	The percentage of the passive resources shared (e.g., backhaul link capacity, baseband processing capacity, etc.)	"Processing capacity"
configurable		Boolean	Denotes whether the quota is configurable or not. The purchase might be smaller than the quota (maximum value)	true
validFor		Time period [9]	Date of expiration of the current quota description	
isUnique		Boolean	Denotes whether the quota value is unique or not	false
resourceSpecCharacteristicValue		List of objects	A list of quota values	
	valueType	String	The type of the value	"Numeric"
	value	Float	Quota value	20.0
	unitOfMeasure	String	The unit of measure	"Percentage [%]"

8.4 Information model for VNF/CNF product offers

Table 8-21: VNF ResourceCandidate Information Model

Parameter	**Type**	**Description**
id	String	Unique identifier in the catalogue
href	DID	Distributed identifier of the VNF resource candidate
name	String	A name given to the VNF resource candidate
resourceSpecification	ResourceSpecificationRef [9]	A reference to the VNF resource ResourceSpecification (see Table 8-22)

Table 8-22: VNF resource ResourceSpecification Information Model

Parameter	**Type**	**Description**
href	DID	Distributed identifier of the VNF resource specification
name	String	The name of the VNF resource specification
resourceSpecCharacteristic	List of ResourceSpecCharacteristic [9]	Description of the VNF resource key features (see Tables Table 8-23 through Table 8-35)

Table 8-23: vnfdId ResourceSpecCharacteristic Information Model

Parameter	**Type**	**Description**
name	String	vnfdId
description	String	Unique identifier of the VNF Package that contains this VNFD
configurable	Boolean	true (any modification of the content of the VNFD or any other modification of the VNF Package shall result in a new VNFD Identifier)
validFor	Time period [9]	Date of expiration of the current description
isUnique	Boolean	true
resourceSpecCharacteristicValue		One object that represent the identifier
valueType	String	UUID
value	String	Identifier

Table 8-24: vnfProvider ResourceSpecCharacteristic Information Model

Parameter	**Type**	**Description**
name	String	vnfProvider
description	String	Provider of the VNF and of the VNFD
configurable	Boolean	false
validFor	Time period [9]	Date of expiration of the current vnfProvider
isUnique	Boolean	true
resourceSpecCharacteristicValue		One object that represent the provider
valueType	String	String
value	String	Provider’s name

Table 8-25: vnfProductName ResourceSpecCharacteristic Information Model

Parameter	Type	Description
name	String	vnfProductName
description	String	Name to identify the VNF Product
configurable	Boolean	false (Invariant for the VNF Product lifetime)
validFor	Time period [9]	Date of expiration of the current vnfProductName
isUnique	Boolean	false
resourceSpecCharacteristicValue		One object that represent the VNF product name
valueType	String	String
value	String	VNF product name

Table 8-26: vnfSoftwareVersion ResourceSpecCharacteristic Information Model

Parameter	Type	Description
name	String	vnfSoftwareVersion
description	String	Software version of the VNF
configurable	Boolean	true (This is changed when there is any change to the software that is included in the VNF Package)
validFor	Time period [9]	Date of expiration of the current vnfSoftwareVersion
isUnique	Boolean	false
resourceSpecCharacteristicValue		One object that represent the VNF software version
valueType	String	Version
value	String	Version of the VNF

Table 8-27: vnfdVersion ResourceSpecCharacteristic Information Model

Parameter	Type	Description
name	String	vnfdVersion
description	String	Specifies the version of the VNFD
configurable	Boolean	false
validFor	Time period [9]	Date of expiration of the current vnfdVersion
isUnique	Boolean	false
resourceSpecCharacteristicValue		One object that represent the VNFD version
valueType	String	Version
value	String	Version of the VNFD

Table 8-28: vnfProductInfoName ResourceSpecCharacteristic Information Model

Parameter	Type	Description
name	String	vnfProductInfoName
description	String	Human readable name for the VNF Product
configurable	Boolean	true (Can change during the VNF Product lifetime)
validFor	Time period [9]	Date of expiration of the current vnfProductInfoName
isUnique	Boolean	false
resourceSpecCharacteristicValue		One object that represent the VNFD version
valueType	String	String
value	String	VNF product info name

Table 8-29: vnfProductInfoDescription ResourceSpecCharacteristic Information Model

Parameter	Type	Description
name	String	vnfProductInfoDescription
description	String	Human readable description of the VNF Product
configurable	Boolean	true (Can change during the VNF Product lifetime)
validFor	Time period [9]	Date of expiration of the current vnfProductInfoDescription
isUnique	Boolean	false
resourceSpecCharacteristicValue		One object that represent the VNF product info description
valueType	String	String
value	String	VNF product info description

Table 8-30: vnfdRef ResourceSpecCharacteristic Information Model

Parameter	Type	Description
name	String	vnfdRef
description	String	A reference to the VNF Descriptor
configurable	Boolean	false
validFor	Time period [9]	Date of expiration of the current vnfdRef
isUnique	Boolean	true
resourceSpecCharacteristicValue		One object that represent the reference to the VNF Descriptor
valueType	String	String
value	String	Reference to the VNF Descriptor

