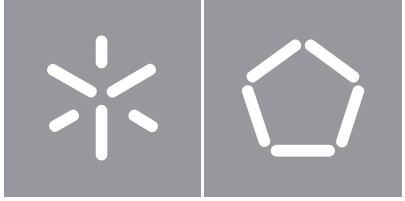
University of Minho
School of Engineering

Duarte Nuno Pereira Moreira

6G - Trust Controller

october 2024



University of Minho
School of Engineering

Duarte Nuno Pereira Moreira

6G - Trust Controller

Masters Dissertation
Master's in Informatics Engineering

Dissertation supervised by
António Luís Duarte Costa

october 2024

Copyright and Terms of Use for Third Party Work

License granted to users of this work:

Acknowledgements

Write your acknowledgements here. Do not forget to mention the projects and grants that you have benefited from while doing your research, if any. Ask your supervisor about the specific textual format to use. (Funding agencies are quite strict about this.)

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, Braga, october 2024

Duarte Nuno Pereira Moreira

Abstract

With the continuous evolving scenario of 6G networks, characterized by ubiquitous connectivity, decentralized artificial intelligence, and augmented service offerings, the need for robust trust management mechanisms is essential. This thesis addresses the challenges of ensuring trust and security in a multi-stakeholder environment by proposing and implementing a Trust Controller architecture. The Trust Controller operates within a broader Trust Manager framework, enforcing trust policies, evaluating the credibility of various network operators and services, and mitigating risks associated with the decentralized and open nature of future networks.

The suggested solution makes use of blockchain technology to develop a reputation-based trust management system that is transparent and resilient to single points of failure. This approach makes it easier for providers and their services to be deployed securely in 6G networks by determining their credibility based on a set of trust characteristics.

This thesis presents a detailed architecture of the Trust Controller, including integration modules for external sources management, and demonstrates its functionality through practical use cases. These examples illustrate the workflow from service activation to trust level calculation and blockchain registration, highlighting the effectiveness of the system.

Apart from the principal contributions, this work also describes possible future improvements that could be made, like enhancing the algorithm used to calculate the trust level and fine-tuning the integration with the Domain Controller entity to improve the collection of metrics events. The results and interactions presented in this thesis provide a comprehensive view of the system's capabilities and try to open the way for future research and development in trust management for 6G networks.

Keywords 6G networks, multi-provider, network slices, trust management, blockchain, trust controller, trust metrics, trust evaluation

Resumo

Com o cenário em constante evolução das redes 6G, caracterizado por uma conectividade ubíqua, inteligência artificial descentralizada e ofertas de serviços aumentadas, a necessidade de mecanismos robustos de gestão de confiança é essencial. Esta tese aborda os desafios de garantir confiança e segurança em um ambiente de múltiplas partes interessadas, propondo e implementando uma arquitetura de um Controlador de Confiança. Este Controlador de Confiança opera dentro de um quadro mais amplo de um Gestor de Confiança, aplicando políticas de confiança, avaliando a credibilidade de vários operadores de rede e serviços, e mitigando os riscos associados à natureza descentralizada e aberta das redes futuras.

A solução sugerida utiliza a tecnologia blockchain para desenvolver um sistema de gestão de confiança baseado em reputação que é transparente e resiliente a pontos únicos de falha. Esta abordagem facilita a implantação segura de provedores e dos seus serviços em redes 6G, determinando a sua credibilidade com base num conjunto de características de confiança.

Esta tese apresenta uma arquitetura detalhada do Controlador de Confiança, incluindo módulos de integração para gestão de fontes externas, e demonstra a sua funcionalidade através de casos práticos. Esses exemplos ilustram o fluxo de trabalho desde a ativação do serviço até o cálculo do nível de confiança e o seu registo na blockchain, destacando a eficácia do sistema.

Além das contribuições principais, este trabalho também descreve possíveis melhorias futuras, como o aperfeiçoamento do algoritmo utilizado para calcular o nível de confiança e o ajuste da integração com a entidade Controlador de Domínio para melhorar a recolha de eventos de métricas. Os resultados e interações apresentados nesta tese fornecem uma visão abrangente das capacidades do sistema e procuram abrir caminho para futuras pesquisas e desenvolvimentos na gestão de confiança para redes 6G.

Palavras-chave redes 6G, multi-provedores, fatias de rede, gestão de confiança, blockchain, controlador de confiança, métricas de confiança, avaliação de confiança

Contents

List of Figures

List of Tables

Acronyms

5G Fifth Generation.

6G Sixth Generation.

ACID Atomicity, Consistency, Isolation, and Durability.

AI Artificial Intelligence.

API Application Programming Interface.

AR Augmented Reality.

CTTC Centro Tecnológico de Telecomunicaciones de Cataluña.

DAG Directed Acyclic Graph.

DC Domain Controller.

DLT Distributed Ledger Tecnology.

E2E End-to-End.

eMBB Enhanced Mobile Broadband.

ER Entity-Relationship.

ETSI European Telecommunications Standards Institute.

IDE Integrated Development Environment.

IoT Internet Of Things.

ITU International Telecommunication Union.

JSON JavaScript Object Notation.

LoT Level Of Trust.

ML Machine Learning.

mMTC Massive Machine-Type Communications.

NSF Network Security Funciton.

NSI Network Security Instance.

NSSO Network Slicing Security Orchestrator.

OV Objective Value.

P2P Peer-to-Peer.

PDL Permissioned Distributed Ledger.

PII Personally Identifiable Information.

PSM Permissioned Distributed Ledger Service Manager.

QoS Quality Of Service.

RAN Radio Access Network.

REK Reputation-Experience-Knowledge.

REST Representational State Transfer.

SC Smart Contract.

SLA Service Level Agreement.

SO Security Orchestrator.

SOTA State of the Art.

SP Security Probes.

TBps Terabyte Per Second.

TC Trust Controller.

THz Terahertz.

TLA Trust Level Agreement.

TM Trust Manager.

TMP Trust Metric Processor.

TOM Trust Objective Metrics.

TPS Trust Provider Score.

TR Trusted Risk.

TSLA Trust Service Level Agreement.

TSLAP Trust Service Level Agreement & Policies.

TSM Trust Subjective Metrics.

TSS Trust Service Score.

URLLC Ultra-Reliable and Low Latency Communications.

VR Virtual Reality.

XR Extended Reality.

ZSM Zero-touch network Service Management.

Chapter 1

Introduction

This chapter serves as an introductory framework, offering a contextualization of the central problem within the fields of Sixth Generation (6G) Networks, Trust, and Blockchain. Furthermore, it points out the core motivations that gave rise to this dissertation. Following this preliminary exploration, the objectives to be realized by the end of this research are outlined. Then, the research methodology and approach adopted in this project are detailed. Lastly, this chapter delineates the comprehensive structure that will be observed throughout the course of the dissertation, offering a preview of the content and its organization.

1.1 Context and Motivation

As the field of 6G networks continues to expand and embrace open and disaggregated approaches[1], trust management assumes a pivotal role in safeguarding the security and service quality delivered by a multitude of stakeholders. Traditionally, trust in suppliers predominantly depends on their established reputation. However, in the realm of multi-vendor environments, dependence on reputation alone encounters formidable scalability challenges.

In response to this pressing issue, Distributed Ledger Tecnology (DLT)[2], specifically Blockchain [3][4], emerges as a promising remedy to establish a robust framework of trust within the telecommunications networks featuring multiple contributors[5][6]. This approach offers the prospect of creating a transparent and dependable system where unalterable records of transactions and contractual agreements find a secure repository. This not only increases reliability but also simplifies the management of trust within the complex domain of multi-party collaboration.

The development of a 6G - Trust Controller (TC) assumes critical significance as it seeks to explore the potential of Blockchain technology in the governance of trust in the highly dynamic and heterogeneous terrain of 6G networks. This dissertation consists on an investigative journey and the practical implementation of this pioneering solution, with the objective of contributing to the safe and efficient evolution of

next-generation networks.

1.2 Objectives

Within the scope of this project, the core activities are directed toward the design, development and validation of a TC. This TC's primary function is to compute and oversee the trustworthiness of service providers actively engaged in the 6G network. Notably, it makes use of Blockchain technology to ensure decentralization, immutability, transparency, and verifiability of trust-related data. The main objective is to establish a system capable of promptly furnishing precise information concerning the most dependable suppliers for meeting specific service requisites within the 6G network.

To achieve this objective, the key actions encompass the implementation of trust calculation algorithms and seamless integration with external sources, especially the Blockchain to ensure secure data storage.

The outcomes of this thesis encompass the development of a fully functional and efficient TC. This TC is poised to enhance the quality and security of 6G services by delivering accurate insights into the reliability of service providers. Additionally, it is expected to foster decentralization and transparency in the realm of trust management, thereby making a substantial contribution to the successful evolution of 6G networks within the multifaceted landscape of multiple vendors. This project assumes a pivotal role in enabling dependable and efficient services in the forthcoming era of telecommunications.

1.3 Methodology

In line with standard dissertation methodology, the research process started with an exhaustive literature review, focusing on the central themes of the study. The primary objective of this phase was to identify pertinent articles that could provide valuable insights for the advancement of the project. To facilitate this effort, the resources of the Google Scholar search engine were used, as well as the knowledge housed at the Centro Tecnológico de Telecomunicaciones de Cataluña (CTTC). The exploration primarily encompassed the following leading databases: IEEE Xplore, ACM Digital Library arXiv, Springer, and ResearchGate.

To ensure the rigor of the search, a systematic approach was adopted. The shearch was limited to articles published within the preceding five years and employed specific keywords, including but not limited to: "trust controller," "trust manager," "5G and beyond networks," "6G networks," "blockchain," and "DLT." Furthermore, citations presented on these papers were too considered important targets to analyse.

Subsequent to this comprehensive literature review, the research workflow transitioned into the stages

of prototype design, implementation, and rigorous testing. The final phase of the methodology involved a critical analysis of the outcomes derived from the project.

1.4 Summary of Contributions

This work has received recognition and has resulted in some contributions in the field of multi-provider 6G network trust management over the past year. To emphasize the potential significance and influence of this dissertation, a summary of their contributions is provided in this section:

- Paper published resulting from this dissertation (D.Moreira): [7]
- Paper published in collaboration with other members of the research group (P.Alemany): [8]
- Trust Controller Implementation (software component)

Firstly, a paper that proposes an architecture for the TC was published, detailing a crucial component intended to monitor trust metrics and guarantee the dependability of service providers. In the course of this thesis, this architecture was implemented and tested, proving its usefulness.

Additionally, contributions were made to another paper, authored and published by the CTTC team, where the Trust Manager (TM) is fully described. The TC is included in this abstract entity, which also incorporates it into a more comprehensive architecture for security and trust management in open and fragmented 6G networks.

These articles highlight the work and acknowledgement within the scientific community in addition to providing confirmation of the developed work. This thesis expands on the information presented in these articles and attempts to advance the State of the Art (SOTA) in trust management for next-generation networks by describing the theoretical foundations and practical implementation of the TC.

1.5 Thesis Structure

This document is structured into seven chapters, with each chapter adhering to a specific framework.

In the first chapter Introduction, is provided an in-depth introduction to the project. Its defined the project's context and motivation, pointed out the primary objectives to achieve, elaborated the research methodology adopted, and offered a detailed overview of the dissertation's structure.

The second chapter State of the Art serves as the foundation for the dissertation. It starts by elucidating fundamental concepts and topics essential for comprehending the project. Although not exhaustive, this

chapter provides adequate knowledge of the major concepts in the area, necessary to support the rest of the dissertation. Furthermore, this chapter encompasses the wealth of knowledge acquired through extensive research within the project's domain, synthesizing and presenting the insights gathered from prior work.

The third chapter, Trust Manager, introduces the abstract component within which the TC is embedded. It provides an overarching view of the project by describing the main workflows involving this component.

The fourth chapter, Internal Architecture, focuses on presenting the detailed constitution of the internal components of the TC and its overall architecture. This includes integrations, data models, data objects, and other key elements.

The fifth chapter, Implementation, delves into the methodology adopted during the project's development and the technological choices made, providing insights into the planning and organizational aspects. It also details the practical realization of the conceptual framework presented. It offers a comprehensive overview of how theoretical concepts were translated into a functional system, detailing code organization, development methodologies, and critical decisions made during the implementation.

The sixth chapter, Tests and Results , describes the rigorous testing processes employed to validate the functionality of the developed components. It presents an overview of the testing methodologies, test cases, and corresponding results obtained during the evaluation phase.

The seventh and last chapter, Conclusion and Future Work, will provide concluding remarks on the project's outcomes and propose potential improvements for future research and development.

Chapter 2

State of the Art

In this chapter, we will delve into a comprehensive and detailed analysis of various fundamental concepts and topics essential for understanding the entire project. Initially, we will explore the notion of 6G networks, which serves as the foundation for the TC. Following that, we will define the concept of Trust to gain insights into its objective and subjective aspects. The topic of Blockchain will also be examined since it forms the basis for the TC's database. And after that, we have a glance and explore the architecture inherent to the TM. Lastly, we will present some related studies that have, in various ways, attempted the development of a TM or Trust Model.

2.1 6G Networks

6G[9][10] is the successor to Fifth Generation (5G) telecommunications technology. Although 5G is still being commercialized over the world and 6G is not yet a functioning technology it's important to note that major infrastructure companies, such as, Huawei, Nokia and Samsung [11] have already committed to the development of this next generation network and it's safe to say that this topic is currently on an early concept research phase, being the deployment and commercial phase schedule to something arround 2030.

According to [12], this new era will consist on the digital, physical and human world seamlessly fuse to trigger extrasensory experiences. Intelligent knowledge systems will be combined with robust computation capabilities to make humans endlessly more efficient and redefine how we live, work and take care of the planet. In [13], authors share the idea that 6G will build on, and extend beyond, our existing 5G ecosystem to foster new innovations which deliver value to customers and simplify network operation. Concurrent to this journey towards the 6G era is the development of network disaggregation and an open, interoperable cloud native architecture.

However, these evolutions do not happen just because they do, but because there are limitations

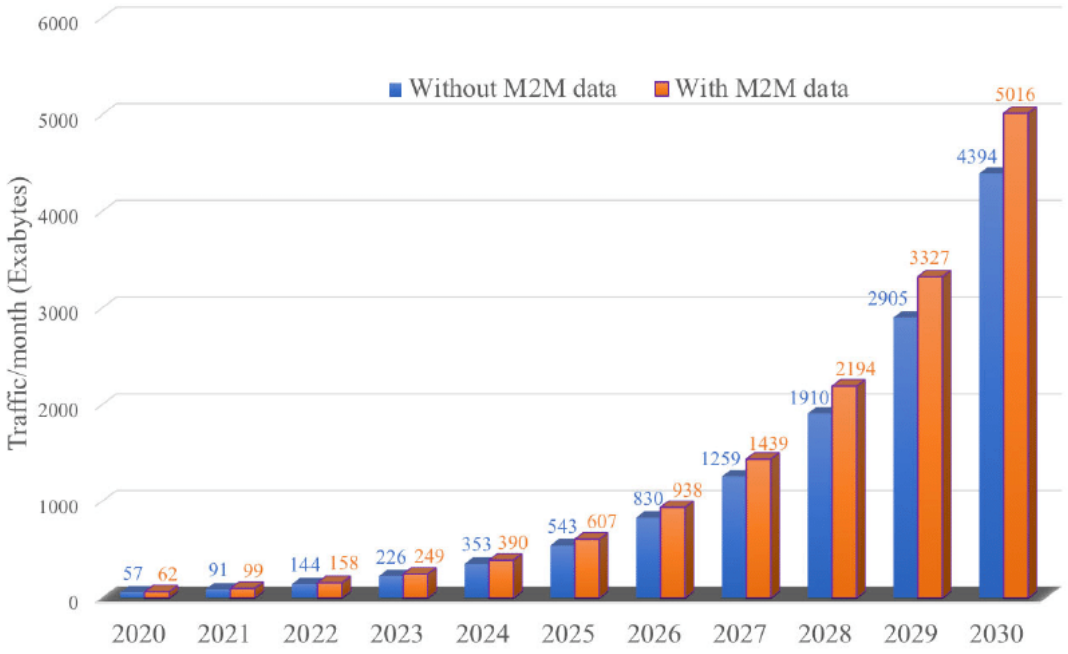


Figure 1: Global mobile data traffic forecast by ITU. Overall mobile data traffic is estimated to grow at an annual rate of around 55 percent in 2020–2030 to reach 607 exabytes (EB) in 2025 and 5,016 EB in 2030. Source: [9]

to what each network generation can support, and in this case, novel service requirements and scale increases are the driving force behind the evolution of the 5G wireless network. The rapid development of emerging applications results, for example, in a never-ending growth in mobile data traffic. According to the forecast by International Telecommunication Union (ITU), global mobile data traffic will reach 5 zettabytes by 2030 [9], as shown in Fig. 1. Upcoming applications (e.g. e-health and autonomous driving) have more stringent requirements for latency and throughput, which will eventually exceed the limits of 5G networks, and so, putting constraints on the 5G communication network. It is expected that 5G will reach its limits in a decade or so and to meet these demands, the main technical objectives for 6G networks, according to [10], will be:

- Ultra-high data rate (up to 1 Terabyte Per Second (TBps)) and ultra-low latency.
- High energy efficiency for resource-constrained devices.
- Ubiquitous global network coverage.
- Trusted and intelligent connectivity across the whole network.

2.1.1 Comparison between 5G and 6G

So far, over the last few years, the world has undergone tremendous developments in information and communication technologies. And the first technological developments related to 5G networks mainly targeted three generic and distinct use cases [14]:

- Ultra-Reliable and Low Latency Communications (URLLC): ultra-reliable means reliability of up to 99.999% and low latency means latency in low single-digit milliseconds.
- Enhanced Mobile Broadband (eMBB): concept that focuses on speed, capacity and mobility for new mobile uses such as high-definition video streaming and immersive Augmented Reality (AR) and Virtual Reality (VR) on the go.
- Massive Machine-Type Communications (mMTC): concept based on connecting large numbers of devices in a given area (up to 1 million devices per square kilometer) that have low data rate requirements and low energy consumption.

As mentioned by [15], 5G network technology came to provide a high standard infrastructure enabling a variety of technologies such as: self-driving cars, Artificial Intelligence (AI), mobile broadband communication, Internet Of Things (IoT) and smart cities. Looking at the past, as highlighted in Fig. 2, it's clear that each generation optimizes the use cases of the previous generation and introduces new ones. This will continue to be the case. 6G will build on top of 5G in terms of many of the technological and use case aspects, driving their adoption at scale through optimization and cost-reduction. At the same time, 6G will enable new use cases[12].

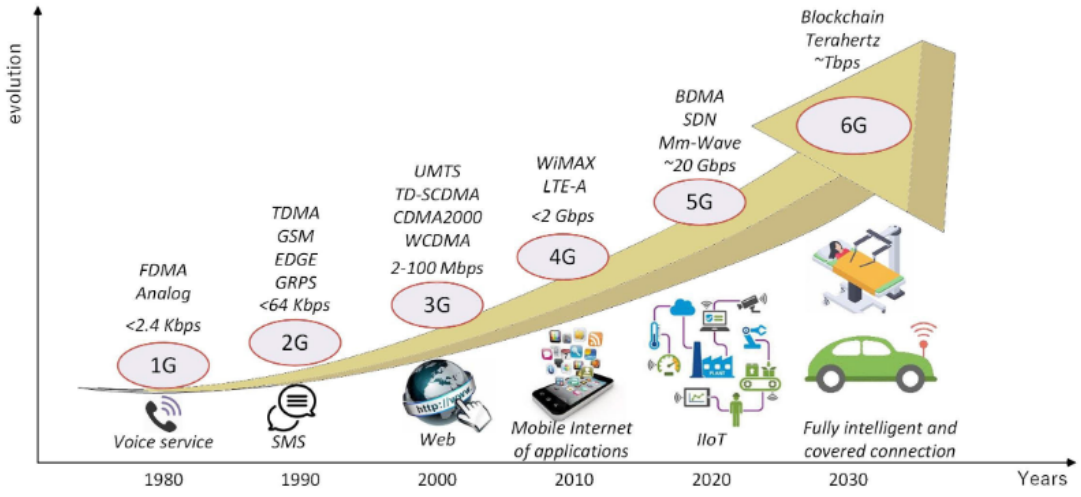


Figure 2: Evolution of mobile wireless systems. Source: [10]

To have an idea of what this generation change will consist of, lets us compare the main specifications and technologies in both 5G and 6G presented on Table 1. As we can see this jump will open the door for a new communication system providing more capacity, extremely low latency, high data transmission, secure error-free communication and full wireless coverage. More can be found on [15] work.

Table 1: Comparison of 5G and 6G Characteristics

Characteristic	**5G**	**6G**
Operating frequency	3 - 300 GHz	up to 1 THz
Uplink data rate	10 Gbps	1 Tbps
Downlink data rate	20 Gbps	1 Tbps
Reliability	10^{-5}	10^{-9}
Maximum mobility	500 km/h	1000 km/h
Processing delay	100 ns	10 ns
Traffic capacity	10 Mbps/m^2	1 - 10 Gbps/m^2
Localization precision	10 cm on 2D	1 cm on 3D
Time buffer	Not real-time	Real-time
Center of gravity	user	service
Satellite integration	No	Fully
AI integration	Partially	Fully
Automation integration	Partially	Fully

With this values in mind, it is expected that 6G can possibly overcome 5G and improve the network performance, integrate different technologies and increase the Quality Of Service (QoS) providing super-smart society with everything connected to the network.

2.1.2 6G Network Vision

As mentioned previously, the number of connected devices to the network, as well as the traffic are both growing and increasing rapidly, which ultimately increases the amount of data that flows through the system. Besides that, the areas of usability of the network are growing exponentially too.

The system of the future, i.e., 6G, is expected to provide seamless and energy-efficient connectivity around the globe, which is almost impossible through current network architecture [16], therefore, a new one needs to be re-designed and built. As already said, 6G is scheduled to launch commercially in 2030, and currently, it is in the pre-development stage, so it is unrealistic to illustrate the architecture of 6G accurately. But despite that, Fig. 3 can show us the known components of 6G in its schematic architecture.

For better understanding of this new technology and its components (Fig. 3), we can subdivide it into three major parts Network Coverage, Network Capabilities, and Service Provisioning.

- Network Coverage: network coverage is defined as the area in which you have access to the respective network and this area is directly proportional to usability, that is, if we connect to more subscribers, then greater network coverage is necessary for effective and efficient network usability[10]. As we already know, the coverage provided by the current network is quite extensive at terrestrial level, allowing access in several urban regions, but both the usability and coverage are expected to increase by many folds with the futuristic 6G networks. Despite that, this network is limited as it doesn't allow for it's own expansion under sea and to high altitudes such as in planes and satellites. According to [17], future use case scenarios and applications will demand that the network can provide network access under the deep sea and at high altitudes. Clearly that along with this, the future 6G network is expected to improve connectivity at a territorial level as well.

- Network Capabilities: as has been mentioned, the 5G network will probably reach his limits and not be able to support the demand of modern and future services, that said, the next generation network should present a capability many times stronger than the current. To categorize the capabilities of a network we can look at some major factors such as: transmission rate, frequency band management, resource management, and efficient use of underlying network infrastructure for modern applications. In Fig. 3, we can see the major supporting features of 6G that will encourage the development of modern applications: Terahertz (THz) network speed, multiple frequency band support, cell-less architecture, intelligent Radio, Distributed AI, Real-Time edge, and Virtual Network, Virtual Storage, and Virtual Compute. These are more detailed and explored on [16].

- Service Provisioning: service provisioning can be defined as the act of setting up a particular service for an end-user. And 6G networks have already aroused the interest of several researchers in the creation of some advanced services. This will encourage the new generation of developers to construct advanced services by providing them underlying network features. Some of these advanced services that are in demand in the near future and that are going to be supported by the 6G network are: Smart Healthcare, Autonomous Transportation System, Smart Security Systems, Extended Reality (XR) and Quantum Processing [16].

Another very important and adjacent subject of 6G wireless communications will be its areas of application and the ability to improve and transform existing areas. These proposed applications will reduce deployment costs and increase the flexibility of our communication technology [10]. Some examples of

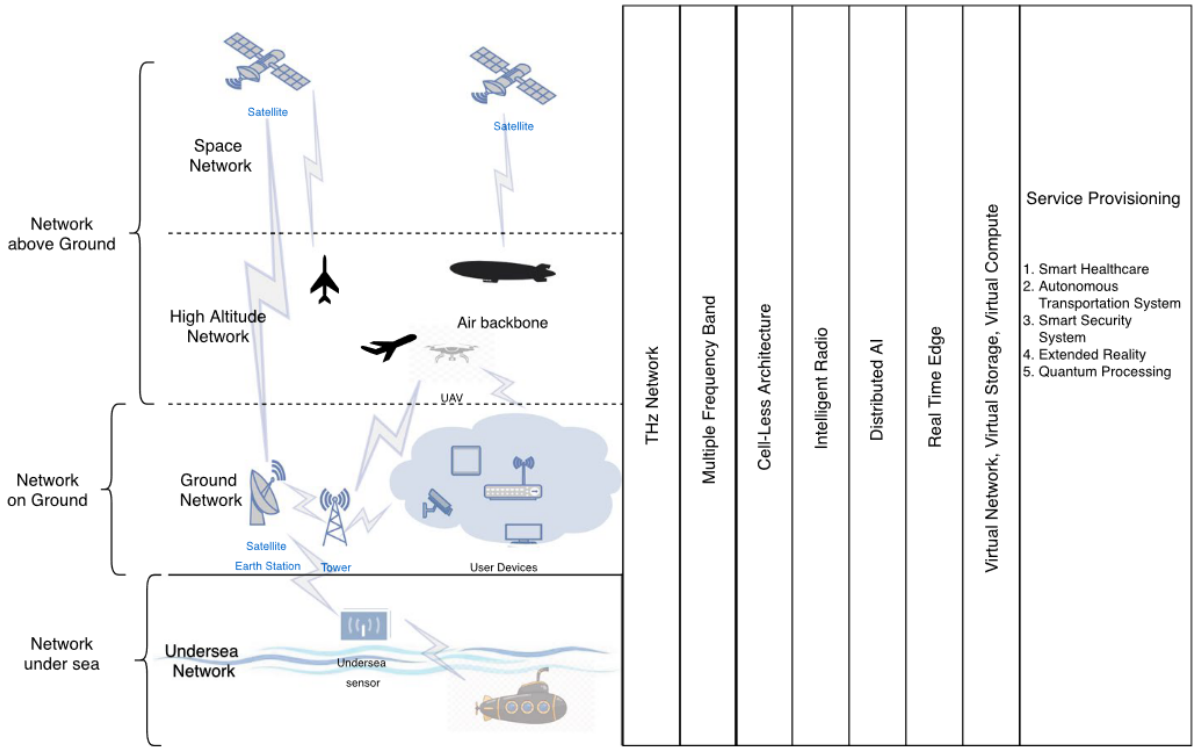


Figure 3: 6G technology architecture. Source: [16]

these applications are: Smart Society, Smart Healthcare, Wireless Brain–Computer Interactions, Connected Robotics and Autonomous Systems, Nervous System Information Transfer, Manufacturing and Its Automation, Blockchain and Distributed Ledger, Multi-sensory XR Applications, etc, [16]. As one of them, Blockchain and Distributed Ledger, meets the purpose of this work, a brief introduction is presented:

A blockchain is a list of records called blocks that are closely linked to each other using cryptography [10]. Based on 6G wireless communication, a blockchain is resistive in updating the information. It is a type of distributed ledger that is used to record the whole process between the two users and transfers the transaction values between them, without the requirement of any backbone like central coordination. It is based on an electronic ledger that keeps the complete data record, and it will be transmitted using a 6G wireless communication system [16].

2.2 Trust

In the giant landscape of advanced networking, the concept of trust can be considered a keystone, moving through the complex web of connections and relationships in the evolving scenario of 6G technology. This section tries to explain and clarify the abstract and subjective concept of trust in this specific realm.

From a foundational exploration of the very essence of trust and its definition to its dynamic role in a multi-stakeholder scenario, following into the metrics and calculations that underpin this critical element. Moreover, the concept of Trust Service Level Agreement (TSLA) its mentioned, as it is expected to be a important component, since its objective is to align trust objectives with service expectations.

One of the focal point of this project is succinctly encapsulated in the quest for a singular, quantifiable measure of trust — a mathematical value that transforms the subjective into the objective. As the ultimate aim, this value should represent trustworthiness as a percentage that goes from 0% (i.e., not trustworthy) to 100% (i.e., trustworthy), offering clarity to stakeholders, especially clients. In essence, it seeks to empower decision-makers to measure the risk associated with trusting a service provider, facilitating informed choices in a landscape full of choices and alternatives.

2.2.1 Trust definition

In human communities, the unpredictability of strangers behavior introduces uncertainty, leading individuals to steer clear of interactions with those they do not trust. Trust plays a pivotal role in facilitating interactions within such uncertain environments. This concept extends to the realm of technology, emphasizing that trust is a vital factor for successful online interactions, as reinforced in [18], and should be a key criterion for service selection. Trust seems to emerge as the most intricate bond between entities, characterized by its highly abstract, volatile nature, and the formidable challenges associated with its measurement and management. Yet anyway, as [5] work highlights it becomes necessary to have a transparent and reliable way for service clients to evaluate and identify whether a provider is trustworthy enough to participate in a service deployment. This is a complex decision to make due to the difficulty of bringing a subjective concept such as "trust" into an objective field such as network management.

Trust is a concept that has been considered as a key foundation for decision making in different areas and computer science is one of them. Although is definition varies, in all cases the same idea is shared: trust is a relationship in which an entity, often called the trustor, depends on someone or something, called the trustee, based on a given criterion [19], and for this case its value can be measured by trust metrics. The calculated value of trust is mentioned as Level Of Trust (LoT) throughout this project.

Despite this definition being relatively close to expected, [20] work also presents some interesting correlation definitions worth mentioning:

- Definition 1: Trust is referred to the recognition of entity's identity and the confidence on its behaviors. Trust is subjective behavior since entity's judgement is usually based on its own experiences. Trust is described by trust value.

- Definition 2: Trust value or trust degree is used to measure the degree of trust. Trust value often depends on special time and special context.
- Definition 3: Direct trust means trust that is obtained by entities' direct interaction.
- Definition 4: Indirect trust or recommended trust means trust that is obtained from credible third party who has direct contact with the designated one. Recommended trust is one important way to obtain trust degree of unknown entities.

2.2.2 Trust in a Multi-Stakeholder Scenario

As been claimed until now, 6G ecosystems will grow in such manner caused by the increasing virtualization and disaggregation of networks that the proliferation of connected devices and diverse applications will be significant. This trend leads to a multi-vendor situation appearing in the telco arena (i.e., hardware suppliers, software function providers, vendors, operators) that communications service providers will need to deal with. This multi-stakeholder scenario complicates provisioning and the established relationships among actors based on operation agreements - Service Level Agreement (SLA) - that define the requirements, penalties, and prices that generate a certain level of trust among them[5]. That said, the 6G threat vector will be defined by 6G architectural disaggregation, open interfaces and an environment with multiple stakeholders. And, as stated in [19] and [21], establishing trust in such an open and diverse ecosystem is a cornerstone for a global adoption of the technology and should be assured across devices, sub-networks, heterogeneous-cloud, applications and services.

As multiple entities with different interests and responsibilities participate in a common network, building a foundation of trust becomes instrumental in achieving collective goals and maintaining a peaceful and efficient ecosystem. This bridge can be expected to bring benefits such as: enhanced communication, risk mitigation, efficiency and productivity, long-term relationships, positive reputation, among others.

2.2.3 Trust Metrics and Trust Calculation

In [20] a trust classification method is presented. Authors mention that trust can be divided into different categories according to different standards:

- According to attributes: identity trust and behavior trust
- According to obtaining way: direct trust and recommended trust
- According to role: code trust, third party trust and execution trust, etc

- According to based theory: subjective trust and objective trust

The project associated with this work follows the last mentioned classification where the basis of a trust value comes from the combination of objective and subjective trusts. This implies the existence of objective and subjective trust metrics. The objective ones are defined as quantifiable and measurable data that is not influenced by personal feelings, interpretations, or prejudice. These metrics are based on factual data and are commonly used for making informed decisions, assessing performance, or benchmarking against standards. As outlined in [22], examples of such metrics include response time, latency, execution time, throughput, reliability, domain-specific measurable properties, and other services' measurable properties. On the other hand, subjective metrics are referent to evaluations or measurements influenced by personal feelings, interpretations, or individual perspectives. These metrics are based on personal opinions and perceptions and are often utilized to evaluate user satisfaction, personal preferences, or qualitative experiences. Examples can include user satisfaction ratings, perceived QoS, user reviews, and subjective evaluations.

Regarding the calculation of the trust value - or level of trust - the methods currently used, among the few that currently exist, are diverse and depend greatly on the type of trust model adopted and in turn, on the type of associated metrics. Take as an example the work [23], where the authors opt for an implementation based on the threefold Reputation-Experience-Knowledge (REK) model that presents, as the name suggests, three main attributes:

- Knowledge: trustor's general understanding about the trustee
- Experience: trustor's previous experience with the trustee
- Reputation: public opinion on the trustee

At this point, it is necessary to take into account the application, the environment and the problem in which we find ourselves and, depending on this, establish the most suitable trust metrics for each trust attribute so that an analytical value can be attributed to them, as the authors do, for example, for knowledge: right to decision making; for experience: right to restrict processing; and for reputation: right to be informed. Trust metrics that suited the explored use case related to Personally Identifiable Information (PII)[24], that involves all and any sensible data of a person.

There are numerous methods available to estimate the trust attributes, ranging from numerical methods, probabilistic methods, and belief theory to data analytics methods such as association rule learning, classification tree analysis, genetic algorithms, Machine Learning (ML), sentiment analysis, and social network analysis. For this specific case [23], and in most common scenarios, the mathematical approach

used to find the trust level between trustor i and trustee j, as later leveraged in this dissertation, involves assigning weights to each of the trust metrics, taking into consideration the following factors:

$$K_{ij} = \alpha_1 K_1 + \alpha_2 K_2 + \ldots + \alpha_n K_n \tag{2.1}$$

$$E_{ij} = \beta_1 E_1 + \beta_2 E_2 + \ldots + \beta_n E_n \tag{2.2}$$

$$R_{ij} = \gamma_1 R_1 + \gamma_2 R_2 + \ldots + \gamma_n R_n \tag{2.3}$$

$$\text{Trust}_{ij} = \theta_1 K_{ij} + \theta_2 E_{ij} + \theta_n R_{ij} \tag{2.4}$$

Where α,β,γ, and θ, are weighting factors that normalize each metric in between 0 and 1. K_x, E_x and R_x represent the trust attributes of Knowledge, Experience, and Reputation, respectively. And n represents the total number of trust metrics for each trust attribute.

2.2.4 Trust Service Level Agreement

A TSLA its a component that plays a crucial role in the management of 6G networks and its said to be a SLA based on trust parameters or, in other words, trust metrics. Following the idea presented on [5], when a client seeks a service, they must specify the trust-related criteria that should be met for selecting the most suitable provider. The idea of TSLA has evolved from the traditional SLA concept. Like mentioned before, in TSLA the focus is on trust requirements, incorporating factors like reputation or trust metrics values associated with various actions and the minimum level of trust expected by the client from service providers. This innovative approach ensures that trust considerations are integrated into the agreement, outlining the specific expectations and standards for trustworthy service delivery.

Moreover, TSLAs contribute to the transparency and clarity of trust-related interactions, allowing for informed decision-making in selecting service providers. By establishing a clear framework with defined trust metrics, TSLAs simplify the complex process of evaluating and comparing providers, ultimately enhancing the trustworthiness of the entire 6G network.

2.3 Distributed Ledger Technology

Before talking about blockchain, we can take a step back, and explore the concept of DLT. DLT's can be defined as a form of technology used to distribute, exchange, or store data among users via public or private networks. Basically, it is a database that is stored and located across many nodes situated at different geographic locations. Each computer in the network is known as a node. Distributed ledgers can also be thought of as communal datasheets maintained on several distributed nodes[25]. This technology is centered on distributed systems, meaning that, unlike traditional databases, distributed ledgers have no central data store or administration functionality, as presented in Fig. 4.

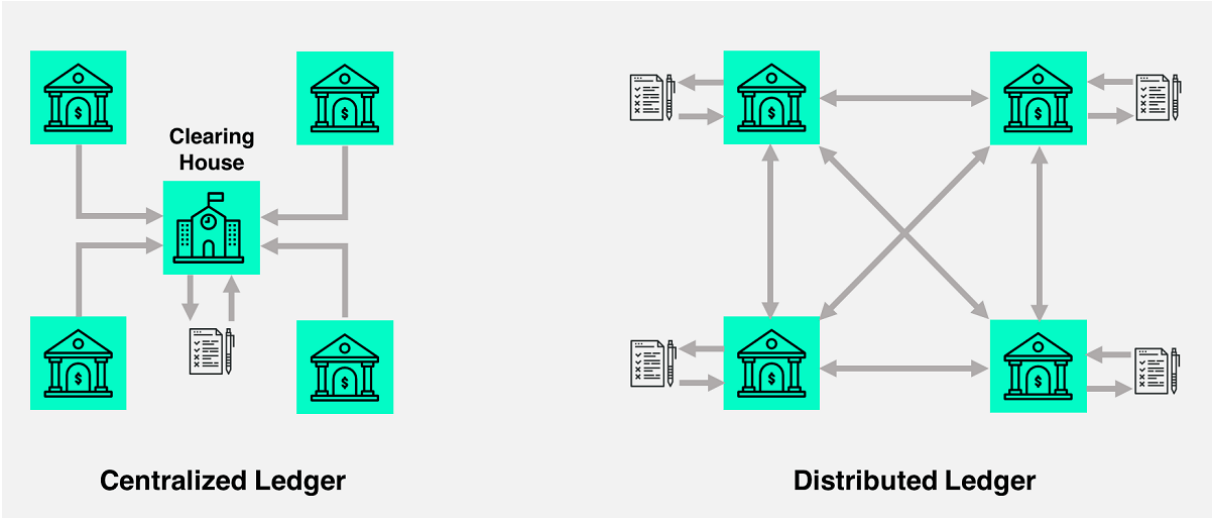


Figure 4: Centralized and Distributed Ledger comparison. Source: [26]

One of the areas where distributed ledgers have been considered is IoT. According to [25] innovative technologies like cloud computing and big data have been utilized by the IoT bringing greater availability, accessibility, personalization, precision, and lower-cost delivery of services, and DLT appears to be the likely next step to be converged with IoT and other smart and connected technologies. Its ability to build trust through distributed networks without requiring a third party is an advancement that may alter multiple industries, by potentially fixing major problems and addressing weaknesses and vulnerabilities of today 's client/server cloud IoT models. For instance, as mentioned in [27], cloud servers can be disabled by software bugs, cyber attacks, or other mechanical issues. In contrast, IoT systems that utilize blockchain technology are not vulnerable to a single point of failure because identical data is maintained on multiple devices and computers.

The same work [27], also points out that recently, with the blooming of 5G, cloud computing technology has given the IoT the ability to analyze data and translate information into real-time action and

up-to-the-minute knowledge. This unparalleled IoT expansion has generated new means of sharing and accessing information, including a foundational open data model. However, one of the main vulnerabilities of these new methods of information sharing is a lack of confidence. While centralized architectures have contributed substantially to IoT growth, when it comes to data transparency, they are opaque and leave participants questioning how their data will be used. That being said, reimagining the way IoT data is handled, would empower users to rely on a decentralized, independent, and robust data management system that guarantees data ownership. Such a system would include, according to [28]:

- Access Control: access is managed using DLT/blockchain-based decentralized, auditable mechanism that ensures data ownership and encrypted information sharing.
- Secure Data Storage: data is maintained in such a way that it is verifiable, immutable, and trusted.
- IoT Compatibility: allows data to be appended by one writer and viewed by multiple readers.

In the world of DLT, diverse categories exist to delineate its diverse applications, from blockchain to Directed Acyclic Graph (DAG)[29]. For the scope of this thesis, the emphasis is aimed towards blockchain technology, specifically focusing on a permissioned blockchain. A Permissioned Distributed Ledger (PDL)[30], in essence, operates on the foundation of restricted access, where participants need to obtain explicit permission to be part of the network. This approach contrasts with public blockchains, offering a more controlled and regulated environment. The permissioned blockchain is particularly suitable for scenarios where privacy, scalability, and trust are important requirements.

2.3.1 Blockchain

Blockchain, first introduced in 2008 through the publication "Bitcoin: A Peer to Peer Electronic Cash System" [31], represents a decentralized and immutable database ledger. According to [25], it comprises a continuously growing series of interconnected records, or blocks, secured through cryptography. This is considered to operate on a Peer-to-Peer (P2P) topology, where participants manage the blockchain network using private-public key pairs. New data records can be added through mining, a process that involves solving a consensus problem via distribution. Each block of the network includes a cryptographic hash of the previous block, a timestamp, and transaction data organized in a Merkle tree[32][33] structure, as shown in the Fig. 5. Once recorded, every transaction data becomes unalterable without modifying all preceding blocks. The blockchain's core unit is a transaction, authenticated by block miners and encrypted into secure blocks. The synchronization across miners occurs regularly, and consensus mechanisms

ensure the ledger maintains the longest chain in case of discrepancies. Merkle trees play a crucial role, providing secure confirmation of data consistency. These structures condense block transactions through hierarchical hashing, creating a digital fingerprint for verification. The resulting Merkle Root represents the ultimate hash of all transactions in a block. This robust framework ensures the integrity, security, and decentralization of the blockchain, making it a foundational technology with wide-ranging applications.

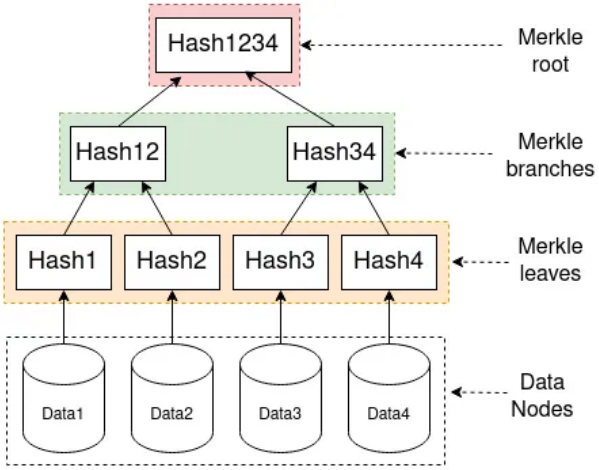


Figure 5: Merkle tree example. Source: [33]

The concept of blockchain, underscores its fundamental role in revolutionizing future mobile communication technologies and addressing critical challenges in the domain of data management and connectivity. In accordance with [10] and as mentioned before, blockchains are characterized as distributed ledger-based databases, providing a secure platform for registering and updating transactions without the need for central intermediaries. The inherent features of decentralization, tamper-resistance, and anonymity make blockchain an ideal candidate for diverse applications, ensuring stronger security features throughout communication processes. Furthermore, [15] highlights the role of blockchain in managing big data and organizing connectivity in 6G, emphasizing its utility in spectrum sharing to address the spectrum requirements of 6G while guaranteeing secure, low-cost, and efficient spectrum utilization.

In a similar way, [9] points that blockchain technology is prepared to play a crucial role in securing and authenticating future communication systems. Decentralization and transparency contribute to faster processing, a critical aspect for 6G systems. The virtualization of network resources in 6G necessitates simultaneous resource allocation and authentication, a role where blockchain is expected to excel. The adaptive nature of blockchain security aligns with the evolving requirements of different scenarios, such as ensuring extreme security and privacy when offering services or enabling quick, controlled access during emergency responses via 6G systems.

In [25] authors recognize IoT as a highly promising technology, yet existing IoT systems often deploy

devices with constrained resources, leaving them susceptible to cyber threats. These networks face challenges in scalability, maintenance complexities, and persistent vulnerabilities related to single points of failure. Furthermore, the impermanence of IoT data and concerns about data security and privacy compound these issues. In response to these challenges, blockchain technology emerges as a viable solution due to its inherent decentralization, data creation and storage methodologies, and robust consensus mechanisms as mentioned in [34][35]. Farahani's work[25], highlights that blockchain can effectively address the limitations of centralized IoT systems. Numerous use cases have demonstrated the applicability of blockchain across all facets of an IoT ecosystem. Its potential applications include authenticating and encrypting data in communication networks, managing and securely storing device identifications, and maintaining the integrity of Cloud data as well as information stored by distributed objects or devices. By leveraging a distributed network of nodes for P2P data exchange, blockchain significantly reduces the risk of data compromise, making it highly resistant to unauthorized access. The consensus mechanism inherent in blockchain technology serves as an additional layer of security by preventing compromised nodes from infiltrating an IoT network and rejecting data from compromised sources, thereby safeguarding data integrity.

2.3.2 Smart Contracts

Smart Contract (SC)s serve as foundational components within distributed ledgers and blockchain systems. As defined by both European Telecommunications Standards Institute (ETSI) [36] and in [25], a SC is a self-executing digital agreement designed to facilitate the seamless transfer of digital assets among participants, subject to specific conditions on the blockchain. In essence, it automates the enforcement and execution of contractual terms when predefined conditions are met. The significance of a SC lies in its ability to deliver transparent execution, permanence, and decentralization, ensuring the secure execution of programmed logic. In essence, a SC is a snippet of code stored on the blockchain, playing a pivotal role in fostering a secure and well-coordinated decentralized system.

As for the benefits that SCs offer when merging IoT with blockchain, these are some of the pointed out in Farahani's work[25]:

- Autonomic Interactions: Governed by SCs, blockchain allows IoT devices or subsystems to interact automatically without human intervention, in which there is no need for any traditional central role, such as governments.

- Contract Execution: SCs, a multiparty agreement stored using blockchain, enables stakeholders

to execute contact arrangements when specific criteria are met. SCs are able to automatically authorize payments when the contractual service requirements have been met.

- Improved Security: Blockchain and SCs can also increase system security by updating device firmware automatically in order to address vulnerabilities.

- Improved Functionality: Blockchain provides greater functionality through the programmable logic of SCs and the ability to handle interactions as transactions. SCs also provide security functions around confidentiality, authentication, and access control to improve IoT security.

2.4 Zero Touch Network & Service Management

The Zero-touch network Service Management (ZSM) is a comprehensive architectural framework created by the ETSI ZSM group in 2017, offering both horizontal and vertical end-to-end capabilities [37] and its the one adopted to the component that will contain the TC developed in this dissertion, the TM.

ZSM serves as a framework designed to facilitate the construction and integration of loosely connected management functions that provide essential management services. These functions collectively deliver specific domain and end-to-end capabilities, enabling intervention-free (zero-touch) management of network and infrastructure services. The goal of ZSM is to achieve highly automated networks driven by high-level policies and rules, allowing networks to have self-configuration, self-monitoring, self-healing, and self-optimization without requiring constant human intervention [38].

The strategic approach involves developing independent modules, each performing specific functions, and exposing endpoints for consumption. This modular design allows for the assembly of various modules to create more specialized services and functionalities for specific use cases [37]. The architecture diagram of this framework is presented in Fig. 6.

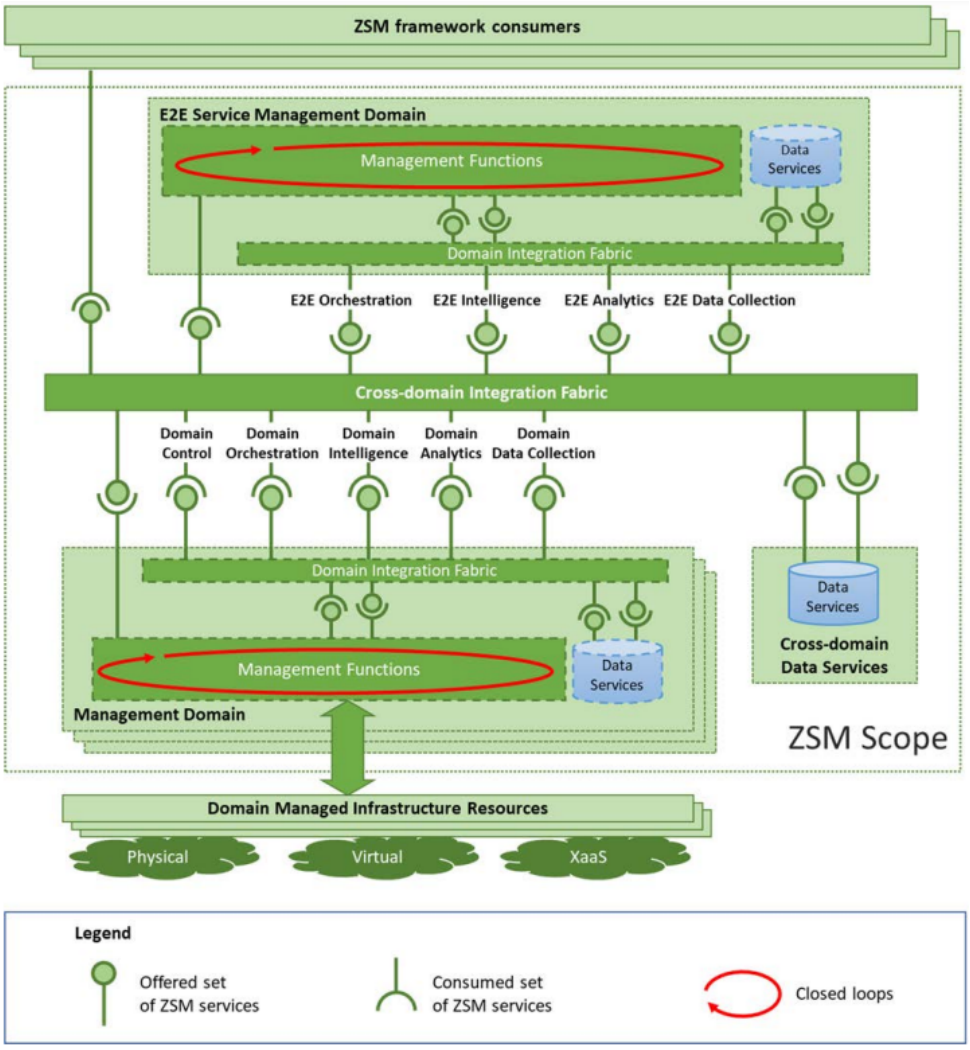


Figure 6: ZSM framework reference architecture. Source: [37]

As we can see the ZSM framework blueprint lays out a set of building blocks that work together to create advanced management services and functions. Essentially, the ZSM framework is made up of scattered management and data services grouped into management domains and connected through an integration fabric. This fabric not only facilitates the use of management services but also supports communication and collaboration with external management systems. Additionally, a cross-domain data service allows sharing information between different areas. As outlined in ETSI's documentation [37], the description and behavior of the primary components within the ZSM architecture are detailed below:

- Management Services: a management service is one of the most important building blocks. They provide capabilities through service end-points, which define their function regarding the entities they manage. These capabilities can be shared among multiple consumers, promoting automation and continuity in interactions between management domains. To deliver their services, service

producers may interact with infrastructure resources directly via their interfaces or indirectly by consuming other services through their end-points.

- Management Functions: management functions are entities that can be labeled as “service producers” if they produce (offer) capabilities of management services, and/or “service consumers” if they consume capabilities of management services.

- Management Domains: management domains are used to partition responsibilities and to create ”separation of concerns” within a given ZSM deployment. Domains provide service capabilities and can consume services from other domains.

- End-to-End (E2E) Service Management Domain: is a special management domain that provides E2E management of customer-facing services, composed from the customer-facing or resource-facing services provided by one or more management domains.

- Integration fabric: allows management functions to communicate within and across management domains. It handles tasks like registering, discovering, and invoking management services and can both provide and consume capabilities for service integration.

- Data services: data services enable consistent means of shared management data access and persistence by authorized consumers across management services within or across management domains. They also eliminate the need for management functions to handle their own data persistence, thereby allowing to separate data persistence from data processing.

2.5 Related Work

This section serves as a comprehensive exploration of some of the current existing studies, works, and insights that lay the groundwork for understanding the role of trust in a network scenario. Through a meticulous examination of related works, the point is to contextualize this research within the broader scope of current advancements, identify gaps, and gather valuable insights that propel us forward in the pursuit of innovative solutions.

2.5.1 Work Foundation

The work in [5], describes one of the most recent works related to the implementation of a Blockchain-based TM and the main foundation for this project. The authors follow the standards from ETSI [39] and

adopt the concept of reputation as the right candidate to approximate a value of trust, since reputation can be measured. So the first step involves describing a set of reputation parameters to compose a value associated with the risk of trusting each vendor called Trusted Risk (TR). Among different possibilities, three actions were selected to obtain the corresponding reputation parameters as follows:

1. Provisioning rate: a metric that assesses the percentage of service requests accepted by a provider out of the total received. For instance, if 9 out of 10 requests are accepted, the provisioning rate is 90%. This metric is crucial for evaluating a provider's readiness to handle concurrent services and indicates the effectiveness of its resource management strategy.