Table 8-31: localizationLanguage ResourceSpecCharacteristic Information Model

Parameter	Type	Description
name	String	localizationLanguage
description	String	Information about localization languages of the VNF
configurable	Boolean	false
validFor	Time period [9]	Date of expiration of the current localizationLanguage
isUnique	Boolean	False
resourceSpecCharacteristicValue		A list of Strings that represents the localization languages of the VNF
valueType	String	String
value	String	Localization language of the VNF

Table 8-32: configurableProperties ResourceSpecCharacteristic Information Model

Parameter	Type	Description
name	String	configurableProperties
description	String	Describes the configurable properties of the VNF (e.g., related to auto scaling and auto healing)
configurable	Boolean	false
validFor	Time period [9]	Date of expiration of the current configurableProperties
isUnique	Boolean	False

resourceSpecCharacteristicValue		A list of objects that represents the configurable properties of the VNF
valueType	String	ConfigurableProperty
value	Object	Property + Boolean

Table 8-33: cpuRequirements ResourceSpecCharacteristic Information Model

Parameter	**Type**	**Description**
name	String	cpuRequirements
description	String	CPU requirements of the VM/ Virtualisation Container/OS Containers realizing this VNF (vdu)
configurable	Boolean	False
validFor	Time period [9]	Date of expiration of the current cpuRequirements
isUnique	Boolean	False
resourceSpecCharacteristicValue	List of Objects	A list of objects that represents the amount of CPU required based on the particular deployment modes
valueType	String	CpuRequirementsDeployBased
value	Object	Amount of CPU required + reference VDU/etc…
unitOfMeasure	String	cpu_num * frequency [n * Ghz]

Table 8-34: memoryRequirements ResourceSpecCharacteristic Information Model

Parameter	**Type**	**Description**
name	String	memoryRequirements
description	String	Memory requirements of the VM/ Virtualisation Container/OS Containers realizing this VNF (vdu)
configurable	Boolean	False
validFor	Time period [9]	Date of expiration of the current memoryRequirements
isUnique	Boolean	False
resourceSpecCharacteristicValue		A list of objects that represents the amount of memory required based on the particular deployment modes
valueType	String	MemoryRequirementsDeployBased
value	Object	Amount of Memory required + reference VDU/etc.
unitOfMeasure	String	Memory size [GB]

Table 8-35: storageRequirements ResourceSpecCharacteristic Information Model

Parameter	**Type**	**Description**
name	String	storageRequirements
description	String	Storage requirements of the VM/ Virtualisation Container/OS Containers realizing this VNF (vdu)
configurable	Boolean	False
validFor	Time period [9]	Date of expiration of the current storageRequirements
isUnique	Boolean	False
resourceSpecCharacteristicValue		A list of objects that represents the amount of storage required based on the particular deployment modes
valueType	String	StorageRequirementsDeployBased

8.5 Information models for Network Slice and Network Service product offers

These information models are based on work from the TMForum Information Framework. Following the TMForum 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. 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.

Table 8-36 and Table 8-37 outline the main parameters of a network slice in the service catalogue, via the ServiceCandidate and the ServiceSpecificationIM, which is adopted as (reusable) core building block to assemble service catalogue and inventory entries in the 5GZORRO marketplace. Similarly, Table 8-38 and Table 8-39 depict the ServiceCandidate and the ServiceSpecification of network services, respectively.

Table 8-36: Network slice ServiceCandidate

Parameter	Type	Description
id	String	Unique identifier in the catalogue (e.g., UUID)
href	DID	Distributed identifier of the network slice candidate
name	String	The name of the network slice candidate
resourceSpecification	ResourceSpecificationRef	A reference to the network slice ServiceSpecification

Table 8-37: Network slice ServiceSpecification Information Model

Parameter	Type	Description
id	String	Unique Service Specification id in the catalogue
href	DID	Distributed identifier of the network slice specification
name	String	The name of the network slice specification
serviceSpecCharacteristic	List of ServiceSpecCharacteristic	Description of the network slice characteristics representing key features that are relevant for network services to be supported by the slice. Note that attributes here included (values, configurability, etc.) are inherited from the resource's specifications associated to the slice.

Table 8-38: Network service ServiceCandidate Information Model

Parameter	Type	Description
id	String	Unique identifier in the catalogue
href	DID	Distributed identifier of the network service candidate
name	String	The name of the network service candidate
serviceSpecification	ServiceSpecificationRef	A reference to the network service ServiceSpecification

Table 8-39: Network service ServiceSpecification Information Model

Parameter	Type	Description
id	String	Unique Service Specification id in the catalogue
href	DID	Distributed identifier of the network service specification
name	String	The name of the network service specification

serviceSpecCharacteristic	List of ServiceSpecCharacteristic	Description of the network service characteristics representing key features that are relevant for customers obtaining this service via product offers.