2. Non-forced termination rate: a metric that evaluates the percentage of successfully completed services in relation to the total accepted and deployed services. For example, if a provider successfully terminates 8 out of 9 deployed services, the non-forced termination rate is 88.88%. This metric provides insights into a provider's reputation for fulfilling services as expected and terminating them according to client requirements.

3. SLA mitigation rate: a metric that assesses a provider's effectiveness in responding to SLA violations. It is calculated by determining the percentage of SLA violations successfully mitigated compared to the total number of violations generated throughout a service life cycle. For instance, if a provider mitigates 13 out of 15 SLA violations, the mitigation rate is 86.66%. This metric reflects the provider's reputation for promptly addressing and resolving problems related to SLA violations after deploying the requested service.

The following step involves the process of creating a mathematical expression to compute these parameters and obtain a TR value. The model uses the three specified parameters and sums them all, each with a specific weight (w) to define its individual importance over the overall result. The resulting equation for this specific case is illustrated as follows:

$$\begin{aligned} \text{TR} &= \sum w * R_x \\ &= \alpha R_{prov\_rate} + \beta R_{term\_rate} + \gamma R_{mit\_rate} \\ &= \alpha \frac{S_{deployed}}{S_{requested}} + \beta \frac{S_{terminated}}{S_{deployed}} + \gamma \frac{SLA_{mitigated}}{SLA_{violated}} \end{aligned} \tag{2.5}$$

In the end of this process we get a single percentage value that goes from 0% to 100% (assuming that $\alpha + \beta + \gamma = 1$) used to assign a level of trustworthiness to each provider. Finally the last call its from the blockchain and the SCs that are used to trigger the process to update and manage the trust among

multiple stakeholders. Apart from that, the blockchain network will be the element to keep the record evolution of all reputation values and the TR's associated with each provider because of its transparency strength. The SC funcionalities envolve:

- store and distribute service requests to the correct provider based on TSLA requirements (client demands) and the current reputation and TR values
- to trigger the TR computation after a service is terminated
- to save and share the updated reputation and TR values corresponding to the correct provider

Regarding the internal architecture of the system, it is composed of an abstract entity - TM - which includes three main components: Trust Service Level Agreement & Policies (TSLAP), TC, and Permissioned Distributed Ledger Service Manager (PSM). The first is divided into two subcomponents and is mainly responsible for managing TSLAs and policies. On the other hand, the PSM has the role of controlling all information flow or communications, acting as a gateway, both to and from the blockchain. Lastly, our focus, the TC, is the main component when it comes to calculating a trust value since its function is to manage the process of calculating multiple reputation values to generate a TR associated with each provider. That said, this component comprises three distinct subcomponents:

- Evaluation: it gathers and evaluates which data are going to be used
- Metrics: it makes use of data models and historical logs from different providers to generate reference data to pass to the score computation component
- TR Computation: it processes data coming from the previous two components to obtain the updated and latest reputation values and TR for those providers involved in terminated services

From this point on, the system is ready to receive requests from clients with a set of specific requirements aimed at deploying a particular service. This marks the first of the two main points of the system, with the second being the process of updating reputation values and TR.

2.5.2 Other Works

Apart from this work, several other studies have addressed the challenges of trust management in a multi-stakeholder scenario.

Elmadani *et al.* [1] focus on a similar goal within the context of 5G networks. By adapting an existing technique, AppFlow, to monitor provider interactions as presented in [40], they aim to establish a trustworthy value for providers using a trust indicator known as Evidence. This indicator branches into two fields: direct and indirect evidence, which parallel the objective and subjective metrics discussed in this thesis. Similarly, Niu *et al.* [41] propose a complex and meticulous trust model using the cloud model algorithm. This model explores the subjective side of trust, focusing on evaluating network slices rather than providers and services themselves. However, their work does not leverage decentralized technologies like blockchain. On the other hand, the work presented in [35] leverages blockchain technology to build a trust-reputation management solution aimed at minimizing the impact of managing the larger amount of resources that will be required in next-generation networks. This approach allows nodes to share resources based on their trustworthiness. However, this method differs from the one proposed in this dissertation, which focuses on evaluating the trustworthiness of service providers and the services they offer.

Casola *et al.* [42], despite being an older project, present the design and implementation of a TM called TruMan, which aims to select the best provider based on a set of requirements. However, as pointed out by [5], many of these related works are planned to be on top of the architecture, centralizing processes, what leads to potential bottlenecks and single points of failure[43]. This centralization simplifies the way an entity might gain undue advantage by controlling or tampering with the stored data, as it is placed within a single element. Therefore, removing centralization is crucial, and distributed systems are emerging as a viable solution to these issues.

This chapter describes the main technologies and topcis of interest to the work. The chapter ended with a summary of the related main works, highlighting one that is used as a starting point, which is "Blockchain-based trust management collaborative system for transport multi-stakeholder scenarios"[5]. The next chapter describes the TM component and the workflows that are one of the main goals to successfully achieve with the work developed alongside this dissertation.

Chapter 3

Trust Manager

In this chapter, the aim is to introduce the abstract component in which the TC is embedded, as well as to provide an overarching view of the project at hand describing the main workflows in which this component is involved. This work is further explored and explained in [8].

3.1 Introduction

Typically, in any scenario involving multiple participants or actors, the perception of trust that one entity has in another is often shared among all. For instance, if a person fails to fulfill a certain task, others around them will spread that information, which can be detrimental to the individual's future interactions as others become more cautious and suspicious. It is therefore clear that the concept of trust can be easily influenced by simple information, which leads to the need to establish a transparent and shared way to compute, manage, and distribute trust-related information.

The solution of a TM aims to address this issue, particularly in a scenario where multiple service providers are competing with each other. As illustrated in the following section, Fig. 7 [44], a TM is an entity designed to operate across all network domains, such as Radio Access Network (RAN), Cloud, Transport, Edge, non-terrestrial networks, personal networks, etc., involved in the concept of network slicing. This distribution occurs because this entity is the key element, as mentioned, capable of calculating a trust value based on metrics and transparently distributing it throughout the system, maintaining an immutable record of how reliable providers are in fulfilling their requests. The composition of a TM is presented in the next section.

3.2 Architecture

This architecture is based on the use of the ZSM principles discussed in section 2.4, of which the main ones adopted are:

- Multilevel Domain: an approach that defines the granularity of automation processes, allowing different levels of abstraction and decomposition of network and service functions, from E2E to domain.
- Integration Fabric: a layer that connects all domains, allowing interoperability, data exchange, and orchestration between them.
- Closed Loop: a mechanism that enables continuous monitoring, analysis, and optimization of network and service performance.

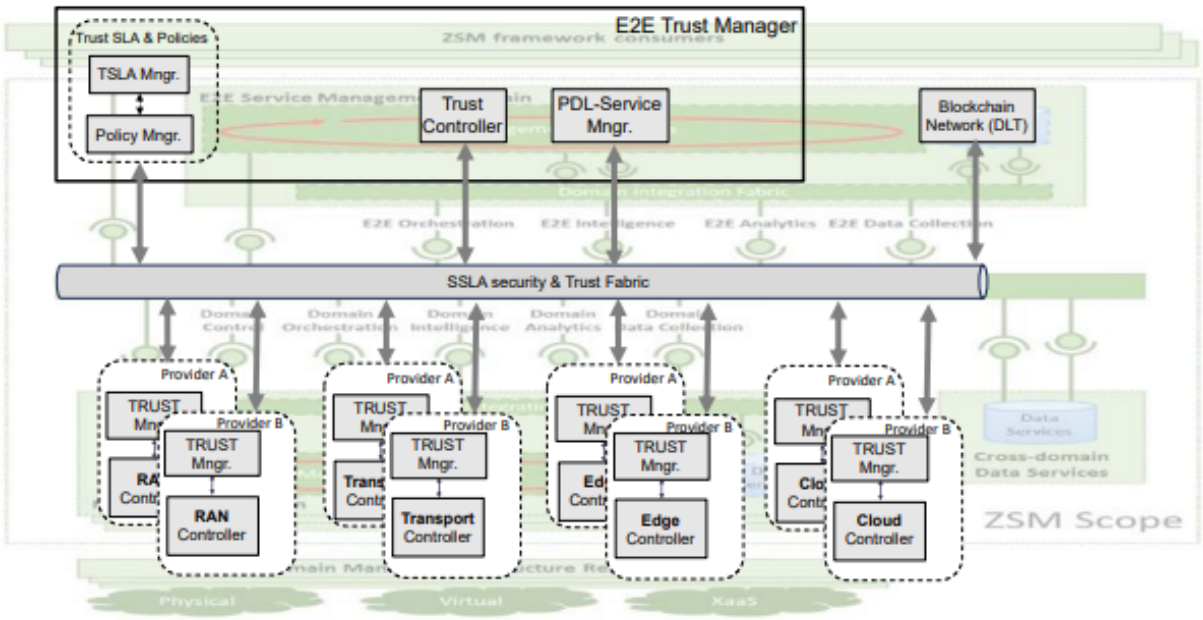


Figure 7: General Framework Mapping. Source:[44]

Fig. 7 [44] demonstrates the mapping of the TM and its different components in the general ZSM framework. Among these components, the following are highlighted:

- TSLAP: its main function is to create and process the Operator's TSLA, that establish how services are configured to meet a certain LoT. This component will integrate a set of deterministic rules to translate the TSLA requirements into trust policies to be later applied.

- TC: it is the core of the service that interacts with all elements and manages the state of processes and workflows in the TM. Its main function is to execute level of trust calculations within its domain or at the E2E level. Additionally, at the domain level, it converts/translates policies or capabilities defined in the SC into specific configurations and metrics to be collected for use in calculations.

- PSM: this component, identified in Fig. 7 [44] as PDL-Service Manager acts as the gateway for all DLT interactions, including SC management (definition, offers, requests, activation, monitoring, etc.).

3.3 Workflows

This section provides an overview of the key workflows involving the TM and the TC, focusing on their primary roles in orchestrating various processes. While some components mentioned may not be relevant to the current case and belong to a parallel security-focused project, the emphasis remains on trust management.

Initialization Workflow: The initialization process unfolds across four distinct phases. Firstly, provider resources are uploaded, i.e., the resources they want to offer to the operators, followed by the establishment of SLA and Trust Level Agreement (TLA) requirements. Subsequently, SC creation and offers are generated based on the defined SLAs. Finally, the offers are distributed among respective domains. Notably, the TC does not directly participate in these phases. This workflow is detailed in Appendix A, Fig. 35.

Deployment Workflow: The deployment workflow, also segmented into four phases, involves different actors at each stage. Initially, providers are selected based on service and requirement criteria. Next, service and security deployment activities are executed by components developed in another internal project. The TC assumes a pivotal role in the subsequent phase, overseeing trust deployment and activation processes (further elaborated in subsection 3.3.1). Finally, a closed-loop configuration phase concludes the deployment, managed in a different internal project. This workflow is detailed in Appendix A, Fig. 36.

Termination Workflow: Similar to the deployment workflow, the termination process comprises four phases, with each phase managed by different actors. The TC's involvement is highlighted in subsection 3.3.2, where its role is terminate deployed elements. This workflow is detailed in Appendix A, Fig. 37.

LoT Treatment Workflow: The final workflow, primarily orchestrated by the TC, revolves around handling LoT. This critical process is elaborated upon in subsection 3.3.3.

The workflows outlined in the subsequent subsections constitute the primary focus of this thesis,

elucidating the operational intricacies of trust management within the defined contexts.

3.3.1 Trust Deployment and Activation Request

As previous mentioned the TC plays a crucial role in the third phase of the deployment workflow, as illustrated in Fig. 8 [8], the Trust Deployment & Activation phase. At this stage of the workflow, there is a Network Security Instance (NSI) with the basic service and Network Security Funciton (NSF)s already deployed and configured in terms of security (achieved in the two previous phases), now, the idea is to follow a similar procedure but looking to achieve the trust requirements defined in the selected E2E TSLA. To do so, the E2E Network Slicing Security Orchestrator (NSSO) requests the activation of the E2E TSLA (step 1) to the E2E TC, which forwards this information to the E2E PSM (step 2) so this one may generate the event for the right domain and provider type (steps 3 – 5). After the right domain PSM takes the event, it requests the TSLA configuration to the TC (step 6), which in turn needs to get the translation of the E2E TSLA into policies using the E2E and domain TSLAP components (steps 7 – 11). In this moment and depending on the policies, the TC may deploy and configure new required NSFs focused on trust or simply configure those already deployed elements from the previous phase (steps 12 – 14). The same applies for the Security Probes (SP) illustrated in steps 15 to 18. In both cases, once these steps are done, the resulting outcomes are forwarded back to the E2E PSM through the domain PSM and DLT (steps 19 - 21), leaving the TSLA and the selected provider registered as actives. Finally, this information is also sent back to the E2E NSSO through the E2E TC to update the NSI (steps 22 – 24).

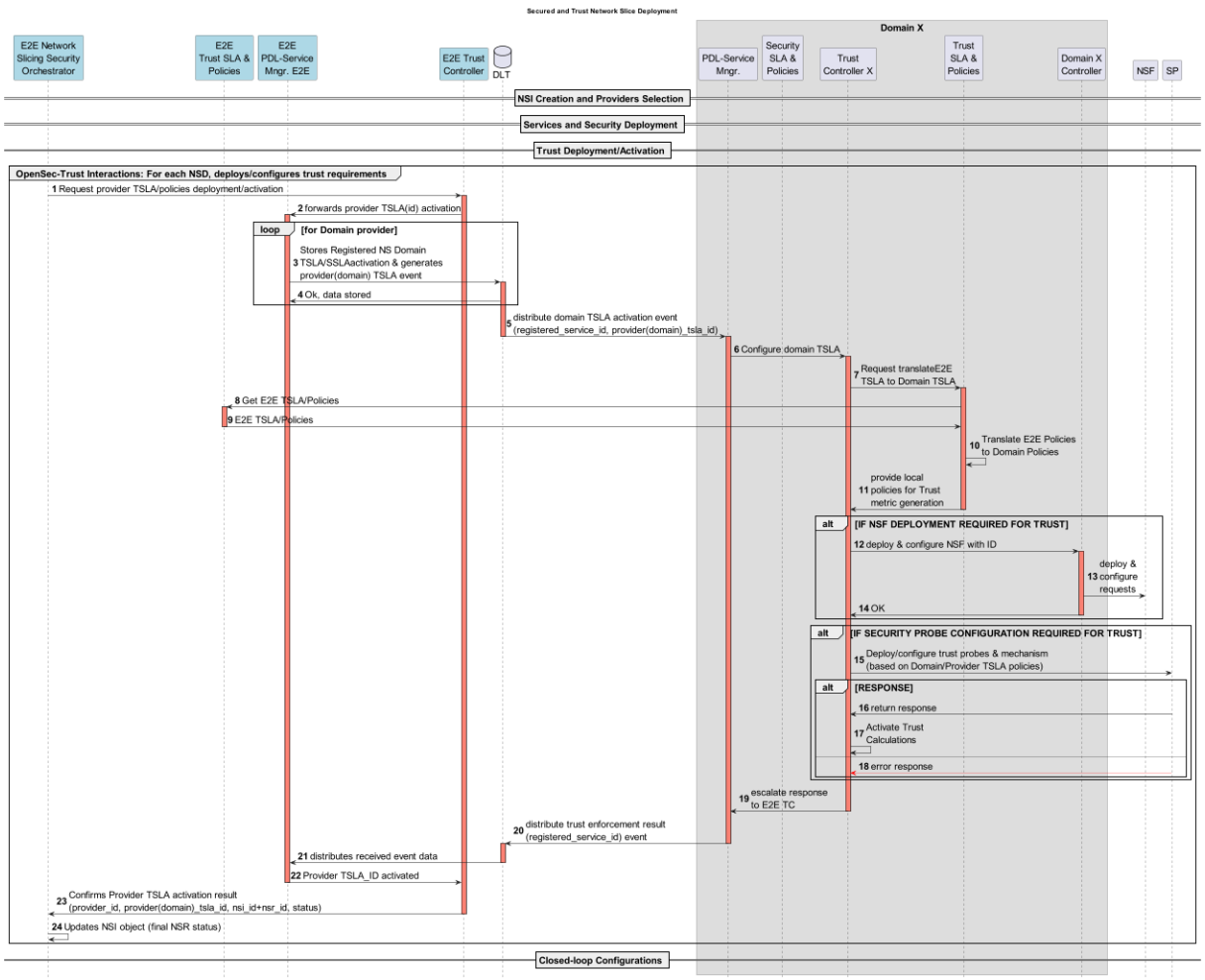


Figure 8: Trust Deployment/Activation Workflow. Source:[8]

3.3.2 Trust Deactivation Request

Another responsibility of the TC is to remove the trust-related configurations as illustrated in Fig. 9 [8] in the Trust Deactivation phase. To do that the E2E NSSO first interacts with the E2E TC (step 1) which through the E2E PSM reaches the different domain PSMs (steps 2 – 5). Then, each involved PSM requests to the TC to apply the actions to the affected deployed NSF and SP (steps 6 – 12) and answers back to the E2E PSM (step 13) through the DLT (steps 14 - 15). So, the E2E TC is finally informed about it (step 16) together with the E2E NSSO (step 17) that updates the NSI data (step 18).

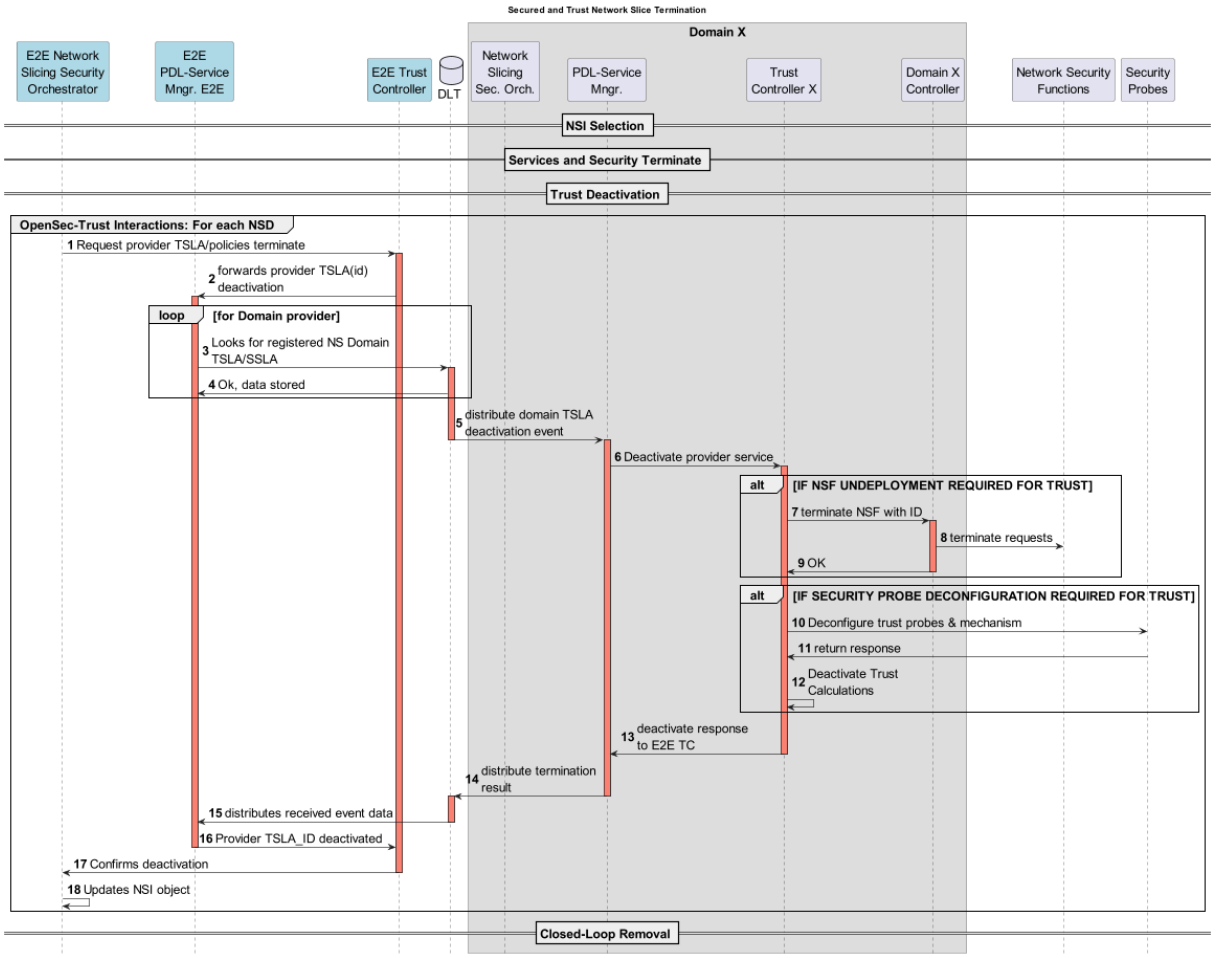


Figure 9: Trust Deactivation Workflow. Source:[8]

3.3.3 Level of Trust Treatment

Once a NSI and its NSF and SP are deployed, there is one more action done by the control systems that neither the operator nor the provider need to be aware of. This is illustrated in Fig. 10 [45] and is the LoT treatment, which implies the update of the LoT values associated to the selected NSFs and their providers. This phase begins when a domain TC, using the metrics defined in the selected SLA computes the Service LoT (step 1), and sends it to the PSM (step 2) so this one registers the updated value in the DLT and generates an event for the E2E PSM (steps 3, 4). When the event is received, the E2E PSM requests to the E2E TC to compute the updated value of the provider delivering the affected service (steps 5 - 7), and then the new value is updated by the E2E PSM in the DLT (step 8), so when a new providers selection is required for a another NSI deployment, the latest LoT values can be used.

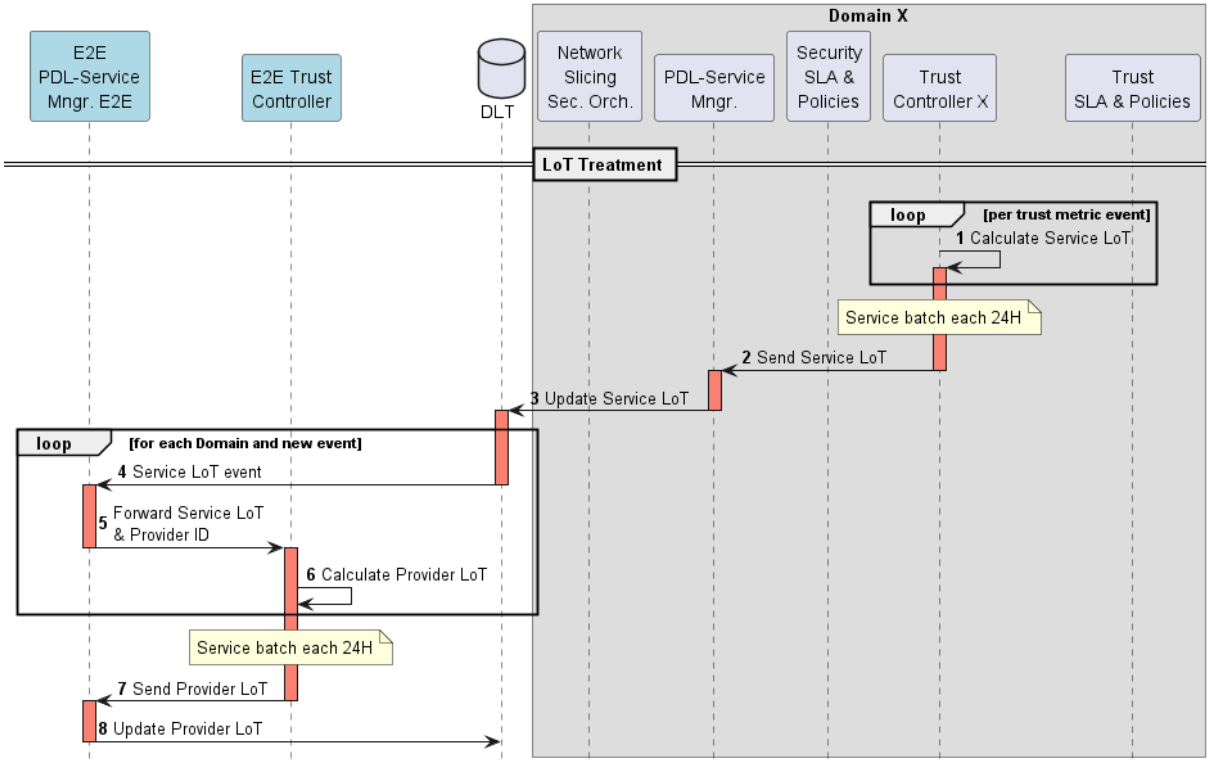


Figure 10: Level of Trust Treatment Workflow. Source:[45]

Chapter 4

Internal Architecture

In this chapter, the focus shifts towards the constitution of the internal components of the TC and its overall architecture. It is noteworthy to mention that there was an intention to align with the work presented in [5], where the TC is embedded within a more intricate and abstract component known as the TM. However, before delving into these details, it is imperative to clarify and comprehend the two distinct use cases in which the TC may find itself, depending on the context. These arise due to the architecture inherent to the construction of the TM called ZSM:

- E2E Domain Context: it has a role more focused on the overall management of the various TCs from each domain. It also possesses functions as input/output in relation to systems external to the TM.
- Local Domain Context: its role is focused on the very essence of a TC, that is, it is dedicated to the functionality of reliability calculation and integrations with the different components that make up the TM.

For simplification reasons, from this point forward, the two contexts are referred as E2E and Domain, respectively. As illustrated in Fig. 11, we will have an E2E TM for each E2E domain and a TM for each local domain. Within a TM, a TC will always be found. As previous mentioned, the functionality and objective will change depending on the context in which is used.

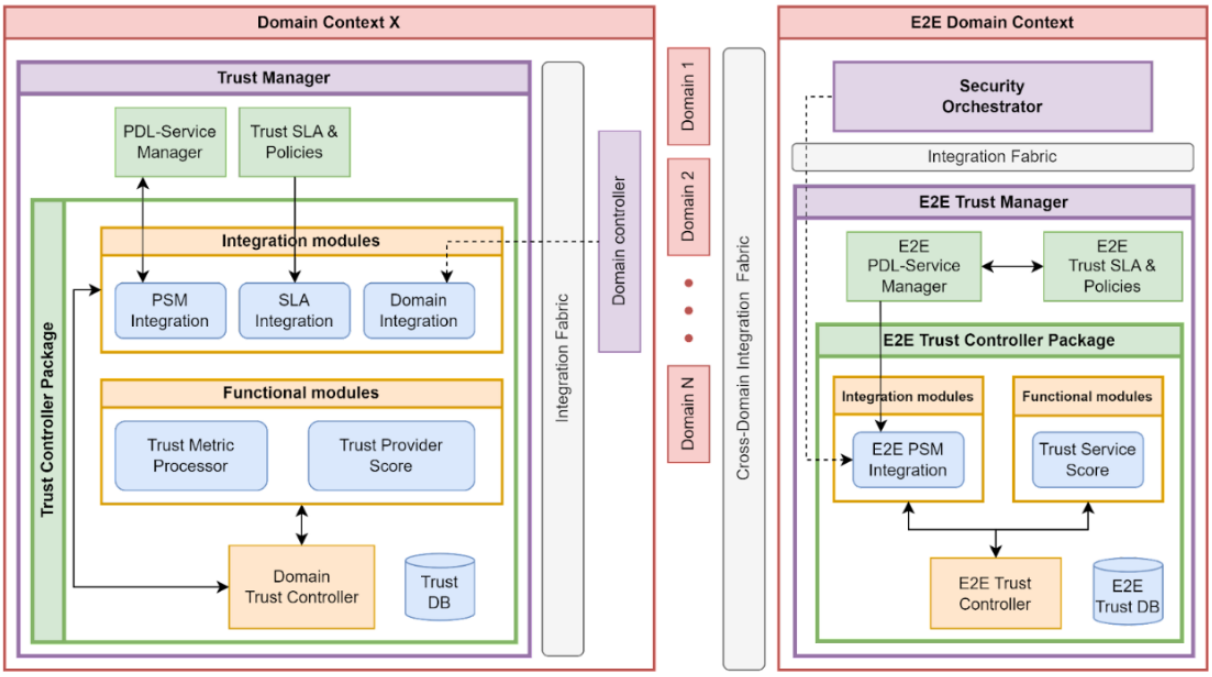


Figure 11: Comprehensive system diagram displaying all integrated components from the Trust Controller point of view.

Both the E2E TC and the TC package are structured into distinct modules, each designed with a specialized function. There are modules allocated for managing inbound and outbound communication (integration modules), others dedicated to executing precise operational functionalities (functional modules), and a central hub (E2E/Domain TC) that orchestrates the overarching system control and oversees the database management.

4.1 End-To-End Trust Controller

The E2E TC, illustrated in Fig. 12, is organized into several modules, each designed with specific functionalities to maximize cohesion and minimize complexity. This modular approach enhances scalability and maintenance.

The primary objective of the E2E TC is to manage E2E operations within the TC scope. It serves as a proxy between the E2E PSM and the Security Orchestrator (SO), while also calculating an E2E LoT using Domain LoTs data provided by the PSM entity. The system leverages message brokers for internal communication and provides a Representational State Transfer (REST) Application Programming Interface (API) for external sources.

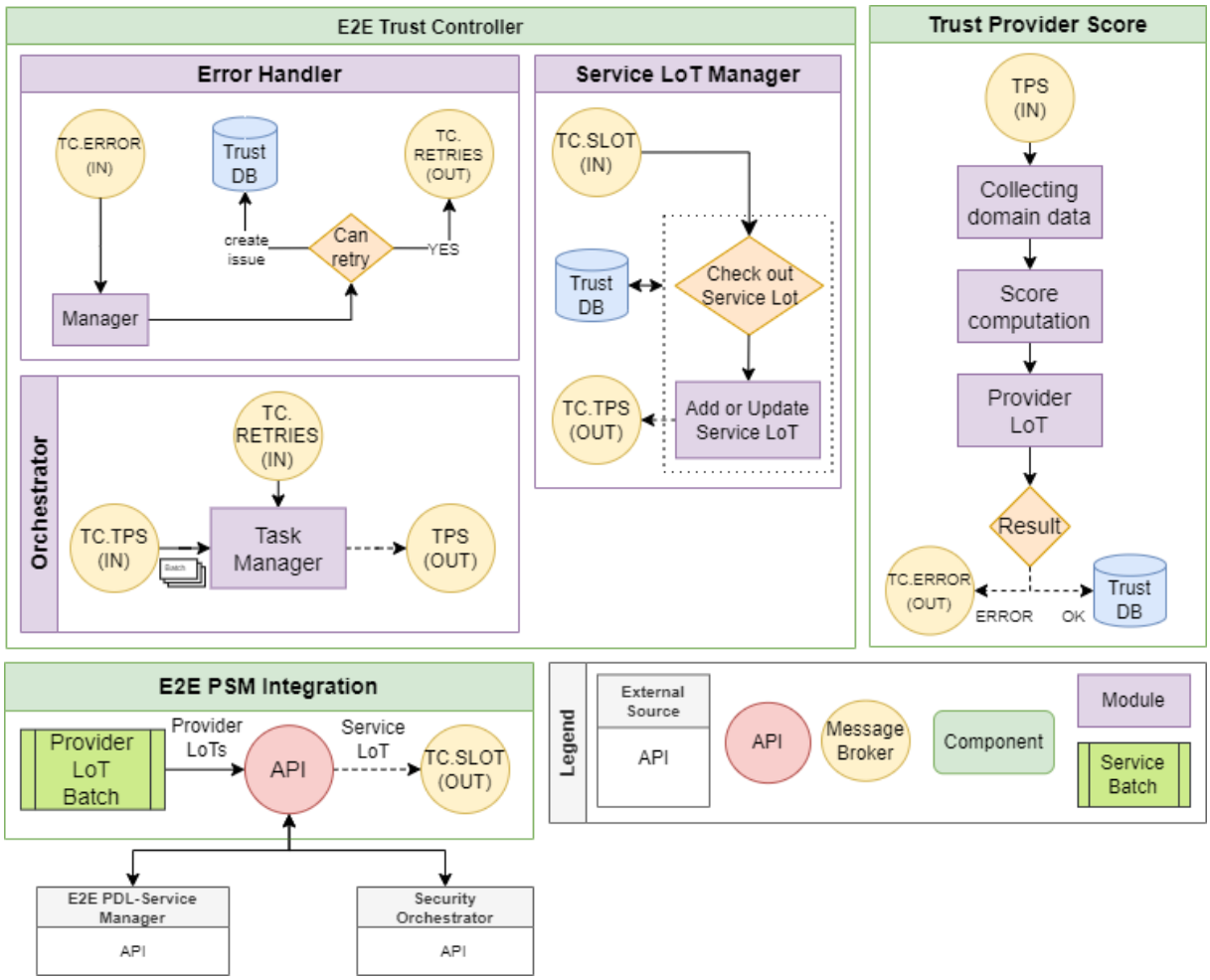


Figure 12: E2E Trust Controller internal architecture

To provide an overview, the E2E TC (subsection 4.1.1) consists of several modules dedicated to effective management. These include an Error Handler, a Service LoT Manager, and an Orchestrator that handles the relaunching of error requests. Additionally, the Trust Provider Score (TPS) (subsection 4.1.2) module is responsible for score computation, generating an E2E LoT from the known Domain LoTs. Finally, there is an E2E PSM Integration module (subsection 4.1.3) designed for integration with external sources such as the PSM and the SO.

4.1.1 E2E Trust Controller

The E2E TC (Fig. 13) is designed to manage the entire package scope, comprising three distinct modules, each with a specific task.

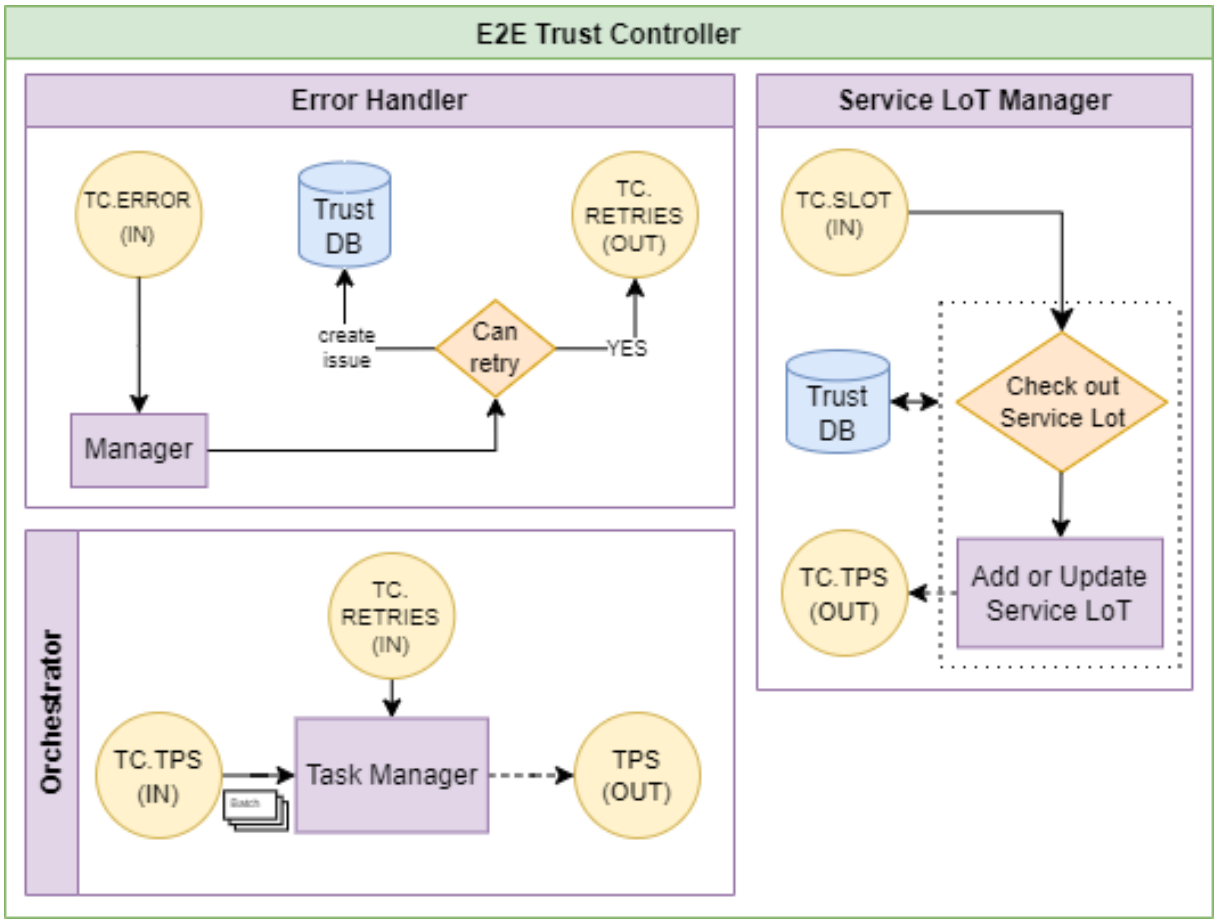


Figure 13: E2E Trust Controller module details

The Service LoT Manager module handles Domain LoT entries by consuming messages from the TC.SLOT queue. Its primary function is to create or update these values in the local database. Subsequently, it forwards the updated information to the Orchestrator via the TC.TPS queue, initiating the processing of a new E2E LoT.

The Orchestrator implements a batch service to streamline the processing of large datasets and eliminate duplicate executions. This approach is crucial for managing the substantial volume of LoT information from various domains, optimizing resource utilization, and enhancing the overall efficiency of trust assessments within the network. Its main function is to queue IDs related to Domain LoT changes and send them to the TPS module. Additionally, it incorporates a retry mechanism, discarding or retaining retry requests based on a timestamp comparison with the last registered execution in the database.

Respect to the Error Handler module, this module manages all failed requests or flows. Depending on the nature of the error, requests can either be retried or stored in the database for later analysis.

Responsibilities

- It is responsible for storing and updating all Domain LoT records received.
- It is responsible for stocking providers IDs and forward them to the TPS module.
- It is responsible for managing retry requests.
- It is responsible for handling errors at the E2E level and addressing domain issues that have been escalated from individual domains.

4.1.2 Trust Provider Score

This module is responsible for calculating the E2E LoT, also known as the Provider LoT, for a service provider.

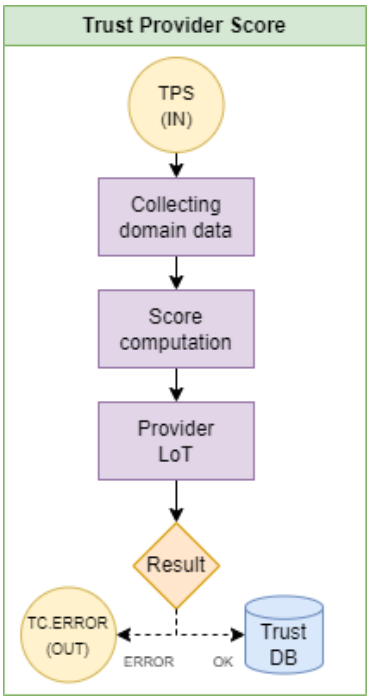


Figure 14: E2E Trust Provider Score module details

As illustrated in Fig. 14, the TPS module's primary function is to compute and store a new E2E LoT. It receives provider IDs through the TPS queue, indicating that the LoT for the associated provider has changed and needs to be updated or created. The process begins by fetching all the Domain LoTs associated with the given provider and applying a specific formula to generate the new E2E LoT. This computed value is then stored in the database.

Responsibilities

- It is responsible for generating an E2E LoT through Domain LoT data associated.
- It is responsible for registering all status of the execution steps in the database.

4.1.3 E2E PSM Integration

This module serves as an intermediary between the E2E PSM and the SO entities, facilitating their communication. Its internal structure primarily consists of a REST API and a batch service.

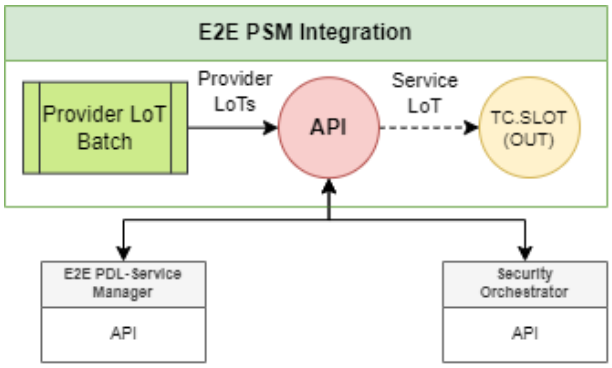


Figure 15: E2E PSM Integration module details

As illustrated in Fig. 15, the REST API exposes an endpoint for activating or deactivating trust capabilities at the E2E level. The SO uses this endpoint to send requests to the E2E PSM, which responds with a list of service providers that have their SC activated or deactivated. The module then relays this information back to the SO entity.

Additionally, the module acts as a receiver for Domain LoT (Service LoT) data, accepting input from the E2E PSM and dispatching it to the TC.SLOT queue for processing.

The module also includes a batch service designed to collect all E2E LoTs with a pending status (those not yet stored in the blockchain) and send them to the E2E PSM entity. Upon receiving a successful response, the batch service updates the status of the E2E LoTs: active ones become inactive, and pending ones become active. This batch service is triggered and scheduled to run twice a day.

Responsibilities

- It is responsible for receiving a request for activating/deactivating trust capabilities from the SO entity and return a provider list.

- It is responsible for receiving Domain LoT from the E2E PSM component and forward it to the Service LoT Manager.
- It is responsible for sending E2E LoT to the E2E PSM through a batch service.

4.2 Domain Trust Controller

The Domain TC, illustrated in Fig. 16, is structured similarly to the E2E TC. It consists of several modules, each designed with specific functionalities to enhance cohesion and reduce complexity, thereby ensuring scalability and ease of maintenance. The primary objective of the Domain TC is to activate trust calculations for a service provider and generate a Domain LoT using continuously monitored trust metrics.

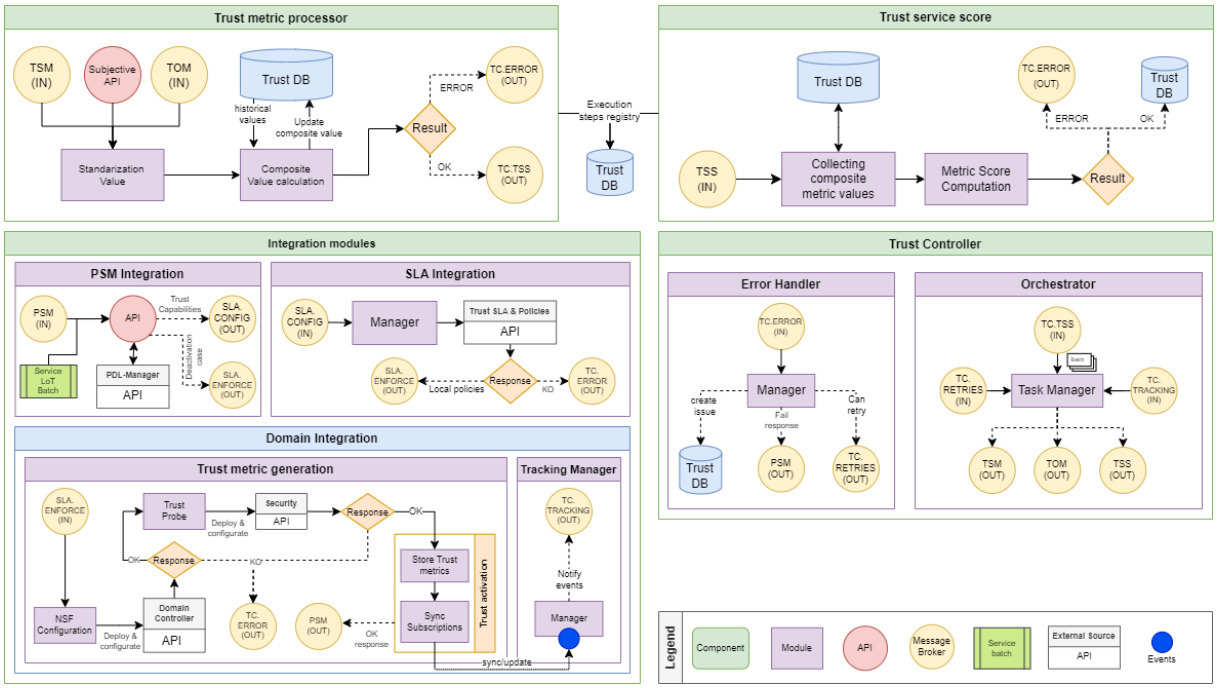


Figure 16: Domain Trust Controller internal architecture

To provide an overview, the functionality of the Domain TC has been highly modularized. It includes; integration modules: these modules encompass the business logic required for integrating with external artifacts, such as the PSM, TSLAP, and Domain Controller (DC); metric data collection and trust computation modules: these modules are dedicated to the collection of trust metrics and the computation of trust scores; orchestrator: to manage the entire trust flow, including task management and scheduling, and; error handler: to oversee error logic, including retries and error management, ensuring robust error handling and recovery.

By modularizing the system in this manner, the Domain TC is able to efficiently manage and process trust metrics, maintain high reliability through its error handling and retry mechanisms, and facilitate seamless integration with external systems.

4.2.1 Trust Controller

The Domain TC is a module designed to manage the entire scope of trust-related processes.

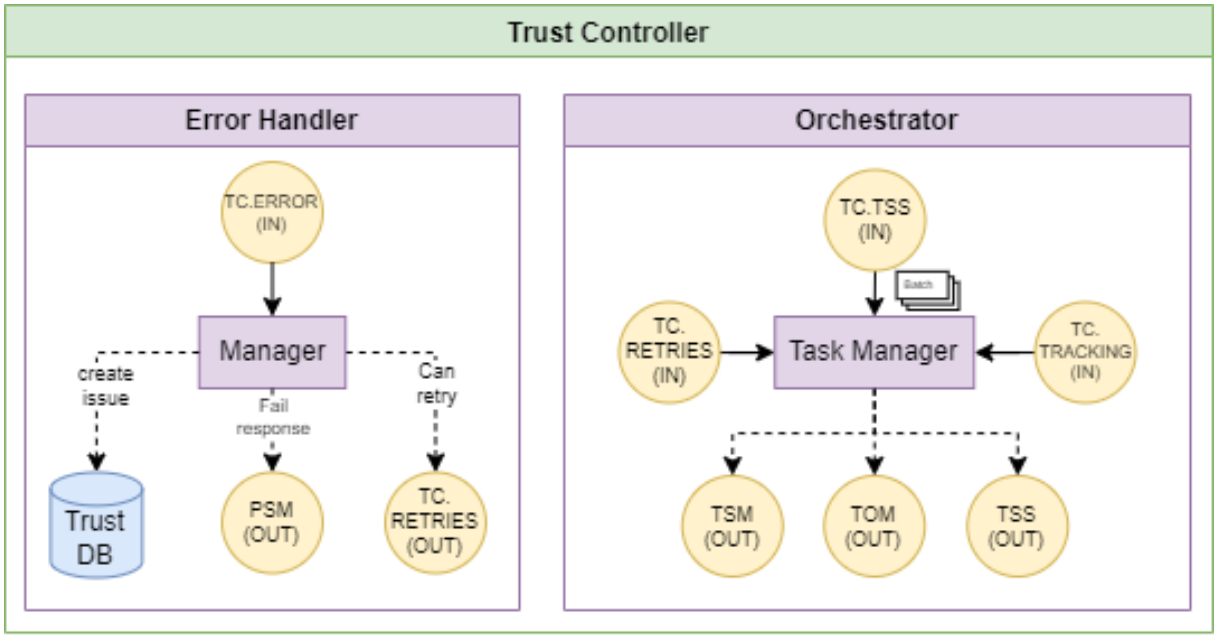


Figure 17: Domain Trust Controller module details

As illustrated in Fig. 17, the TC module consists of two submodules: the Orchestrator and the Error Handler.

The Orchestrator functions similarly to the Orchestrator at the E2E level (detailed in subsection 4.1.1). It includes a batch service connected to the TC.TSS queue, which is responsible for storing service IDs and dispatching them to the Trust Service Score (TSS) queue. Additionally, it utilizes the TC.RETRIES queue to manage and process retry requests. A new feature in this Domain TC is the TC.TRACKING queue, which receives notifications of metric changes and signals the Trust Metric Processor (TMP) module via the Trust Objective Metrics (TOM) and Trust Subjective Metrics (TSM) queues, depending on whether the metric is objective or subjective.

The Error Handler's primary role is to manage all failed requests or workflows. Depending on the type of error, requests can either be retried or logged in the database as issues for later analysis. Moreover, it handles failed requests originating from the PSM entity using the PSM queue.

Responsibilities

- It is responsible for stocking service IDs and forward them to the TSS module.
- It is responsible for managing retry requests.
- It is responsible for signal the TSS module on metric changes.
- It is responsible for handling errors at the Domain level.

4.2.2 Trust Metric Processor

The TMP is a crucial module responsible for receiving raw metric data, checking for potential penalizations related to applicable policies and/or SLAs, and generating a standardized value for subsequent use in calculating a composite value. This standardized value is then averaged with historical values to compute the final composite value.

It is important to distinguish between objective and subjective metrics, as detailed in subsection 2.2.3. To manage these metrics, the system utilizes a TOM queue, a TSM queue for retries, and a REST API (Subjective API) for new records. This structure is illustrated in Fig. 18.

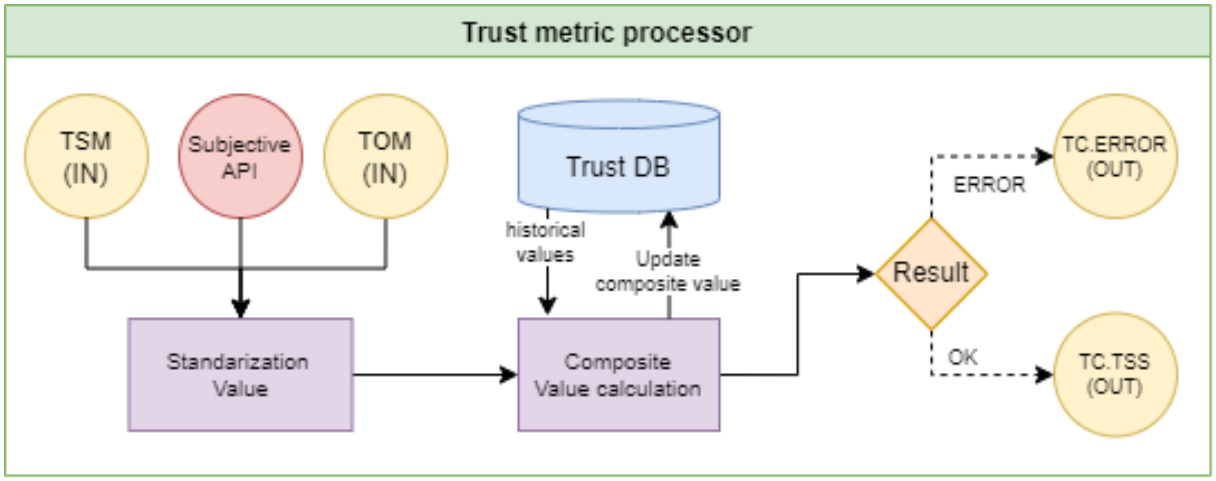


Figure 18: Trust Metric Processor module details

When a request is received through any entry point, such as the TOM queue or the Subjective API, the TMP initiates a new execution process. The goal of this execution is to compute a new composite value based on the received metric value, considering penalizations associated with the policies and/or SLAs of the metric, as well as historical values. This process is detailed in subsection 4.6.2.

If the computation is successful, the information is forwarded to the TC.TSS queue, which triggers the Domain LoT calculation by the Orchestrator. If an error occurs, the information is redirected to the TC.ERROR queue for further evaluation.

Responsibilities

- It is responsible for log all pertinent activities in the database, capturing essential execution details such as metrics, timestamps, steps of the process, current status, and the metric's type (objective or subjective).
- It is responsible for accurately retrieve and analyze policies and SLAs related to the metric group, and verify the existence of applicable penalizations.
- It is responsible for compute a standardized and composite value for each metric execution it receives.
- It is responsible for sending a signal to the TSS module to initiate a new Domain LoT calculation.

Subjective consideration

A subjective metric approach was considered and taken into account during the implementation phase. However, it is far from being fully operational as it is not within the scope of the current project. Therefore, it remains a pending topic for future work.

4.2.3 Trust Service Score

This module, depicted in Fig. 19, serves as the engine that synthesizes and evaluates composite metric values to produce a Domain LoT.