By using the aforementioned service candidates and specifications, product offers are created to expose network slices and network services available for discovery and purchase in the 5GZORRO Marketplace. Relevant fields of the ProductOfferingInformation Model used for network slices and network services offers are shown in Table 8-40 and Table 8-41, respectively.

Table 8-40: Network slice ProductOffering information model

Parameter	Type	Description
id	String	Unique internal id for the network slice offer in the catalogue
href	DID	Distributed identifier of the network slice offer
name	String	Name of the network slice offer
agreement	List of AgreementRef	Agreements of the network slice provider with network slice customers, for instance, service providers
place	List of PlacRef	Geographical places for which the network slice offer is valid
productOfferingPrice	List of ProductOfferingPriceRef	Reference to the pricing models available for the network slice offer.
serviceCandidate	ServiceCandidateRef	Reference to the network slice service candidate
productOfferingTerm	List of productOfferingTerm	Business conditions for which the network slice offer is valid
serviceLevelAgreement	List of SLA Ref	Service Level Agreements available for the network slice offer

Table 8-41: Network service ProductOffering information model

Parameter	Type	Description
id	String	Unique internal id for the network service offer in the catalogue
href	DID	Distributed identifier of the network service offer
name	String	Name of the network service offer
agreement	List of AgreementRef	Agreements of the network service provider with network service customers
place	List of PlacRef	Geographical places for which the network service offer is valid
productOfferingPrice	List of ProductOfferingPriceRef	Reference to the pricing models available for the network service product offer.
serviceCandidate	ServiceCandidateRef	Reference to the network service candidate
productOfferingTerm	List of productOfferingTerm	Business conditions for which the network service offer is valid
serviceLevelAgreement	List of SLARef	Service Level Agreements available for the network service offer

9 References

[1] D2.2: Design of the 5GZORRO Platform for Security & Trust

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

[3] D4.1: Design of Zero Touch Service Management with Security & Trust Solutions

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

[5] TM Forum, ZOOM Information Model Snapshot, TD234 Release 14.5.1, 2015

[6] ETSI GS MEC 010-2 V1.1.1, "Mobile Edge Computing (MEC); Mobile Edge Management; Part 2: Application lifecycle, rules and requirements management" (2017-07)

[7] ETSI GS NFV-IFA 011 V4.1.1, "Network Functions Virtualisation (NFV) Release 4; ETSI, Zero-touch network and Service Management (ZSM); Reference Architecture, GS ZSM 002 - V1.1.1, August 2019.

[8] 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.

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

[10] GSM Association, Official Document NG.116 - Generic Network Slice Template, 2019

[11] S. D'Oro, L. Bonati, F. Restuccia, M. Polese, M. Zorzi and T. Melodia, "Sl-EDGE: Network Slicing at the Edge", ACM Mobihoc 2020

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

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

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

[15] TM Forum, ODA Production Implementation Guidelines, GB999, Version 4.0.1, 2020

[16] Geographic Location API REST Specification, TM Forum Specification, TMF675, Release 17.5.1, May 2018.

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

[18] Decentralized Identifiers (DIDs) v1.0. Drummond Reed; Manu Sporny; Markus Sabadello; Dave Longley; Christopher Allen. W3C. 28 July 2020. W3C Working Draft. Available online: https://www.w3.org/TR/did-core/

[19] 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/

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

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

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

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

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

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

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

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

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

[29] Trust Over IP Foundation. Available online: https://trustoverip.org/

[30] Cordentity. Corda Marketplace. Available online: https://marketplace.r3.com/solutions/cordentity

[31] SOFIE (Secure Open Federation for Internet Everywhere)**.** Available online: https://www.sofie-iot.eu/en

[32] PyDentity Project. Available online: https://github.com/OpenMined/PyDentity

[33] TrustID Module. Available online: https://trustos.readthedocs.io/en/latest/id.html

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

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

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

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

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

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

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

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

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

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

[44] Agreement Management API REST Specification, TM Forum Specification, TMF651, Release 19.0, July 2019.

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

10 Abbreviations

AIOps	Artificial Intelligence for IT operations
CNF	Cloud Native Function
DID	Distributed Identifier
DIF	Decentralised Identity Foundation
DLT	Distributed Ledger Technology
DPKI	Decentralised Public Key Infrastructure
EC	European Commission
FaaS	Function as a Service
ISSM	Intelligent Slice and Service Manager
K8s	Kubernetes
LCM	LifeCycle Management
MANO	Management and Orchestration
MEC	Mobile Edge Computing
NBI	Northbound Interface
NFV	Networks Function Virtualization
NFVI	Networks Function Virtualization Infrastructure
NFVO	Networks Function Virtualization Orchestrator
NPM	Node Package Manager
NS	Network Service or Network Slice depending on the context
NSM	Network Service Mesh
POP	Product-Offering Price
RAN	Radio Access Network
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>