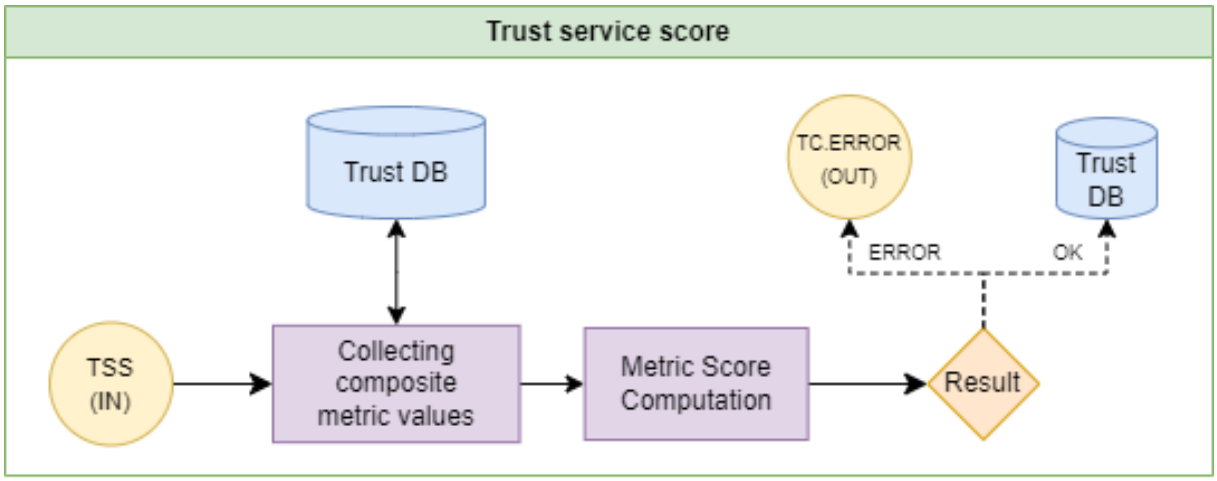


Figure 19: Trust Provider Score module details

Upon receiving an entry in the TSS queue, the module initiates a new execution process. This process involves retrieving a complete list of composite metric values associated with the service ID specified in the request. Following this, the module computes a new Domain LoT using a specific formula detailed in subsection 4.6.2.

If the computation is successful, the information is stored in the database. If an error occurs, the information is redirected to the TC.ERROR queue for further evaluation.

Responsibilities

- It is responsible for log all pertinent activities in the database, capturing essential execution details such as timestamps, steps of the process, current status, and Domain LoTs.
- It is responsible for collecting all composite metric values of a certain service for the subsequent calculation of his Domain LoT.

4.2.4 Integration Modules: PSM Integration

Effective communication with various stakeholders within the TM framework, including the PSM, is crucial. To facilitate this, integration modules have been created. The first module to be presented is the PSM Integration.

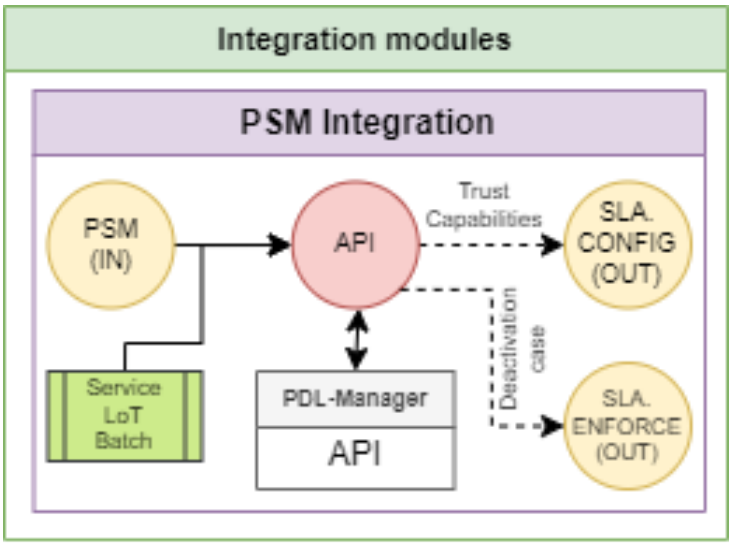


Figure 20: PSM Integration module details

As illustrated in Fig. 20, this module provides a REST API that exposes an endpoint for activating or deactivating the trust score for a specific service at the Domain level. In the activation scenario, the module receives a new request, which is then redirected to the SLA.CONFIG queue to trigger the configuration of new metrics. In the deactivation scenario, the request is redirected to the SLA.ENFORCE queue, triggering the removal of configurations and cessation of metric tracking.

Additionally, similar to the module presented in ssubection 4.1.3, this integration module includes a batch service. However, instead of collecting E2E LoTs, it collects Domain LoTs, or Service LoTs, which are then sent to the PSM entity to be stored in the blockchain. This batch service operates on the same way, being triggered and scheduled twice a day.

Responsibilities

- It is responsible for receiving requests for activating/deactivating trust scores from the PSM entity.
- It is responsible for sending Domain LoTs to the PSM through a batch service.

4.2.5 Integration Modules: SLA Integration

Effective communication with various stakeholders within the TM framework, including the TSLAP, is crucial. To facilitate this, integration modules have been created. The second module to be presented is the SLA Integration.

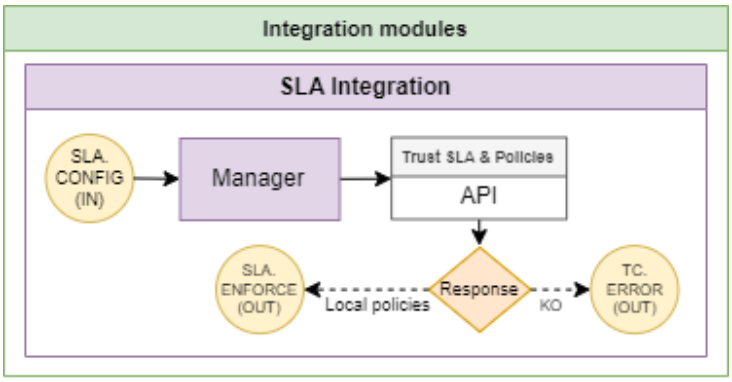


Figure 21: SLA Integration module details

This module, shown in Fig. 21, is responsible for retrieving local policies using a TSLA ID provided in the trust activation request. To achieve this, it uses the SLA.CONFIG queue to receive the request and interact with the TSLAP API to send the necessary information for retrieving the local policies.

The module is expected to store minimal information about SLAs and policies to apply penalizations, such as thresholds or target values. Additionally, it should be capable of retrieving the list of metrics and associating them with these policies and SLAs. The system for tracking these metrics should be established here in advance. This information is then stored in the database. Finally, the list of metrics is sent to the SLA.ENFORCE queue for configuration.

Responsibilities

- It is responsible for getting metric service related infromation from the TSLAP API given a TSLA ID.
- It is responsible for creating SLAs, Metrics and Associations in the database.

4.2.6 Integration Modules: Domain Integration

Effective communication with various stakeholders within the TM framework, including the DC, is crucial. To facilitate this, integration modules have been created. The third module to be presented is the Domain Integration.

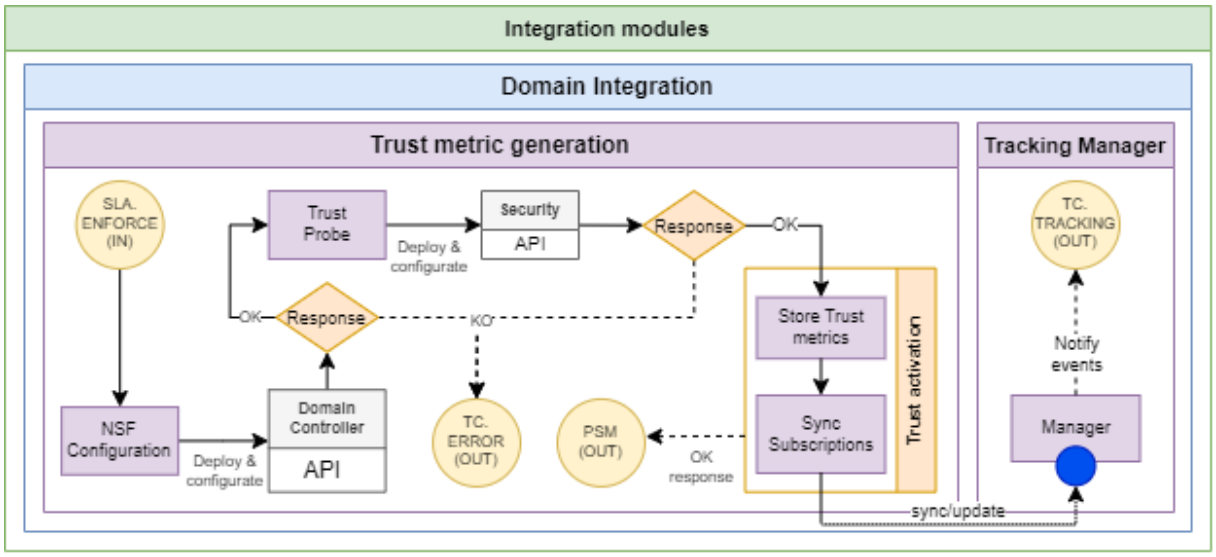


Figure 22: Domain Integration module details

This module, illustrated in Fig. 22, is responsible for registering all types of metric generation through subscriptions with probes and configuring/deploying NSFs. Once these tasks are completed, trust scoring is considered activated for a provider within the domain context. The artifact comprises two separate services: the trust metric generation service and the tracking manager service.

The trust metric generation service operates through the SLA.ENFORCE queue, which receives NSF and Security Probes (SP) configurations. It then enforces these configurations by sending NSF configurations to the DC and probes configurations to the Security component. Subsequently, the module stores all data associated with metric subscriptions in the corresponding table of the Domain Trust database. If everything is in order, it activates trust scoring and sends a confirmation message to the PSM queue.

Once activated, the tracking manager service begins receiving metric events from the SP component and redirects them to the TC.TRACKING queue, initiating the process of LoT calculation.

Responsibilities

- It is responsible for deploying and configure NSF.
- It is responsible for deploying and configure SP.
- It is responsible for create subscriptions to get metric events.
- It is responsible for notify the Orchestrator of new trust metric events.

4.3 Data Model

The Data Model section provides a detailed overview of the data structures and relationships that support the TC framework. This section is divided into three parts, each focusing on different aspects of the system's data architecture.

First, the E2E Level subsection presents the Entity-Relationship (ER) diagram specific to the E2E trust management, highlighting the primary entities and their interactions designed to effectively monitor and evaluate service providers.

Next, the Domain Level subsection delves into the data model at the domain level, detailing the entities and their relationships that facilitate the computation and management of trust metrics within individual domains.

Finally, the Common subsection identifies the tables and data structures shared between the E2E and Domain contexts, ensuring consistency and coherence across different levels of the trust management system.

4.3.1 E2E Level

This subsection presents the ER diagram shown in Fig. 23 for the E2E level, along with its detailed data types. The diagram provides a comprehensive visual representation, showing the different relationships and data structures that define the component's role within the broader system architecture.

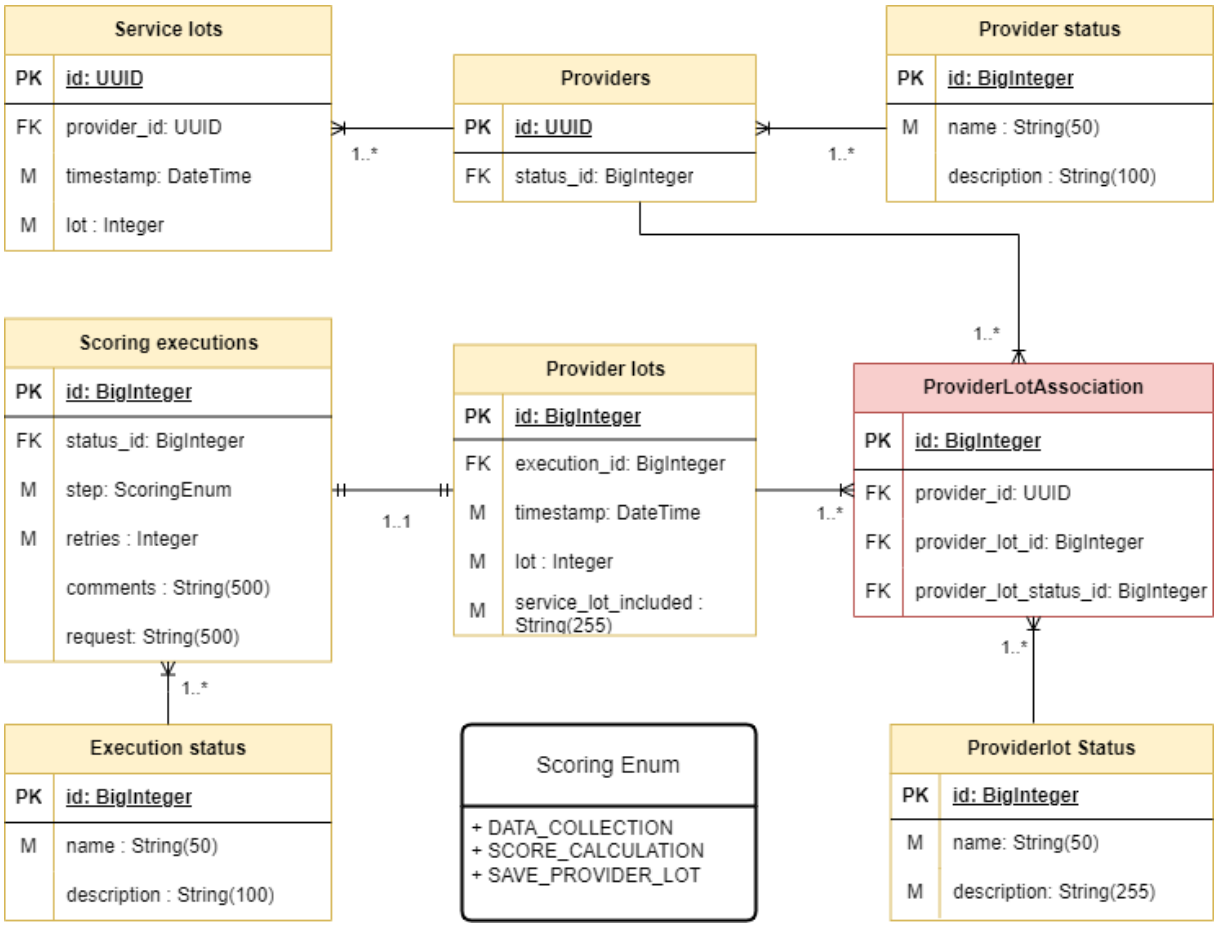


Figure 23: Entity-Relationship diagram of the E2E database

In the E2E context, the data model is designed with simplicity in mind, aiming to monitor providers effectively. The primary entities include Providers, representing the service providers themselves, and ProviderLoT, which captures the corresponding E2E LoT. Provider status is tracked through the ProviderStatus entity, denoting whether a provider is active or inactive. To compute the E2E LoT, each provider must have a set of associated Domain LoTs, facilitated by the ServiceLoT entity. ProviderLoTStatus is employed to monitor the status of all E2E LoTs, indicating whether they are active, pending, or inactive. The scoring process for LoT computation is managed by ScoringExecution, which logs each calculation step with the help of the ScoringEnum enumeration. The status of each execution is tracked by the ExecutionStatus entity, delineating whether the process is complete, ongoing, canceled, or failed. Lastly, the ProviderLoTAssociation table establishes the relationship between a provider, their ProviderLoTs, and their respective statuses.

4.3.2 Domain Level

This subsection focuses on presenting the ER diagram shown in Fig. 24 for the Domain level, along with its detailed data types. The diagram provides a comprehensive visual representation, showing the different relationships and data structures that define the component's role within the broader system architecture.

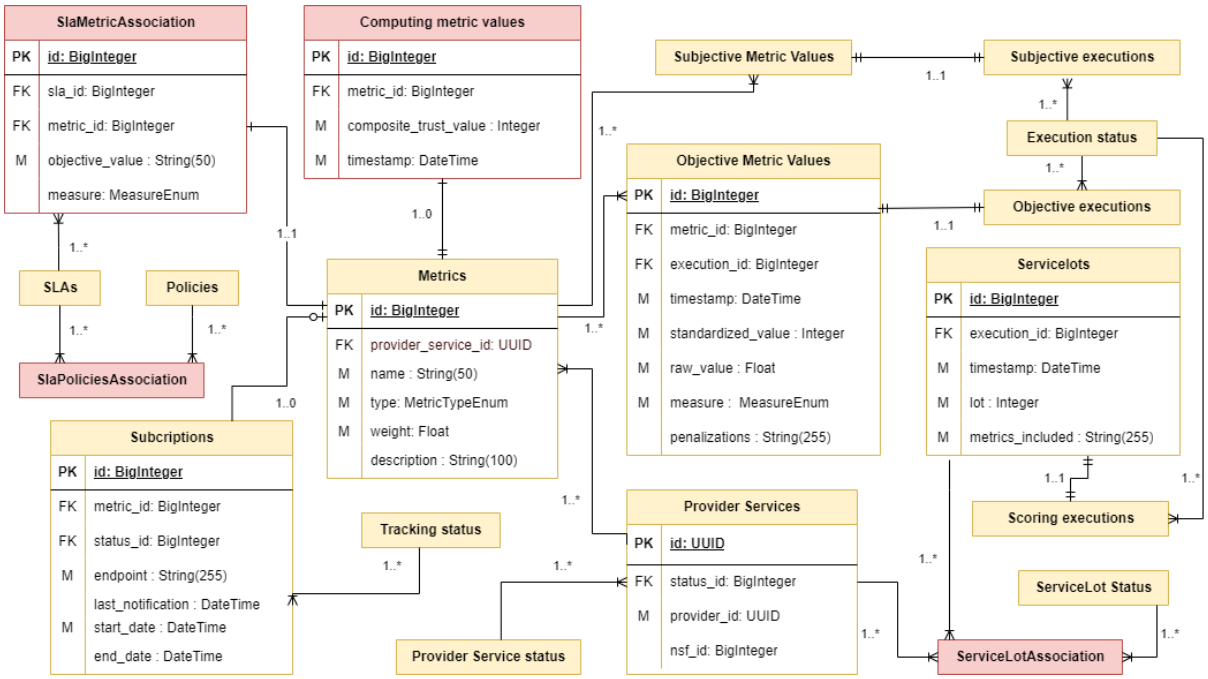


Figure 24: Domain Database Entity-Relationship Diagram (simplified)

In this data model, key elements include a table for storing metrics (Metrics entity) and another for capturing their values and related information (ObjectiveMetricValues or SubjectiveMetricValues). Additionally, separate junction tables are established to associate Policies and SLAs with the metrics. A dedicated table for computing the LoT, named ComputingMetricValues, is implemented to store the final value of a metric. This table includes a timestamp for confirmation purposes, enabling the verification of any deviations between the last value considered for calculating the new composite value and the current value. As in the E2E context, we have execution tables, ScoringExecutions, ObjectiveExecutions and SubjectivesExecutions, whose purpose is to log execution steps and manage retry control. The remaining tables related to providers and services serve the same purpose as in the E2E level, but within the Domain context. Lastly the table Subscriptions is mainly responsible for store metric related information, such as, where to get access to the respective probes, i.e., the metric events associated to the service. A full detailed version can be seen in Appendix A, section A.5.

In Fig. 25, the enumerations utilized for the Domain context are presented. Among these, Scoring

and Metric Process serve as enumerations to log execution steps, while the other two belong to metric configurations: one denotes its type, and the other describes how it is measured.

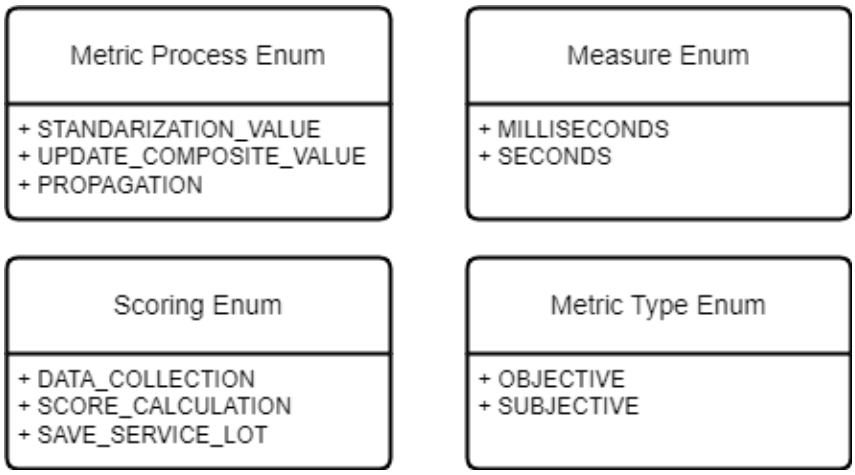


Figure 25: Domain Database Enumerations

4.3.3 Common

As presented in Fig. 26, certain tables are shared between the E2E and Domain contexts. These mainly, relate to errors and system parameters, as they contain data that remains consistent across both levels.

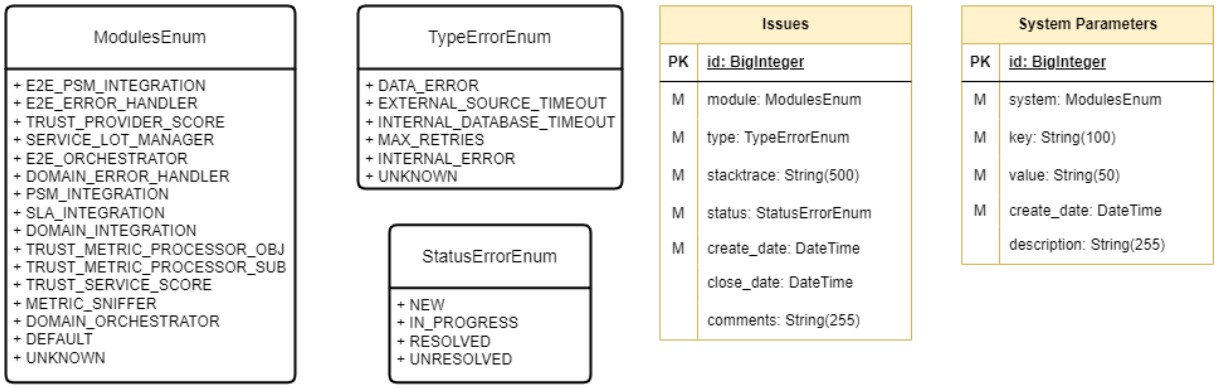


Figure 26: E2E and Domain Common Data Model

4.4 Data Objects

Regarding the design of data objects, it played a crucial role in the project's development phase. Firstly, understanding the content type received at each API endpoint was essential to ensure proper data handling. Secondly, it was important to align with the expected data objects from external sources. These two are named external data objects. Meanwhile, internal data objects were built to seamlessly traverse RabbitMQ

queues. Typically, each queue is associated with a specific data object, except in scenarios such as, for example, if we are processing the request of a provider score, TPS queue, it can either be a new request, so the data object will contain the provider id, or a retried request (a request that failed and its being re done), and the data object will contain an execution id instead, so that the previous process computation can be restored if possible. All data objects are of dictionary type, ensuring JavaScript Object Notation (JSON) serialization for efficient production and consumption from queues. The only exception are the Error, ExecutionError, and ErrorRequest classes designed to handle system errors. However these classes are equipped with methods for proper serialization.

The error classes, utilized by a dedicated error handler, differentiate between various types of errors that occur within the system. Both the ExecutionError and ErrorRequest classes are related to errors that happen in specific modules of the system. The ExecutionError class, as the name suggests, deals with execution errors typically encountered in modules like TPS or TMP, where computational requests are initiated. These errors allow for a predefined number of retries before being marked as failed, as specified in the system parameters table. On the other hand, the ErrorRequest class addresses errors originating from external sources, particularly in integration modules. Unlike the former, these errors lack a robust retry mechanism. Lastly, the Error class encompasses all other errors occurring within the core system, including TC or E2E TC errors.

For a detailed breakdown of all the defined data objects, refer to Appendix A, section A.4.

4.5 External Integrations

This section aims to summarize all the interactions of the TC with external sources in both contexts, E2E and Domain.

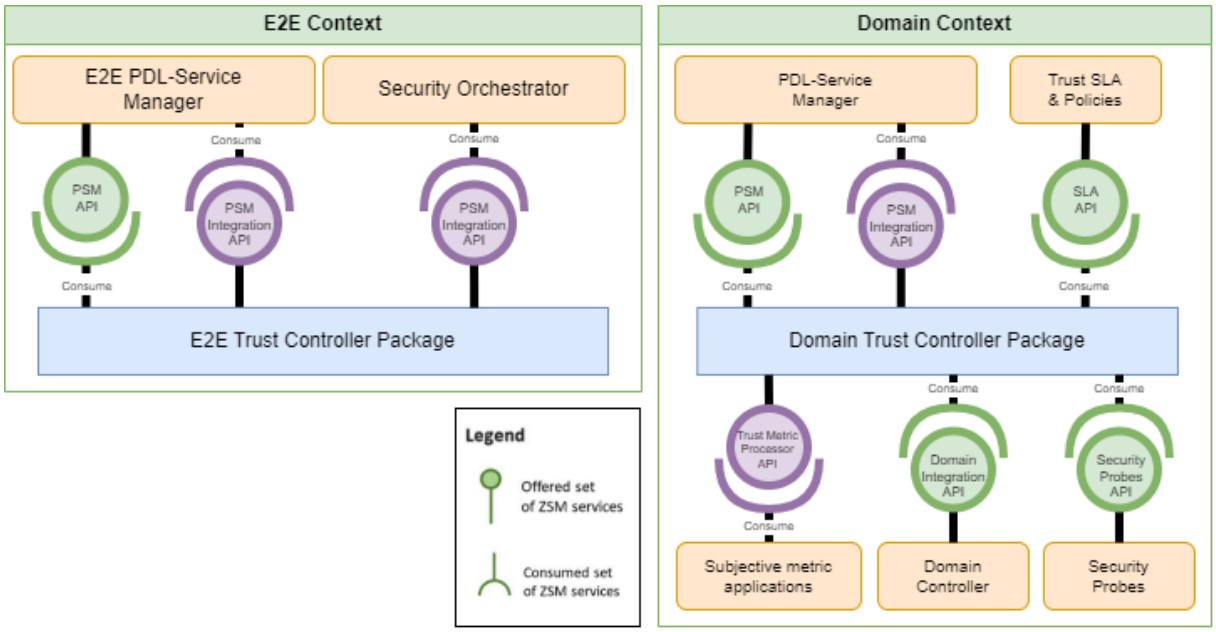


Figure 27: E2E and Domain Context External Integrations

Beginning with the E2E level, presented on the left side of Fig. 27, as we can see the E2E TC can either consume or provide information from and to external sources. This functionality is facilitated through the implementation of an API featuring two distinct endpoints: one for processing activation (POST) or deactivation (DELETE) requests from the SO entity and another for receiving service LoTs from the E2E PSM. On the consumption side, the E2E TC interacts with two endpoints: the first, provided by the SO entity, handles redirection of activation (POST) or deactivation (DELETE) requests, while the second sends newly calculated provider LoTs to the E2E PSM for blockchain registration.

In contrast, at the Domain level, illustrated on the right side of Fig. 27, the scope of interactions is more extensive. Here, an additional API was developed with a singular endpoint dedicated to receiving activation or deactivation requests for trust capabilities specific to a service from the PSM entity. This operation occurs asynchronously. Furthermore, the TMP component features an API for managing client reports and feedback that are handled by subjective metrics. This API offers endpoints for listing available service metrics (GET) and receiving client feedback (POST). Although a text-based feedback endpoint for analysis by ML models was considered during the design phase, all this subjective consideration was considered beyond the project's scope. On the consumption front, the TC does it when the cron service is activated to send newly computed service LoTs to the PSM. Additionally, it consumes data from the TSLAP to get all the service related information like NSF and Probes configurations and metric details, it also consumes from the SP entity to request the initialization of metric events, and finally from the DC to deploy and configure the NSF.

For a detailed breakdown of the exposed and consumed endpoints, refer to Appendix A, section A.1 and A.2.

4.6 Level of Trust Model: Analysis and Computation

LoT its a percentage value that measures the level of trust of a provider or service, depending on the context it appears, E2E or Domain, respectively.

LoT model has been designed to provide a comprehensive and accurate assessment of the trust associated with a provider or service. There are two levels of LoT, E2E and Domain, with the latter carrying more complexity, integrating both quantitative and qualitative factors. This model consists of two main elements: one focused on the processing of objective metrics and another dedicated to the calculation of the LoT.

Table 2 presents the trust provider rank classification adopted in this project. Dividing this classification into five categories simplifies decision-making by offering distinct trust states. Using categories instead of continuous percentage values provides a clear and intuitive framework, allowing stakeholders to make quick and informed decisions. This approach balances granularity with accessibility, making trust assessments straightforward and actionable.

Table 2: Trust rank classification

Trust Rank	Completely Untrusted	Untrusted	Weak Trusted	Strong Trusted	Completely Trusted
LoT (%)	0-20	20-40	40-60	60-80	80-100

4.6.1 E2E Level

In the E2E model, the process is efficiently managed through the integration of three key modules:

1. Receiving the Domain LoT: this function is a part of the E2E PSM Integration module. It uses a RESTful API to receive Domain LoTs, ensuring effective communication and data exchange. The request for these LoTs originates from the blockchain and is sent by E2E PSM component.

2. Recording and updating existing LoT: the E2E TC module is tasked with recording and updating E2E LoTs in the local database. This process provides a reliable and current snapshot of each domain's value associated with a provider.

3. Calculation of Provider LoT: E2E TPS module is dedicated to generating the E2E LoT, this module plays a crucial role in maintaining the integrity and accuracy of the service process. When a calculation thread or process initiates, the module fetches all the latest available Domain LoTs for a specific service. It then calculates the average of the available values at that moment, in relation to the number of domains for which data is available.

$$\text{E2E LoT} = \frac{\sum_{x=1}^{n} DLoT_x}{n} \tag{4.1}$$

Where:

- $DLoT$ is the Domain Level of Trust
- n is the total number of records

4.6.2 Domain Level

At the Domain level the calculation process of the LoT of a service incorporates the integration of several modules within the TC and can be summarized by the following four steps:

1. Metrics Tracking: the system continuously monitor service metrics through the tracking manager module. This includes real-time monitoring of metric values via subscriptions established within the trust metric generation module upon communication with the DC component. An event should be triggered whenever there is a change in a metric's value.

2. Processing and Standardization: raw metric values undergo processing to yield standardized values. This entails the application of associated policies and SLAs, including the appropriate penalties. The formula employed for this process is as follows:

$$SV_x = 100 - P_x \tag{4.2}$$

Where:

- SV_x is the standardized value of the metric x
- P_x is the penalty value associated with metric x

As previous mentioned, each metric has an associated threshold, derived from SLAs, that represent acceptable performance limits. The penalty value is calculated based on the distance from the requested limit, i.e. Objective Value (OV). This proportional approach ensures that penalties are fair and reflective of the actual performance. For instance, a slight deviation from the threshold incurs a minimal penalty, while a significant deviation results in a substantial penalty. This choice is grounded in real-world applicability as in practical scenarios, stakeholders are concerned with how far a metric deviates from acceptable limits, as this directly impacts service quality and reliability. Therefore, this approach aligns well with practical needs and expectations. The following formulas are used to calculate the penalty in different cases:

(a) When the OV has a maximum limit:

$$P_x = \max(0, \min(100, 100 * \frac{RV_x - OV}{OV})) \tag{4.3}$$

(b) When the OV has a minimum limit:

$$P_x = \max(0, \min(100, 100 * \frac{OV - RV_x}{OV})) \tag{4.4}$$

(c) When the OV has both a minimum and maximum limit:

$$P_x = \max(0, \min(100, 100 * \frac{\max(RV_x - OV_{max}, OV_{min} - RV_x)}{OV_{max} - OV_{min}})) \tag{4.5}$$

(d) When the OV doesn't exist:

$$P_x = 100 - (100 * RV_x) \tag{4.6}$$

Where:

- RV_x is the raw value of the metric x
- OV is the objective value of a metric, representing its required limits (minimum and/or maximum)

3. Calculation of the Composite Value: The next step is computing a composite value, derived as the weighted average between the standardized value and its historical counterpart. The formula employed for this calculation is expressed as:

$$CV = \frac{\sum_{x=1}^{n} SV_x}{n} \tag{4.7}$$

Where:

- SV_x is the standardized value of the metric x
- n is the total number of records

4. Calculation of the Domain LoT: once the composite value has been calculated, the process to recalculate the Service LoT can began. Predefined weights are assigned to the metrics based on the SLA of the respective service. The sum of the weights must be 1. With this understanding, the LoT is determined by calculating the average of the composite values of the existing metrics, each multiplied by its respective weight:

$$DomainLoT = \sum_{x=1}^{m} CV_x * weight_x \tag{4.8}$$

Where:

- CV_x is the composite value of the metric x
- m is the total number of metrics
- $weight_x$ is the contribution of the metric x to the LoT calculation

At this point a Domain LoT data object representing the comprehensive trust level of the service within a domain, is created to be stored in the local database and subsequently transmitted to the blockchain for secure record-keeping and storage, facilitated by the PSM component. This four-step computation pipeline is done within three of the modules composing the Domain TC. Each of the modules accomplishes the following of the tasks. First, the Domain Integration module (subsection 4.2.6) is in charge of the first task (i.e., metrics tracking). Then, the TMP (subsection 4.2.2) takes care to apply the second and third tasks (i.e., processing and standardization, calculation of composite value). Finally, the TSS (subsection 4.2.3) module is the responsible for the last task (i.e., calculation and propagation of the Domain LoT).

Chapter 5

Implementation

This chapter, in a initial phrase, delves into the methodology adopted during the project's development and the technological choices made to ensure efficiency, scalability, and maintainability. From project management to code implementation, each decision was carefully considered to align with the project's objectives and requirements. Its worth mention that some of the choices here presented were taken as a team and not by me individually.

Later on, is explored the practical realization of the conceptual framework outlined in the preceding chapters. It provides a comprehensive overview of how the theoretical concepts were translated into a functional system, detailing the organization of code, development methodologies employed, and critical decisions made during the implementation process. The chapter is structured to highlight the progression through various development phases, each representing a significant milestone in the project's evolution. Additionally, critical decisions that shaped the project's architecture and functionality are described, offering insights into the rationale behind key design choices.

5.1 Work Methodology

Adhering to the principles of Scrum and Agile methodologies, the project followed a sprint-based approach with weekly iterations. This facilitated the agile principle of "delivering working software frequently" by breaking down tasks into manageable units and scheduling them into sprints. Task management and collaboration were streamlined through the GitLab platform, serving as both project management tool and code repository. The distribution of tasks spanned various areas including requirements, technology investigation, system design, implementation, testing, and support.

5.2 Technological Choices

During the development of the project, careful consideration was given to selecting the appropriate tools and technologies to ensure efficiency, scalability, and maintainability. Here are presented the main reasons behind these options.

As already mentioned, one of the first options taking in consideration was the use of GitLab for project and team management, as well as code repository hosting. This tool provides a comprehensive set of features for collaboration, issue tracking, continuous integration/delivery, and version control, streamlining the development workflow and enhancing the team productivity.

As for the primary programming language, Python was chosen. It offers a large set of libraries and frameworks, making it an ideal choice for building complex systems. Furthermore its simplicity, readability, and extensive community support where also topics with great weight on the choice. PyCharm was utilized as Integrated Development Environment (IDE) for its robust features, intelligent code completion, and seamless integration with GitLab.

For a efficient and decoupled communication between the system components, the message broker chosen was RabbitMQ, since it offers and facilitates asynchronous communication through queues. Apache Kafka was also considered but it ended discarded since its not suitable for real-time processing.

Regarding database management, the choice of PostgreSQL is based on its advanced capabilities and robustness. Other than that, PostgreSQL offers Atomicity, Consistency, Isolation, and Durability (ACID) compliance, meaning that database transactions are processed in a reliable and correct way. To access and observe the database data, the management tool chosen was DBeaver due to its versatile and user-friendly interface.

Between Flask and FastAPI, the later was chosen for building the REST APIs, primarily, due to its support and greater capacity to handle asynchronous programming. Its fast performance and documentation of API endpoints are also important features.

Docker is used as a tool for containerization, enabling the application and its dependencies to be packed into lightweight and portable containers. Docker ensures consistency across development, testing, and production environments, streamlining deployment and scaling processes.

Lastly, one important choice was, as Python's native unit testing framework, the use of Unittest. This tool allowed to conduct rigorous testing of individual portions of code, ensuring the correct operation of the different software components.

5.3 Code Structure

The architecture of the TC component is structured in a way that tries to meet the principles of modularity, reusability, and efficiency, as well as streamline both development and maintenance workflows. Following a microservices-based approach, the application is divided in small and autonomous services interconnected via RabbitMQ queues. This approach facilitates scalability allowing the system to grow independently.

The code is organized with a focus on delineating responsibilities and fostering cohesion among related functionalities. Each microservice is structured with a clear set of modules, where some have multiple functionalities and others are dedicated to singular tasks. The starting point of each component is always a Main.py file, that manages the initialization of threads for each module, that, in turn instantiate specific services where the core logic of the component resides. To ensure clear delineation of responsibilities within each component, normally the code is organized into separate files: one for the service logic, another for handling execution functions such as database calls, and a final one dedicated to implementing database access queries.

As mentioned, the entire codebase is hosted within a single GitLab repository. At the top level, folders are organized in both the E2E and Domain contexts, alongside the database, and an utility library project shared among all components. Within these folders, each microservice is documented in a README file, explaining its purpose, functionality, and deployment steps.

5.4 Development Phases

The development of this project can be concised in 5 different phases. This section presents and describes the most important tasks performed in each one of them.

5.4.1 Phase 1: Database Implementation

The first phase of development consisted on a initial stage for establishing foundations. The first step was the development of both E2E and Domain databases as well as the messaging service. It was decided to deploy the RabbitMQ server alongside the database, since they're both needed for almost all the components in the system. So, the first container to be deployed will host the database and the messaging service at the same time on a common network specified in the docker file. In order to promote code reusability and encourage modularity it was developed a common-utility project that's used as an internal library, minimizing redundancies. This library includes, for example, database and RabbitMQ related calls,

as well as a common error handler implemented from scratch and constants that are used and shared among all components in the system. The next step of this phase consisted in the creation of a functional template including the minimal technologies required for all the artifacts, such as, a set up for an API, messaging queues, etc, ensuring that every new component begins with a consistent, well-defined structure equipped with endpoints for basic operations and a set of unit tests. Lastly, it is defined at this point, an unit and integration test plan for all use cases, in both, E2E and Domain contexts.

5.4.2 Phase 2: E2E Implementation

The second phase focused on implementing and testing all components within the E2E context, followed by refactoring and documentation. The starting point was the E2E TC component (section 13), that incorporates three distinct modules with dedicated queues. The simpler one, Service LoT Manager, in terms of implementation was pretty straightforward since it's only task is to execute database calls. Slightly increasing the complexity, the Error Handler module, manages system errors by analyzing their type (where did it came from, how many times happen, etc) and deciding whether to log them in the database or send them to the next module, Orchestrator, to be retried. The Orchestrator is the core operational module, managing new and retried requests within a specific timeframe or message count before dispatching them for processing. This procedure is done on the next component built, the TPS (subsection 4.1.2), that didn't bring any implementations challenges. Its main quest is to compute a new E2E LoT based on Domain LoTs, following the formula presented on subsection 4.6.1. The most challenging aspect was developing the E2E PSM Integration (subsection 4.1.3), which includes an API service built with FastAPI to communicate with the external E2E PSM source. This service receives and dispatches new requests to the Orchestrator and acts as an intermediary between E2E PSM and SO entities. Additionally, a cron service was implemented to update E2E LoTs in the blockchain periodically. This service is schedule to activate two times a day with a small difference, every failed execution on the first time can be retried on the second time before being canceled and stored in the database for later analysis. The periodicity chosen aims to minimize excessive calls to the blockchain, since the LoT will be constantly changing for each metric event received. To ensure the integrity and reliability of the developed components, unit tests were defined for each one of them, and integrated into the Docker deployment process. Deployment is aborted if any test fails. Each artifact was also documented thoroughly in README files. Finally, the previously established integration test plan was conducted to ensure proper connectivity between containers hosting artifacts and those hosting the database and RabbitMQ server, verifying the correctness of interactions and data flow.

5.4.3 Phase 3: Partial Domain Implementation

Similarly to the second one, the third phase involved implementing and testing nearly all components, but this time at the Domain level, also followed by a process of refactoring and documentation. The development of the Domain TC (section 4.2) was based on the E2E TC, with some modifications. For instance, a Service LoT Manager-like module was not required, and the Orchestrator had to manage the metrics tracking system in addition to handling new and retried requests, i.e., every time a metric changed was discovered, the right component needed to be notified. This component, TMP (subsection 4.2.2), was implemented to analyze these metric changes and compute its value after applying the right penalizations based on defined requirements, as explained in subsection 4.6.2, to further send a start request for a new Domain LoT computation. After going through the Orchestrator this request arrives to the TSS (subsection 4.2.3) that works in a similar way to the TPS, computing a new Domain LoT following the predefined formula presented in subsection 4.6.2. The final artifact developed in this phase was the PSM Integration, which, similar to the E2E context, includes an API and a cron service. However, the API in the Domain context receives requests to activate or deactivate the metrics tracking system for a specific service and provider asynchronously, so that the artifact can continue is execution after redirecting them to the right component. The workflow from this point onward is similar to that of the second phase: unit tests are conducted before deployment, followed by documentation of each artifact and execution of the previously established integration tests.

5.4.4 Phase 4: Domain and SLA Integration Implementation

The fourth phase of the implementation focused on the development of the final two integration modules: Domain and SLA. This phase presented several challenges due to its dependency on external sources, which were not accessible at the time of the project's execution. Consequently, this limitation affected and constrained the ability to obtain results. To address this, it was decided to develop and simulate these external artifacts (TSLAP, DC, and SP).

Starting with the SLA Integration module and the TSLA Mockup, the implementation was straightforward. The main objective of the SLA Integration module is to store metric-related information in the local database. Its key functionality involves communicating with the TSLA Mockup to retrieve this data. The Mockup was built with a simple API capable of receiving requests and responding with predefined metric data stored in a local file.

The Domain Integration module required some architectural changes compared to the initial design,

as it needed to mock the two external sources. These changes are illustrated in Fig. 28, primarily involving the addition of a queue to handle metric event tracking. This modification altered the initial logic, as the queue would not have been necessary if these events were sourced externally from the framework.

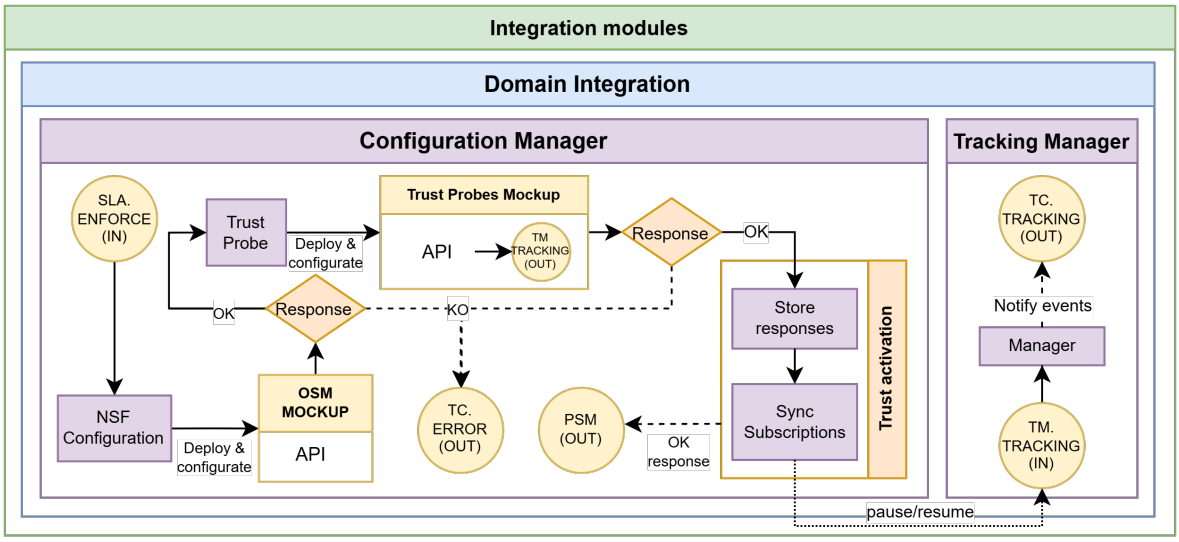


Figure 28: Refactorized Domain Integration Module

The implementation itself was relatively straightforward, with the exception of devising a reliable retry mechanism. This module requires access to two different endpoints for activating/deactivating configurations, and in the event of an error, the retry process must accurately determine which operations were completed, so they don't repeat itself. This was managed using two flags (one for each configuration) to track the state of these operations during retries. Additionally, the tracking manager service is designed to stop reading metric events whenever a new metric configuration is requested to avoid conflicts.

Regarding the mockups, the DC was the simplest to implement due to the limited information about its purpose. It consists of a basic API with one endpoint to deploy or undeploy (post or delete, respectively) NSF configurations, responding with either a positive or negative message depending on the case. The Probes mockup, however, required a more complex and robust logic to generate metric events upon receiving a new configuration. This was achieved using threads to ensure fast, simultaneous operations and a realistic simulation of the component's purpose. To facilitate this simulation, a local file database was created to store metric specifications.

As with the previous phases, a series of unit tests and integration tests were conducted at the end of this process to ensure that each component could communicate with each other correctly.

5.4.5 Phase 5: Workflow Testing

In this final phase, the focus was on testing and simulating the entire workflow from start to finish. This comprehensive testing approach differed from the previous integration tests, which only verified the communication between limited components. Now, all components were deployed for the respective contexts, either Domain or E2E.

The first step involved creating a detailed test plan for various use cases, divided into two main categories: Domain and E2E. Each test was documented with a name, description, set of preconditions, expected results, and an evidence section. The evidence section provided a step-by-step explanation of each test, supported by images that captured artifact logs or database changes to verify the activity and data modifications. A more detailed view of these tests is presented in subsection 6.2.

This approach ensures a comprehensive validation of the system's functionality in both Domain and E2E contexts, covering all critical workflows and operations.

5.5 Critical Decisions

This section aims to highlight some of the decisions made during the implementation of the TC and its components, as well as some key aspects.

One of the initial decisions, more architectural than code related, was the creation of integration modules. These modules serve as intermediaries, hosting APIs to facilitate the communication with external sources. This approach ensures that all data entering and leaving the system is routed through these modules, effectively isolating the other components. The modules are the ones mentioned on subsections 4.1.3, 4.2.4, 4.2.5 and 4.2.6.

Another decision was to design the service operating within the Orchestrator in both the E2E and Domain TCs as a batch service. This choice was made since it is expected that this artifact will handle a large volume of data within short timeframes, potentially leading to duplicated requests. The objective was to minimize unnecessary processing and optimize resource utilization. To achieve this, the service was implemented with a semaphore approach, either accumulating a certain number of requests or waiting for a specified period of time. Once either condition is met, all accumulated requests are dispatched for processing.

Another critical aspect of the system involves a scheduled service implemented in both the E2E and Domain contexts. This service, detailed on subsections 4.1.3 and 4.2.4, is referred to as a batch service due to its purpose. It is encapsulated within a Python file, along with scheduling information specified

in a Dockerfile. The service is programmed to run twice daily, with a one-hour interval, so it can exhibit a different error-handling behavior. During the first execution, any encountered errors are disregarded, allowing the originating processes to be retried during the subsequent run. However, during the second execution, errors are managed and logged for potential human intervention. Additionally, the affected processes are updated with a corresponding failed status.

All components within the TC, at both the E2E and Domain levels, are equipped with an auto-retry feature. This functionality allows each service or process to be automatically retried under certain conditions. The primary objective of this retry system is to provide a second opportunity for errors of the timeout type, which typically arise from unexpected events. Consequently, each process can be retried at least once, specifically when encountering a timeout error. In addition to this general retry mechanism, there exists a dedicated retrying system for execution requests. These requests are associated with the computation modules, wherein an execution object (ScoringExecution, ObjectiveExecution, or SubjectiveExecution) is generated to assist the calculation process. When errors occur within specific modules (TPS, TMP, and TSS), they undergo a retry process first through the general auto-retry feature and then through this specialized system. To do that are utilized specialized queues known as retry queues. The error handler modules play a crucial role in this process by capturing and redirecting errors, while the orchestrators analyze errors, verify their status, and process them as normal requests for retrying. Additionally, the number of retries allowed before logging and storing the error in the database can be adjusted via system parameters. To optimize efficiency, the strategy adopted involves leveraging the execution object to store essential information, allowing the calculation process to resume after an error occurs. Consequently, the information exchanged in the retry queue is kept minimal, typically consisting of an identifier that directs back to the stored object.

In addition to the decisions made during implementation, a notable choice was the development of a custom error handler. While typical error handlers aim to manage errors encountered during system operation, this implementation goes beyond the standard. It features a dedicated queue to efficiently route and manage errors throughout the system, ensuring they reach the appropriate components, notably the Orchestrator, for resolution. Furthermore, the error handler incorporates three distinct error classes, as discussed in section 4.4. These classes facilitate the implementation of the retry mechanism mentioned earlier, enhancing error management and database organization by categorizing errors based on their origin and nature.

Chapter 6

Tests and Results

This chapter describes the rigorous testing processes employed to validate the functionality of the developed components, presenting a comprehensive overview of the testing methodologies, test cases, and the corresponding results obtained during the evaluation phase. The aim of this testing phase was to ensure that each component operates as intended.

6.1 Unit and Integration Tests

The testing phase is initiated with a focus on unit testing, aiming to examine each individual artifact independently before integrating them into the system. These tests are done to guarantee the functionality of each service within the component, ensuring that the expected functions are invoked correctly, inputs and outputs are handled as anticipated, and that errors or exceptions are identified and correctly treated. Unit tests are conducted for every code modification, serving as a fundamental step prior to deploying the artifact with docker. Python's built-in unittest framework, allowed to automate and streamline the unit testing process, enhancing efficiency and reliability.

Regarding the the integration test cases, these ones were executed following the completion of both the E2E and Domain contexts. These tests played a crucial role in validating the seamless communication between microservices and ensuring connectivity with the database and the messaging server. To simulate interactions with external sources like the PSM or the SO component, the Postman's capability to generate mocks was utilized. At this point, each artifact was being deployed using Docker tools, including Docker Compose and Dockerfiles, to facilitate the execution of these tests.

After a successful execution, all unit and integration tests were meticulously documented in the test plan, accompanied by detailed step-by-step explanations. It were executed a total of 86 unit tests and 57 integration tests. A preview of both plans, unit and integration can be seen in Appendix B.1.1 and B.1.2, respectively.

6.2 Workflow Tests

Unlike integration tests, which aim to test connectivity between two artifacts sharing the message queue system and the database, workflow tests aim to ensure the correct functioning of all components involved in different workflows. These workflows include both E2E and Domain level processes. For each one of them, is defined a set of preconditions, such as the microservices to deploy and the external sources that need to be mocked. Additionally, the expected outcomes are defined, such as obtained responses and data changes/creation in the database. The tested workflows include, for the E2E level, activating and disabling provider trust calculation, provider LoT generation, and the process of sending the calculated provider LoTs. And for the Domain level, activating and disabling service trust calculation, service LoT generation, and the process of sending the calculated service LoTs. For some of these use cases, error scenarios were also tested to cover all possible outputs.

As mentioned, at the E2E level, four main tests were established based on the workflows presented in section 3.3. Each test also included an error version to test edge cases by inducing errors at specific points in the execution.

- Enable Provider Trust Calculation: This test corresponds to steps 1-2 of Fig. 8. It involves receiving an activation request for certain providers and registering those with an activated SC in the database.
- Disable Provider Trust Calculation: This test corresponds to steps 1-2 of Fig. 9. It involves receiving a deactivation request for certain providers and registering it in the database.
- Generate Provider LoT: This test corresponds to steps 5-6 of Fig. 8. It involves receiving a new Service LoT, which triggers the calculation of a new Provider LoT.
- Register Provider LoT: This test corresponds to steps 7-8 of Fig. 8. It involves gathering all the new Provider LoTs and sending them to the E2E PSM.

For the Domain level, four main tests were established, aligning with the previously presented E2E tests. Error cases are omitted here for simplicity. These tests also follow the workflows presented in section 3.3.

- Enable Service Trust Calculation: This test corresponds to steps 6-19 of Fig. 8. It involves receiving an activation request for a service, retrieving all necessary configurations, and initiating metric tracking for that service.

- Disable Service Trust Calculation: This test corresponds to steps 5-14 of Fig. 9. It involves receiving a deactivation request for a service and suspending all related activities for that service.
- Generate Service LoT: This test corresponds to step 1 of Fig. 10. It involves receiving a metric event for a service, initiating the calculation process of a new Service LoT, and registering it in the database.
- Register Service LoT: This test corresponds to steps 2-3 of Fig. 10. It involves gathering all new Service LoTs and sending them to the PSM.

It were executed a total of 14 workflow tests and a preview of the test plan can be seen in Appendix B.1.3.

6.3 Practical Example

In this final section, the functionality of the system is demonstrated by presenting some results and interactions. To keep the focus on the main goal and minimize the amount of information the reader needs to process, this example omits several steps of the process. Details such as subscription specifics for the metrics, execution logs for the metrics processing and the LoT calculation, and some interactions with external sources have been streamlined. Three use cases from the domain context were chosen as examples: Enable Service Trust Calculation, Generate Service LoT, and Register Service LoT. These use cases follow the most common workflow required for all new registrations, starting with activating the process for a specific service, followed by generating a new LoT triggered by metric events, and finally, registering this new value into the blockchain.

The process begins when the PSM Integration API receives an activation request. This can be simulated using Postman, Fig. 29.

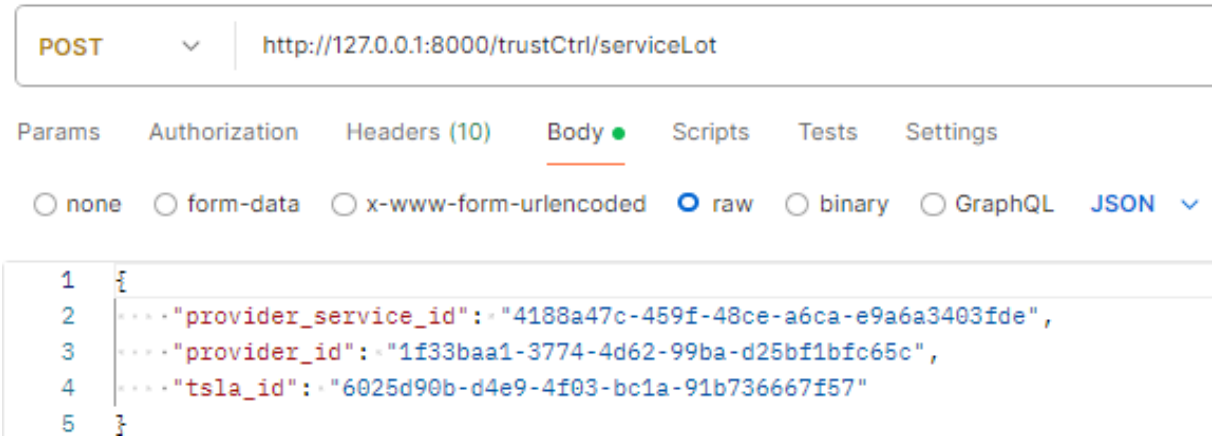


Figure 29: Simulation of an activation request using Postman.

After obtaining the information from the request, the first step is to create this new service in the database with an active status, Fig. 30.

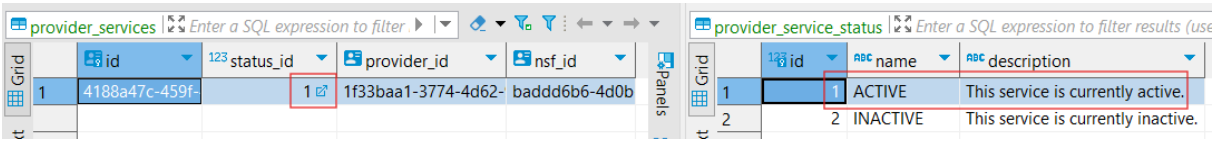


Figure 30: New service created in the database with active status.

The next step involves retrieving the SLA, which establishes the client trust requirements for this service, such as which metrics to monitor and the corresponding thresholds, etc. This information is stored in the database after being retrieved from the external SLA component and is managed by the SLA Integration entity. In this example, we monitor the latency and throughput metrics, with the SLA specifying an OV for latency between 0 and 10 ms and for throughput, at least 1 Mbps. Fig. 31 demonstrates how this information is organized in the database.

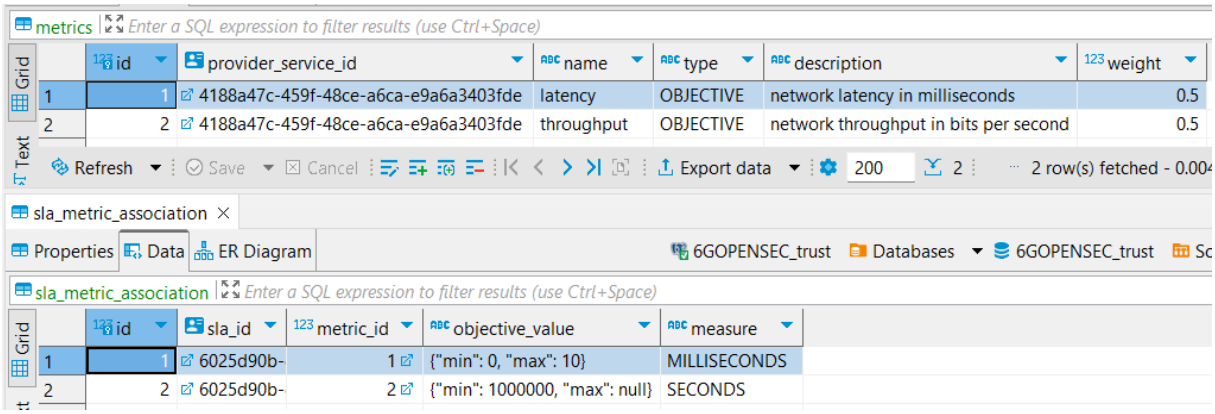


Figure 31: SLA requirements for latency and throughput metrics.

In the Domain Integration component, the deployment of the NSF (DC entity) and the probes configuration (SP entity) is done, allowing the service metrics to start being received and monitored. At this point, the first use case is completed.

The next stage begins when the Domain Integration receives metric events. In this case, the tracking manager inside the Domain Integration component receives two events for each metric. As we can see in Fig. 32, the latency metric did not suffer any penalization as the received values (8 and 9) were within the requested limits ($[0, 10]$). However, the throughput values (484 and 170) were far below the requested limit ($[1000, +\infty[$), resulting in a composite trust value of 0.

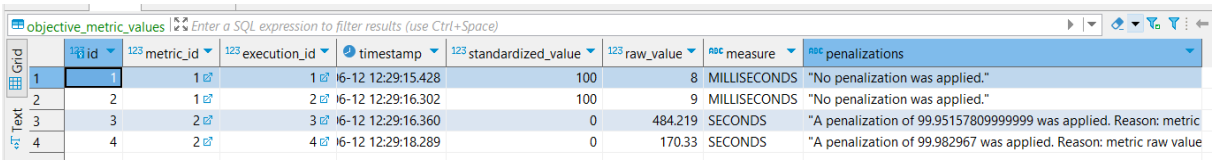


Figure 32: Received metric events and their evaluation.

These new metric values trigger the calculation of a new Service LoT, which is computed as $100 * 0.5 + 0 * 0.5 = 50$, given that both metrics have equal weight. The new LoT is then registered in the database, Fig. 33, concluding the second use case.

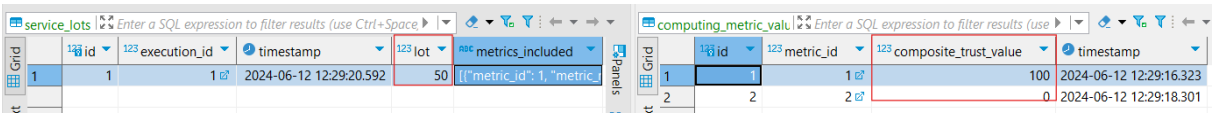


Figure 33: New Service LoT calculated and registered in the database.

The final step and use case focus on delivering this new value to the blockchain through the PSM Integration component. The LoT value is stored in the database with a pending status, a flag that enables this transaction to occur. The automatic cron service gathers all pending LoTs and sends them to the PSM entity. After the process, the status is changed to active as showed in Fig. 34.

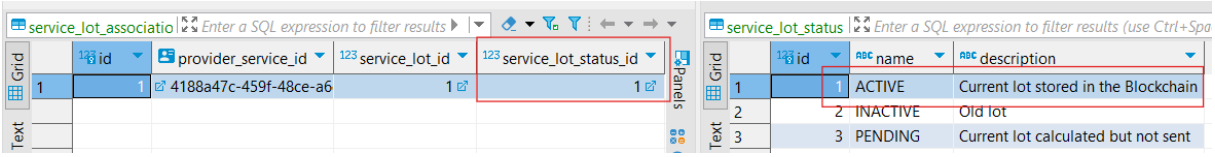


Figure 34: New Service LoT with status updated to active.

In summary, this practical example demonstrates part of the functionality of the system through three key use cases: Enable Service Trust Calculation, Generate Service LoT, and Register Service LoT. The

workflow begins with the activation of the service, where the system retrieves and stores necessary SLA configurations. It then proceeds to monitor and evaluate service metrics, handling both compliant and non-compliant metric events. Finally, the calculated Service LoT is securely registered in the blockchain.

Chapter 7

Conclusions and Future Work

This last chapter presents the key aspects and outcomes of the work done, and identifies potential areas for future exploration and improvement.

7.1 Conclusions

This dissertation explores the design and implementation of a TC architecture within a pre-established TM framework for 6G networks. As the next generation networks walks towards more open and disaggregated architectures, trust management becomes a crucial point for maintaining security and service quality across multiple stakeholders. The proposed TC aims to mitigate this challenge by making use of Blockchain technology to provide a transparent and reliable trust management system.

The implementation involved several key phases, including the development of functional and integration modules and the simulation of external components to validate the system's functionality. Despite challenges, particularly in obtaining real-time data from external sources, a successfully simulation of these components ensured the system's functionality.

A significant aspect of this work was the development and execution of comprehensive test cases. These tests covered various use case scenarios, including enabling and disabling trust calculations, generating and registering LoTs for both services and providers, and handling errors to ensure the system robustness.

In conclusion, this thesis presents a foundational framework for trust management in 6G networks, addressing key challenges and leaving a solid base for future enhancements. The proposed TC architecture and its integration with Blockchain technology offer a promising solution for managing trust in a complex, multi-stakeholder environment. As 6G networks continue to develop, the importance of robust trust management systems will only grow, and this work provides a crucial step towards achieving that goal.

7.2 Future Work

Although this work presents a complex architecture and operational implementation for the TC within the 6G TM framework, there are several areas for further research and development to enhance its effectiveness and scope. Three of them should be pointed out:

- Improvement of the Trust Level Model: The current trust level model is relatively simple and linear, designed primarily to ensure the system's functionality and operational capability. Future work should focus on developing a more sophisticated algorithm to improve the calculation of the LoT. By incorporating more complex factors into the trust model, we could achieve more accurate and reliable trust evaluations. This enhancement will contribute to a more robust trust management system capable of meeting the demands of advanced 6G networks.

- Analysis of Potential Actors for the DC Entity: a critical next step should be to study and analyze potential actors for the DC entity. This involves identifying key players and understanding their roles, capabilities, and interactions within the 6G network ecosystem. By getting deeper insights into these actors, it can be possible to design a more efficient and robust Domain Integration module, facilitating seamless communication with the DC and optimizing the NSF deployment and configuration. Some examples include works like [46] using Open Source Mano or [47] using TeraflowSDN.

- Incorporation of Subjective Metric Analysis: the incorporation of a fully functional component dedicated to handling subjective metrics. Currently, the trust management system primarily focuses on objective metrics. However, subjective metrics, such as client feedback and qualitative assessments, could play a vital role in providing a comprehensive view of the service provider performance. To address this, the development of a component that utilizes advanced text-based AI technologies to analyze and interpret client feedback can be studied. By leveraging natural language processing and ML techniques, this component could effectively translate subjective reports into actionable trust metrics.

In summary, these three areas of future work aim to improve the robustness and comprehensiveness of the TC. By studying more sophisticated algorithms for the LoT model and potential actors for the DC and incorporating AI-driven subjective metric analysis, we can further advance the trust management framework, ensuring it meets the evolving demands of 6G networks.

Bibliography

[1] S. Elmadani, S. Hariri, and S. Shao. Blockchain based methodology for zero trust modeling and quantification for 5g networks. In *2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–9, Los Alamitos, CA, USA, dec 2022. IEEE Computer Society. doi: 10.1109/AICCSA56895.2022.10017914. URL https://doi.ieeecomputersociety.org/10.1109/AICCSA56895.2022.10017914.

[2] Reza Soltani, Marzia Zaman, Rohit Joshi, and Srinivas Sampalli. Distributed ledger technologies and their applications: A review. *Applied Sciences*, 12(15), 2022. ISSN 2076-3417. doi: 10.3390/app12157898. URL https://www.mdpi.com/2076-3417/12/15/7898.

[3] Massimo Di Pierro. What is the blockchain? *Computing in Science & Engineering*, 19(5):92–95, 2017. doi: 10.1109/MCSE.2017.3421554.

[4] Karthik Kumar Vaigandla, RadhaKrishna Karne, Mounika Siluveru, and Madhavi Kesoju. Review on blockchain technology : Architecture, characteristics, benefits, algorithms, challenges and applications. *Mesopotamian Journal of CyberSecurity*, 2023, 2023. doi: 10.58496/MJCS/2023/012. URL https://mesopotamian.press/journals/index.php/CyberSecurity/article/view/68.

[5] Pol Alemany, Raul Munoz, Josep Martrat, Antonio Pastor, Rodrigo Diaz, Diego Lopez, Ramon Casellas, Ricardo Martinez, and Ricard Vilalta. Blockchain-based trust management collaborative system for transport multi-stakeholder scenarios. *Journal of Optical Communications and Networking*, 15 (8):488–496, 2023. doi: 10.1364/JOCN.486503.

[6] Pol Alemany, Ricard Vilalta, Raul Muñoz, Ramon Casellas, and Ricardo Maríinez. Peer-to-peer blockchain-based nfv service platform for end-to-end network slice orchestration across multiple nfvi domains. In *2020 IEEE 3rd 5G World Forum (5GWF)*, pages 151–156, 2020.

[7] D. Moreira, J. García, J. Cunha, and J. García. 6g networks: Trust controller architecture proposal. In *2024 15th International Conference on Network of the Future (NoF)*, pages 1–5, 2024.

[8] P. Alemany, R. Muñoz, R. Vilalta, Ll. Gifre, R. Martínez, R. Casellas, M. Castro, P. Ferreira, D. Moreira, J. García, J. Cunha, I. Núñez, G. Gómez, S. Castro, A. Pastor, and D. López. Security and trust in open and disaggregated 6g networks. In *2024 24th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4, 2024. doi: 10.1109/ICTON62926.2024.10647935.

[9] Faisal Tariq, Muhammad R. A. Khandaker, Kai-Kit Wong, Muhammad A. Imran, Mehdi Bennis, and Merouane Debbah. A speculative study on 6g. *IEEE Wireless Communications*, 27(4):118–125, 2020. doi: 10.1109/MWC.001.1900488.

[10] Tongyi Huang, Wu Yang, Jun Wu, Jin Ma, Xiaofei Zhang, and Daoyin Zhang. A survey on green 6g network: Architecture and technologies. *IEEE Access*, 7:175758–175768, 2019. doi: 10.1109/ACCESS.2019.2957648.

[11] TechTarget. 6g networks - what is 6g & when is it available?, 2023. URL https://www.techtarget.com/searchnetworking/definition/6G.

[12] Nokia. 6g explained | nokia, n/d. URL https://www.nokia.com/about-us/newsroom/articles/6g-explained/.

[13] NGMN Alliance. 6g position statement: An operator view. *NGMN Publications*, 2023.

[14] Ioannis Tomkos, Dimitrios Klonidis, Evangelos Pikasis, and Sergios Theodoridis. Toward the 6g network era: Opportunities and challenges. *IT Professional*, 22(1):34–38, 2020. doi: 10.1109/MITP.2019.2963491.

[15] Samar Elmeadawy and Raed M. Shubair. 6g wireless communications: Future technologies and research challenges. In *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pages 1–5, 2019. doi: 10.1109/ICECTA48151.2019.8959607.

[16] Pawan Meena, Monti Babulal Pal, Praphula Kumar Jain, and Rajendra Pamula. 6g communication networks: Introduction, vision, challenges, and future directions. *Wirel. Pers. Commun.*, 125(2): 1097–1123, jul 2022. ISSN 0929-6212. doi: 10.1007/s11277-022-09590-5. URL https://doi.org/10.1007/s11277-022-09590-5.

[17] Jagadeesha R. Bhat and Salman A. Alqahtani. 6g ecosystem: Current status and future perspective. *IEEE Access*, 9:43134–43167, 2021. doi: 10.1109/ACCESS.2021.3054833.

[18] Yiying Wang, Xin Kang, Tieyan Li, Haiguang Wang, Cheng-Kang Chu, and Zhongding Lei. Six-trust for 6g: Toward a secure and trustworthy future network. *IEEE Access*, 11:107657–107668, 2023. doi: 10.1109/ACCESS.2023.3321114.

[19] Chafika Benzaïd, Tarik Taleb, and Muhammad Zubair Farooqi. Trust in 5g and beyond networks. *IEEE Network*, 35(3):212–222, 2021. doi: 10.1109/MNET.011.2000508.

[20] Wenjuan Li and Lingdi Ping. Trust model to enhance security and interoperability of cloud environment. In Martin Gilje Jaatun, Gansen Zhao, and Chunming Rong, editors, *Cloud Computing*, pages 69–79, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-10665-1.

[21] Volker Ziegler, Peter Schneider, Harish Viswanathan, Michael Montag, Satish Kanugovi, and Ali Rezaki. Security and trust in the 6g era. *IEEE Access*, 9:142314–142327, 2021. doi: 10.1109/ACCESS.2021.3120143.

[22] Zainab M. Aljazzaf, Mark Perry, and Miriam A. M. Capretz. Trust metrics for services and service providers. In *International Conference on Internet and Web Applications and Services*, 2011. URL https://api.semanticscholar.org/CorpusID:17895285.

[23] Upul Jayasinghe, Gyu Myoung Lee, and Aine MacDermott. Trust-based data controller for personal information management. In *2018 International Conference on Innovations in Information Technology (IIT)*, pages 123–128, 2018. doi: 10.1109/INNOVATIONS.2018.8605979.

[24] European Union. General data protection regulation (gdpr). *Official Journal of the European Union*, L119:pp. 1–88, 2016.

[25] Bahar Farahani, Farshad Firouzi, and Markus Luecking. The convergence of iot and distributed ledger technologies (dlt): Opportunities, challenges, and solutions. *Journal of Network and Computer Applications*, 177:102936, 2021. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2020.102936. URL https://www.sciencedirect.com/science/article/pii/S1084804520303945.

[26] South Carolina Emerging Tech Association. Distributed ledger technologies, 2020. URL https://sceta.io/distributed-ledger-technologies/.

[27] Zehui Xiong, Yang Zhang, Nguyen Cong Luong, Dusit Niyato, Ping Wang, and Nadra Guizani. The best of both worlds: A general architecture for data management in blockchain-enabled internet-of-things. *IEEE Network*, 34(1):166–173, 2020. doi: 10.1109/MNET.001.1900095.

[28] Hossein Shafagh, Lukas Burkhalter, Anwar Hithnawi, and Simon Duquennoy. Towards blockchain-based auditable storage and sharing of iot data. In *Proceedings of the 2017 on Cloud Computing Security Workshop*, CCSW '17, page 45–50, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450352048. doi: 10.1145/3140649.3140656. URL https://doi.org/10.1145/3140649.3140656.

[29] Niclas Kannengießer, Sebastian Lins, Tobias Dehling, and Ali Sunyaev. Trade-offs between distributed ledger technology characteristics. *ACM Comput. Surv.*, 53(2), May 2020. ISSN 0360-0300. doi: 10.1145/3379463. URL https://doi.org/10.1145/3379463.

[30] Mohammad Jabed Morshed Chowdhury, MD. Sadek Ferdous, Kamanashis Biswas, Niaz Chowdhury, A. S. M. Kayes, Mamoun Alazab, and Paul Watters. A comparative analysis of distributed ledger technology platforms. *IEEE Access*, 7:167930–167943, 2019. doi: 10.1109/ACCESS.2019.2953729.

[31] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.

[32] Haojun Liu, Xinbo Luo, Hongrui Liu, and Xubo Xia. Merkle tree: A fundamental component of blockchains. In *2021 International Conference on Electronic Information Engineering and Computer Science (EIECS)*, pages 556–561, 2021. doi: 10.1109/EIECS53707.2021.9588047.

[33] Medium Teemu Kanstrén. Merkle trees: Concepts and use cases, 2021. URL https://medium.com/coinmonks/merkle-trees-concepts-and-use-cases-5da873702318.

[34] Tri Nguyen, Ngoc Tran, Lauri Loven, Juha Partala, M-Tahar Kechadi, and Susanna Pirttikangas. Privacy-aware blockchain innovation for 6g: Challenges and opportunities. In *2020 2nd 6G Wireless Summit (6G SUMMIT)*, pages 1–5, 2020. doi: 10.1109/6GSUMMIT49458.2020.9083832.

[35] Guntur Dharma Putra, Volkan Dedeoglu, Salil S Kanhere, and Raja Jurdak. Toward blockchain-based trust and reputation management for trustworthy 6g networks. *IEEE Network*, 36(4):112–119, 2022. doi: 10.1109/MNET.011.2100746.

[36] European Telecommunications Standards Institute. Permissioned distributed ledger (pdl);specification of requirements for smart contracts' architecture and security, 2021. URL https://www.etsi.org/deliver/etsi_gs/PDL/001_099/011/01.01.01_60/gs_PDL011v010101p.pdf.

[37] European Telecommunications Standards Institute. Zero-touch network and service management (zsm); reference architecture, 2019. URL https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/002/01.01.01_60/gs_ZSM002v010101p.pdf.

[38] European Telecommunications Standards Institute. Zero touch network & service management (zsm), 2024. URL https://www.etsi.org/technologies/zero-touch-network-service-management.

[39] European Telecommunications Standards Institute. Permissioned distributed ledger (pdl); reputation management, 2023. URL https://www.etsi.org/deliver/etsi_gs/PDL/001_099/015/01.01.01_60/gs_PDL015v010101p.pdf.

[40] Bithika Khargharia, Haoting Luo, Youssif Al-Nashif, and Salim Hariri. Appflow: Autonomic performance-per-watt management of large-scale data centers. In *2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, pages 103–111, 2010. doi: 10.1109/GreenCom-CPSCom.2010.103.

[41] Ben Niu, Wei You, Hongbo Tang, and Xiaolei Wang. 5g network slice security trust degree calculation model. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pages 1150–1157, 2017. doi: 10.1109/CompComm.2017.8322724.

[42] V. Casola, L. Coppolino, A. Mazzeo, N. Mazzocca, and M. Rak. Design and implementation of truman, a trust manager component for distributed systems. In *Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU'06)*, pages 7 pp.–40, 2006. doi: 10.1109/SECPERU.2006.7.

[43] Sandro Rodriguez Garzon, Hakan Yildiz, and Axel Küpper. Towards decentralized identity management in multi-stakeholder 6g networks. In *2022 1st International Conference on 6G Networking (6GNet)*, pages 1–8, 2022. doi: 10.1109/6GNet54646.2022.9830163.

[44] 6G-OPENSEC project. Deliverable E4 – Trust Manager architecture and interfaces. Technical report, 6G-OPENSEC-TRUST, September 2023.

[45] 6G-OPENSEC project. Deliverable E6 – Trust Manager Systems Preliminary Implementation. Technical report, 6G-OPENSEC-TRUST, April 2024.

[46] Panagiotis Karamichailidis, Kostas Choumas, and Thanasis Korakis. Enabling multi-domain orchestration using open source mano, openstack and opendaylight. In *2019 IEEE International Symposium*

on Local and Metropolitan Area Networks (LANMAN), pages 1–6, 2019. doi: 10.1109/LANMAN.2019.8847036.

[47] R. Vilalta, P. Alemany, Ll. Gifre, R. Martínez, R. Casellas, and R. Muñoz. End-to-end inter-domain transport network slice management using dlt-enabled cloud-based sdn controllers. In *2023 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, 2023. doi: 10.1364/OFC.2023.Tu3D.3.

Appendix A
Support work

Auxiliary results which are not main-stream.

A.1 External Integrations E2E Level

A.1.1 Exposed interfaces

Table 3: E2E Trust Controller exposed interface (1/3)

Operation name: /trustCtrl/providerLot		
Description	POST. Activates trust capabilities smart contract	
Input Parameters	Type	Description
Object	Json	List of providers
Output Parameters	Type	Description
Object	Json	List of potential providers with sc activated

Table 4: E2E Trust Controller exposed interface (2/3)

Operation name: /trustCtrl/providerLot		
Description	DELETE. Deactivates trust capabilities smart contract	
Input Parameters	Type	Description
Object	Json	List of providers and deactivation cause
Output Parameters	Type	Description
–	–	–

Table 5: E2E Trust Controller exposed interface (3/3)

Operation name: /trustCtrl/providerLot/<provider_service_id>		
Description	POST. Receive a Service LoT	
Input Parameters	Type	Description
Object	Json	Service LoT data
Output Parameters	Type	Description
–	–	–

A.1.2 Consumed Interfaces

Table 6: E2E PSM consumed interface (1/3)

Operation name: /opensecTrust/v1/psm/slice/serviceProvider		
Description	POST. Retrieve list of providers to activate.	
Input Parameters	Type	Description
Object	Json	List of providers
Output Parameters	Type	Description
Object	Json	List of potential providers with sc activated

Table 7: E2E PSM consumed interface (2/3)

Operation name: /opensecTrust/v1/psm/slice/serviceProvider		
Description	DELETE. Retrieve list of providers to deactivate.	
Input Parameters	Type	Description
Object	Json	List of providers and deactivation cause
Output Parameters	Type	Description
Object	Json	List of providers to deactivate

Table 8: E2E PSM consumed interface (3/3)

Operation name: /opensecTrust/v1/psm/slice/e2elot		
Description	POST. Send provider lot	
Input Parameters	Type	Description
Object	Json	Provider ID and LoT
Output Parameters	Type	Description
–	–	–

A.2 External Integrations Domain Level

A.2.1 Exposed Interfaces

Table 9: Domain TMP exposed interface (1/2)

Operation name: /trustCtrl/serviceLot/<provider_service_id>		
Description	GET. Provides a list of available subjective metrics	
Input Parameters	Type	Description
Object	Json	ID of the Provider Service
Output Parameters	Type	Description
Object	Json	List of available subjective metrics associated to provider service given

Table 10: Domain TMP exposed interface (2/2)

Operation name: /trustCtrl/serviceLot/feedback		
Description	POST. Receive direct feedback from users, including ratings and comments, associated with a particular provider	
Input Parameters	Type	Description
Object	Json	Subjective metric evaluation info
Output Parameters	Type	Description
–	–	–

Table 11: Domain PSM exposed interface (1/2)

Operation name: /trustCtrl/serviceLot		
Description	POST. Initiate a mechanism for enabling trust activation score	
Input Parameters	Type	Description
Object	Json	Domain trust activation request
Output Parameters	Type	Description
–	–	–

Table 12: Domain PSM exposed interface (2/2)

Operation name: /trustCtrl/serviceLot		
Description	DELETE. Initiate a mechanism for disable trust activation score	
Input Parameters	Type	Description
Object	Json	Domain trust deactivation request
Output Parameters	Type	Description
–	–	–

A.2.2 Consumed Interfaces

Table 13: Domain PSM consumed interface

Operation name: /trustCtrl/serviceLot		
Description	POST. Send Service LoT	
Input Parameters	Type	Description
Object	Json	Provider and Service ID and Service LoT
Output Parameters	Type	Description
–	–	–

Table 14: Domain TSLA consumed interface

Operation name: /inventory/dtsla		
Description	POST. Retrieve service metrics related information for a TSLA of a service.	
Input Parameters	Type	Description
Object	Json	TSLA ID
Output Parameters	Type	Description
Object	Json	List of metrics and configurations.

Table 15: Domain Controller consumed interface (1/2)

Operation name: /nsf_activate_config		
Description	POST. Activate NSF configuration.	
Input Parameters	Type	Description
Object	Json	NSF ID
Output Parameters	Type	Description
–	–	–

Table 16: Domain Controller consumed interface (2/2)

Operation name: /nsf_deactivate_config		
Description	DELETE. Deactivate NSF configuration.	
Input Parameters	Type	Description
Object	Json	NSF ID
Output Parameters	Type	Description
–	–	–

A.3 Workflows

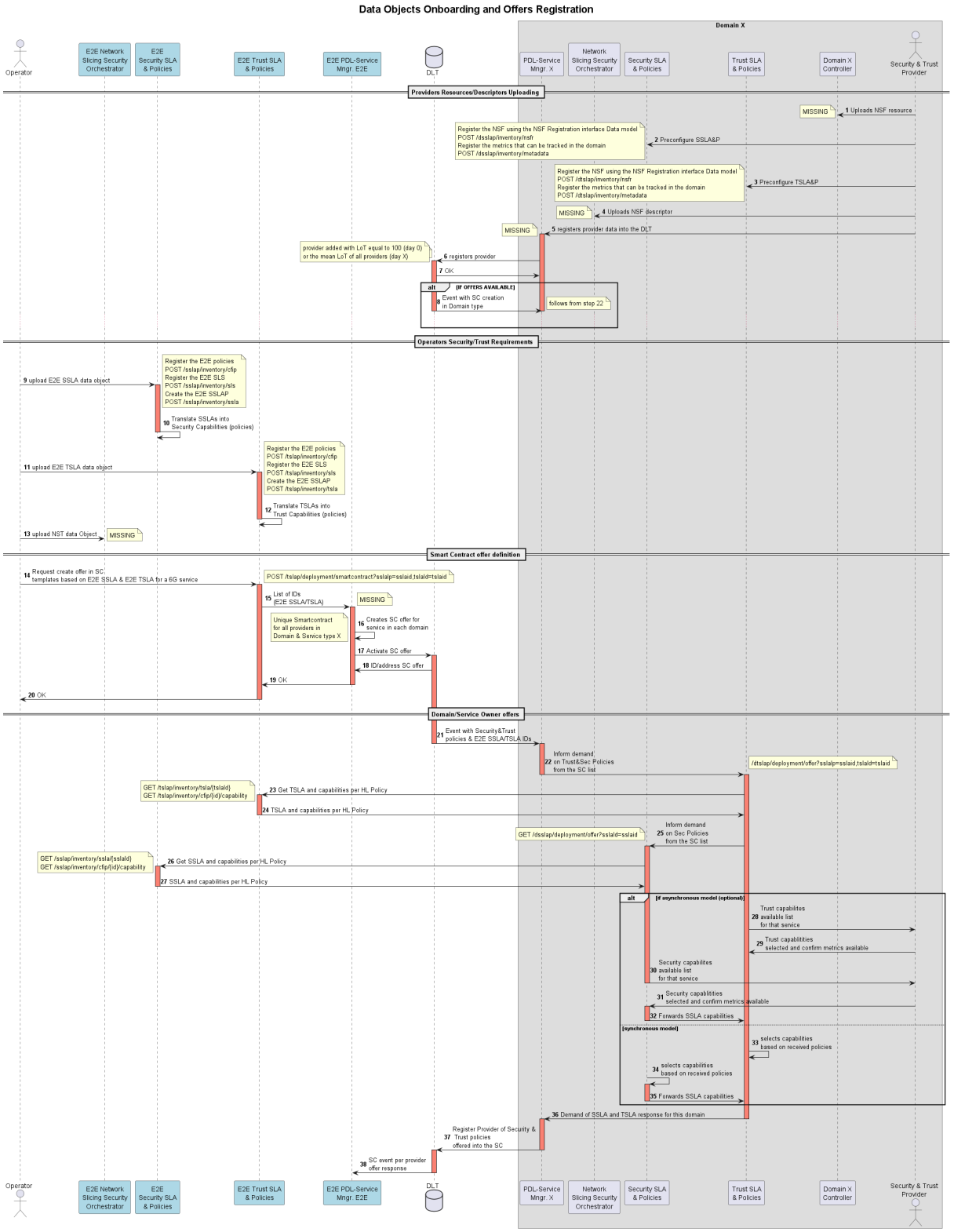


Figure 35: Data Objects On boarding and Offers Registration Workflow

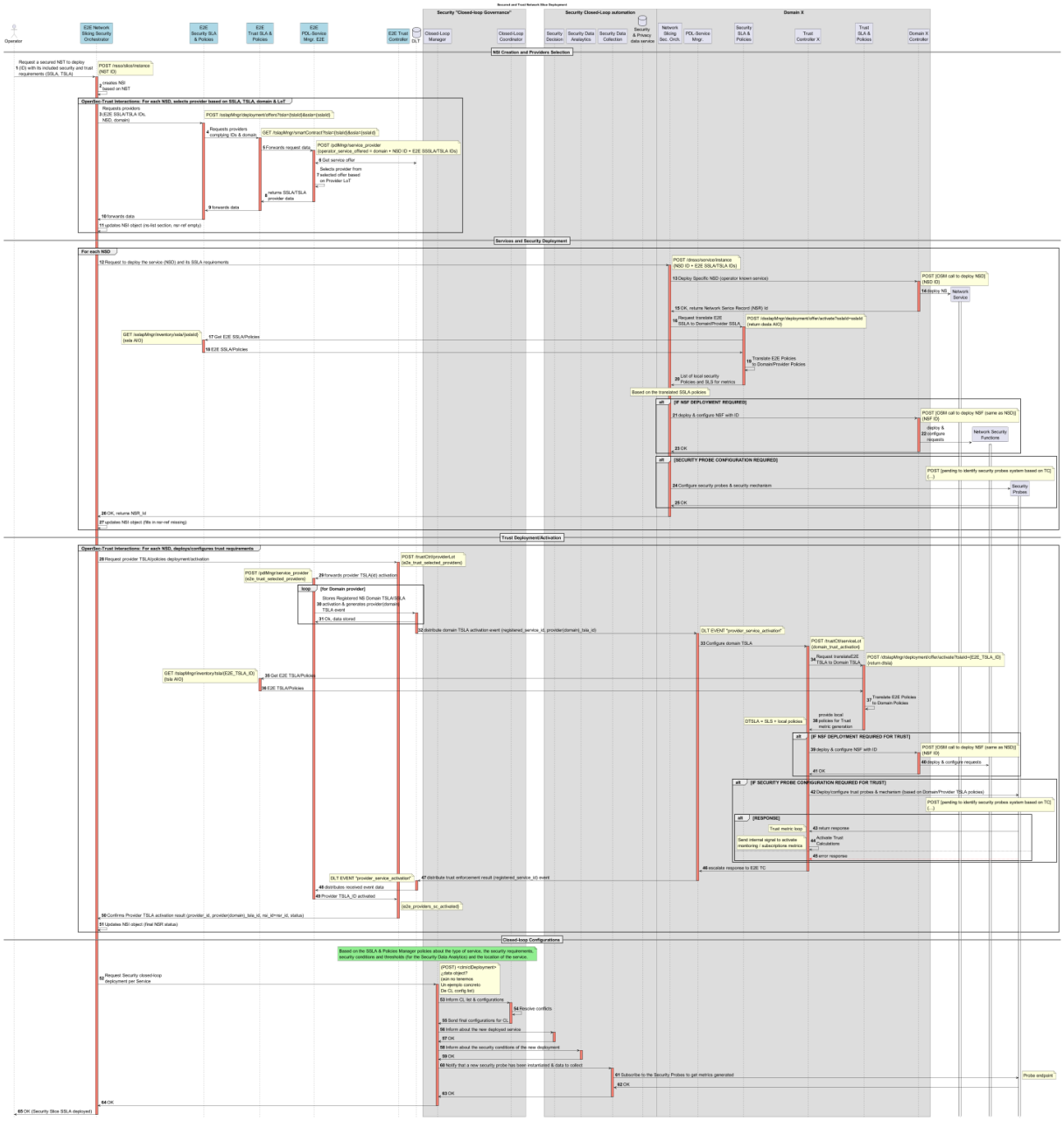


Figure 36: Security and Trust Deployment Workflow

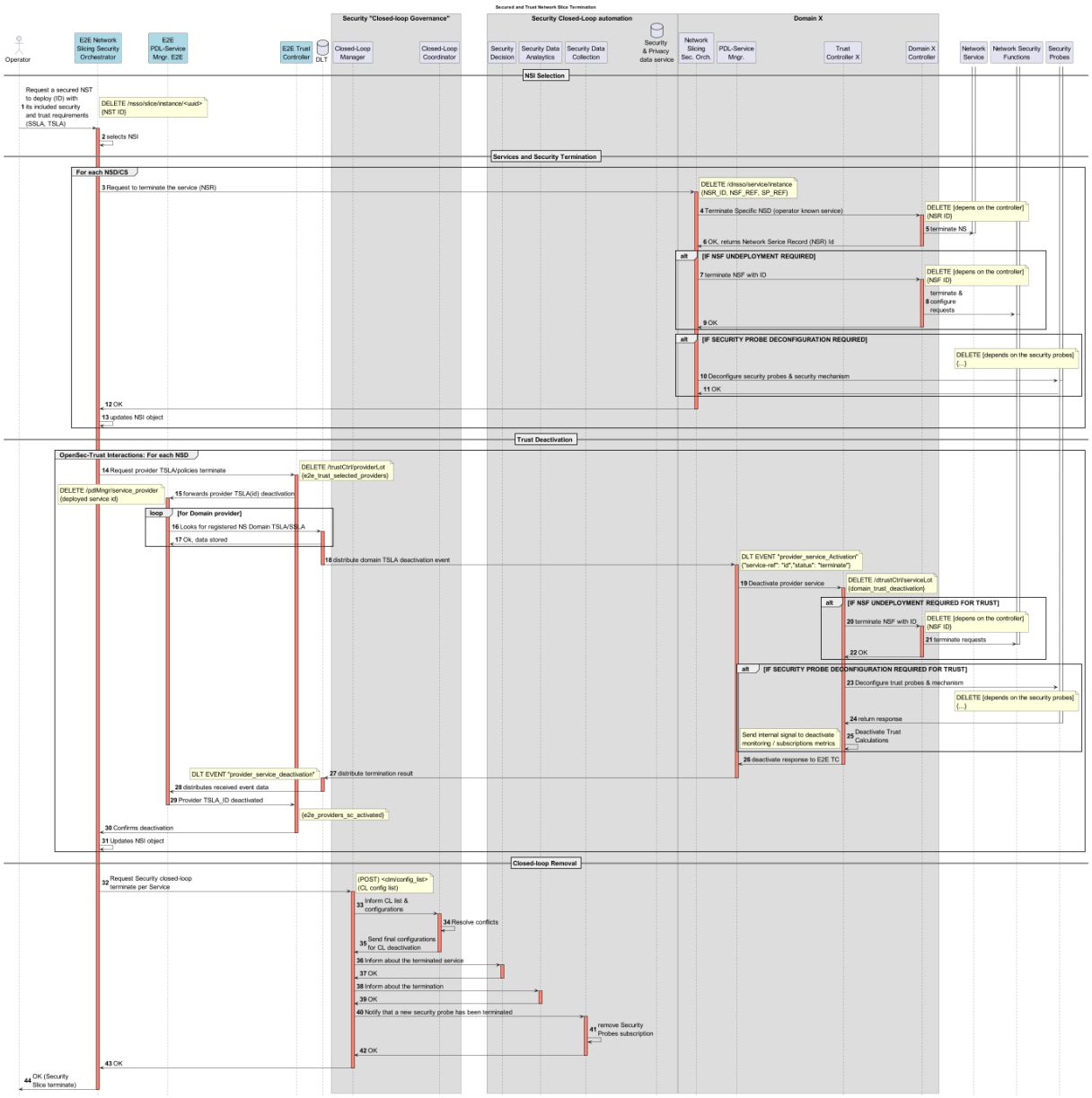


Figure 37: Security and Trust Termination Workflow

A.4 Data Objects

A.4.1 External Data Objects

Table 17: Data Object: e2e_trust_selected_providers_do (activation use case)

Field	Type	Must	Description
nsd_id	UUID	Yes	ID of the NSD
selected_providers	Providers	Yes	List of providers

Table 18: Data Object: e2e_trust_selected_providers_do (deactivation use case)

Field	Type	Must	Description
nsd_id	UUID	Yes	ID of the NSD
selected_providers	Providers	Yes	List of providers
deactivation_cause	DeactivationCauseEnum	Yes	Cause of the deactivation

Table 19: Data Object: Type Providers

Field	Type	Must	Description
id	UUID	Yes	ID of the Provider
ssla_id	UUID	Yes	ID of the SSLA
tsla_id	UUID	Yes	ID of the TSLA
nsr_id	UUID	Yes	ID of the NSR

Table 20: Data Object: e2e_providers_sc_activated_do (list of)

Field	Type	Must	Description
provider_id	UUID	Yes	ID of the Provider
provider_services	List of UUID	Yes	List of provider services IDs activaded

Table 21: Data Object: e2e_provider_lot_do

Field	Type	Must	Description
provider_id	UUID	Yes	ID of the Provider
lot	INT	Yes	Level of Trust

Table 22: Data Object: e2e_service_lot_do

Field	Type	Must	Description
provider_id	UUID	Yes	ID of the Provider
provider_service_id	UUID	Yes	ID of the provider Service
lot	INT	Yes	Level of Trust
timestamp	DATETIME	Yes	Record creation time

Table 23: Data Object: domain_trust_activation_do

Field	Type	Must	Description
provider_service_id	UUID	Yes	ID of the provider Service
provider_id	UUID	Yes	ID of the Provider
e2e_xsla_id	UUID	Yes	ID of the E2E TSLA
service_lot	INT	Yes	Level of Trust

Table 24: Data Object: domain_trust_deactivation_do

Field	Type	Must	Description
provider_service_id	UUID	Yes	ID of the provider Service
provider_id	UUID	Yes	ID of the Provider
cause	DeactivationCauseEnum	Yes	Cause of the deactivation

Table 25: Data Object: dtsla_do

Field	Type	Must	Description
provider_service_id	UUID	Yes	ID of the provider Service
metrics	List of dicts	Yes	List of metrics with the respective information

Table 26: Data Object: domain_service_lot_do

Field	Type	Must	Description
provider_id	UUID	Yes	ID of the Provider
provider_service_id	UUID	Yes	ID of the provider Service
lot	INT	Yes	Level of Trust

Table 27: Data Object: domain_available_metrics_do

Field	Type	Must	Description
provider_service_id	UUID	Yes	ID of the provider Service
available_metrics	AvailableMetrics	Yes	List of available subjective metrics

Table 28: Data Object: Type AvailableMetrics

Field	Type	Must	Description
id	BIGINT	Yes	ID of the metric
name	STRING	Yes	Name of the metric
description	STRING	Yes	Description of the metric

A.4.2 Internal Data Objects

Table 29: Data Object: e2e_tc_slot_do

Field	Type	Must	Description
provider_service_id	UUID	Yes	ID of the provider Service
provider_id	UUID	Yes	ID of the Provider
timestamp	DATETIME	Yes	Record creation time
lot	INT	Yes	Level od Trust

Table 30: Data Object: e2e_tc_tps_do and e2e_tps_do (first request)

Field	Type	Must	Description
provider_id	UUID	Yes	ID of the Provider

Table 31: Data Object: e2e_tc_retries_do

Field	Type	Must	Description
provider_id	UUID	Yes	ID of the Provider
execution_id	BIGINT	Yes	ID of the Execution
timestamp	DATETIME	Yes	Record creation time

Table 32: Data Object: domain_psm_do

Field	Type	Must	Description
request_id	UUID	Yes	ID of the async PSM request
response	JSON	Yes	Response message
http_status	INT	Yes	Http status code

Table 33: Data Object: domain_sla_config_do

Field	Type	Must	Description
request_id	UUID	Yes	ID of the async PSM request
provider_service_id	UUID	Yes	ID of the provider Service
tsla_id	UUID	Yes	ID of the E2E TSLA

Table 34: Data Object: domain_sla_enforce_do

Field	Type	Must	Description
request_id	UUID	Yes	ID of the async PSM request
provider_service_id	UUID	Yes	ID of the provider service
metrics	List of dicts	Yes	List of metrics with the respective information

Table 35: Data Object: domain_tc_tracking_do (objective metric)

Field	Type	Must	Description
metric_id	BIGINT	Yes	ID of the metric
metric_type	MetricTypeEnum	Yes	Type of the metric
raw_value	FLOAT	Yes	Value of the metric
timestamp	DATETIME	Yes	Record creation time
measure	MeasureEnum	Yes	Measure of the metric

Table 36: Data Object: domain_tc_tracking_do (subjective metric)

Field	Type	Must	Description
metric_id	BIGINT	Yes	ID of the metric
metric_type	MetricTypeEnum	Yes	Type of the metric
raw_value	STRING	Yes	Value of the metric
timestamp	DATETIME	Yes	Record creation time
scale	DICT	Yes	Dict(min,max) Scale of the evaluation
source	STRING	Yes	Source of the subjective metric

Table 37: Data Object: domain_tc_tss_do and domain_tss_do (first request)

Field	Type	Must	Description
provider_service_id	UUID	Yes	ID of the provider Service

Table 38: Data Object: domain_tc_retries_do (TSS module use case)

Field	Type	Must	Description
module	ModulesEnum	Yes	Module from where it came
provider_service_id	UUID	Yes	ID of the provider Service
execution_id	BIGINT	Yes	ID of the Execution
timestamp	DATETIME	Yes	Record creation time

Table 39: Data Object: domain_tc_retries_do (remaining modules use case)

Field	Type	Must	Description
module	ModulesEnum	Yes	Module from where it came
execution_id	BIGINT	Yes	ID of the Execution

Table 40: Data Object: domain_tom_do (first request)

Field	Type	Must	Description
metric_id	BIGINT	Yes	ID of the Metric
raw_value	FLOAT	Yes	Value of the Metric
measure	MeasureEnum	Yes	Measure of the Metric
timestamp	DATETIME	Yes	Record creation time

Table 41: Data Object (retry requests): domain_tom_do, domain_tsm_do, domain_tss_do and e2e_tps_do

Field	Type	Must	Description
execution_id	BIGINT	Yes	ID of the Execution

Table 42: Data Object: domain_tsm_do (first request)

Field	Type	Must	Description
metric_id	BIGINT	Yes	ID of the Metric
raw_value	STRING	Yes	Value of the Metric
scale	DICT	Yes	Dict(min,max) Scale of the evaluation
source	STRING	Yes	Source of the subjective metric
timestamp	DATETIME	Yes	Record creation time

If the error is of class type ErrorRequest or ExecutionError, a request ID (UUID) or a execution ID (BIGINT) are added, respectively, to this data object.

Table 43: Data Object: e2e_tc_error_do and domain_tc_error_do

Field	Type	Must	Description
module	ModuleEnum	Yes	Module where the error was triggered
type	TypeErrorEnum	Yes	Type of the error
stacktrace	STRING	Yes	Stacktrace of the error
create_date	DATETIME	No	Record creation time
comments	STRING	No	Any necessary comment
status	StatusErrorEnum	No	Status of the error

A.5 Database Model Information

A.5.1 Domain Level

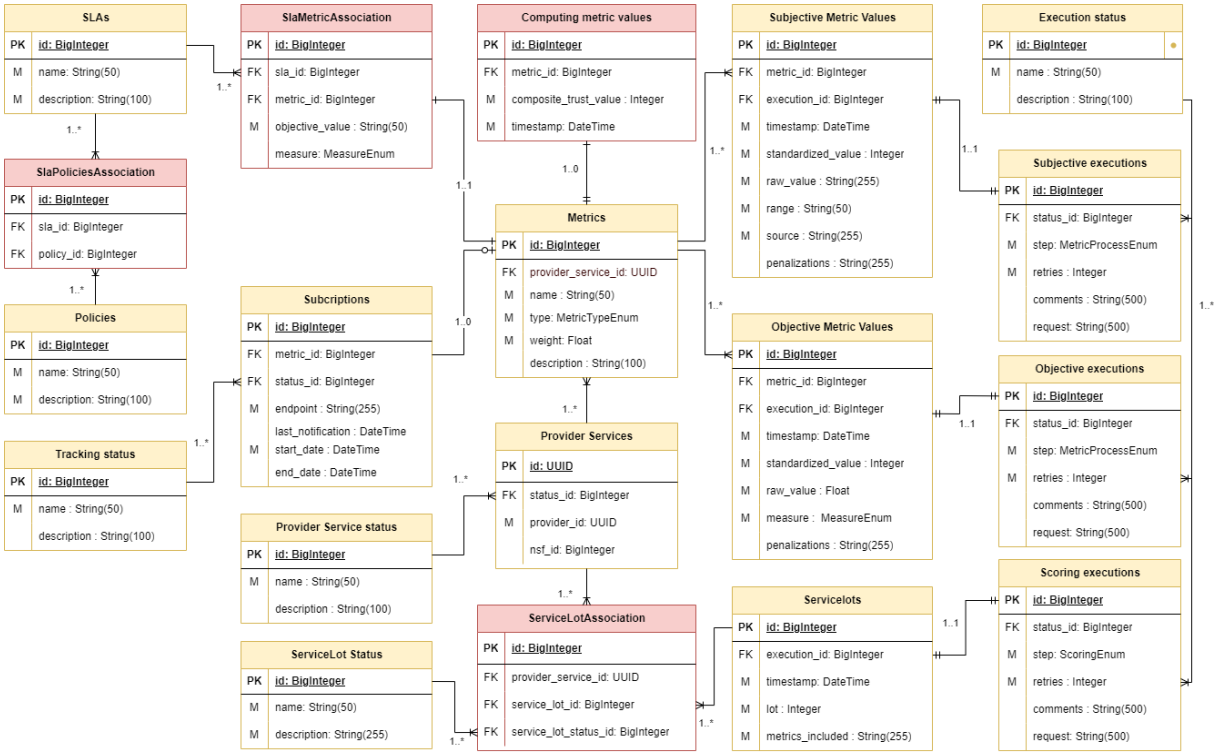


Figure 38: Domain Database Entity-Relationship Diagram

Appendix B
Details of results

B.1 Test Plan

B.1.1 Unit Tests

Context	Module	Test case	Test Name	Description	Preconditions	Expected results	Status	Execution Date	Assign	Evidences
Domain	Orchestrator	UT047	Raised an error while processing tracking request	Orchestrator raised an error while processing a tracking requests	Mock TC.TRACKING queue.	No error should occur. TC.ERROR queue must have one request	OK	5/3/2024	dmorei...	UT047.png
Domain	Orchestrator	UT048	Raised an error while processing execution request (batch service)	Orchestrator raised an error while processing a execution requests	Mock TC.TS queue.	No error should occur. TC.ERROR queue must have one request	OK	5/3/2024	dmorei...	UT048.png
Domain	Trust Metric Processor	UT049	Normal process objective metric	Process metric data that comes from TOM queue, calculate new composite value, updated it and notify TC.TS queue	Mock TOM queue. Mock TC.TS queue	No error should occur. TC.TS queue must have one request	OK	6/3/2024	dmorei...	UT049.png
Domain	Trust Metric Processor	UT050	Retry process objective metric (with execution id)	Process metric data that comes from TOM queue (execution id only), get data from execution.request, calculate new composite value, updated it and notify TC.TS queue	Mock TOM queue. Mock TC.TS queue	No error should occur. TC.TS queue must have one request	OK	6/3/2024	dmorei...	UT050.png
Domain	Trust Metric Processor	UT051	Normal process subjective metric	Process metric data that comes from subjective API, calculate new composite value, updated it and notify TC.TS queue	Mock subjective API. Mock TC.TS queue	No error should occur. TC.TS queue must have one request	OK	6/3/2024	dmorei...	UT051.png
Domain	Trust Metric Processor	UT052	Retry process subjective metric (with execution id)	Process metric data that comes from TSM queue (execution id only), get data from execution.request, calculate new composite value, updated it and notify TC.TS queue	Mock TSM queue. Mock TC.TS queue	No error should occur. TC.TS queue must have one request	OK	6/3/2024	dmorei...	UT052.png
Domain	Trust Metric Processor	UT053	Exhaust all retries in one case (UT049 for example)	Process metric data that comes from TOM queue, raise Timeout, retry, raise Timeout, store failed execution and escalate error to TC.ERROR queue	Mock TOM queue. Mock TC.ERROR queue	TC.ERROR queue must have one request	OK	6/3/2024	dmorei...	UT053.png
Domain	Trust Service Score	UT054	Process Service Lot	Process data that comes from TSS queue and calculate Service Lot	Mock TSS queue	No error should occur	OK	6/3/2024	dmorei...	UT054.png
Domain	Trust Service Score	UT055	Process Service Lot with execution id (retry case)	Process data that comes from TSS (with execution id) queue and calculate Service Lot	Mock TSS queue	No error should occur	OK	6/3/2024	dmorei...	UT055.png
Domain	Trust Service Score	UT056	Exhaust all retries in one case (UT053 for example)	Process data that comes from TSS queue, raise Timeout, retry, raise Timeout, store failed execution and escalate error to TC.ERROR queue	Mock TSS queue. Mock TC.ERROR queue	TC.ERROR queue must have one request	OK	6/3/2024	dmorei...	UT056.png
Domain	Error Handler	UT057	Execution error	Error can be retried, send to Orchestrator	Mock TC.RETRIES queue	TC.RETRIES queue must have one request. No issue must be created	OK	4/3/2024	dmorei...	
Domain	Error Handler	UT058	Execution error max retries	Error cant be retried, create issue	---	One issue must be created	OK	4/3/2024	dmorei...	
Domain	Error Handler	UT059	Error request	Activation/Deactivation failed response, send to PSM	Mock PSM queue	PSM queue must have one request. One issue must be created	OK	4/3/2024	dmorei...	UT057-0...
Domain	Error Handler	UT060	Error	Process simple error, create issue	---	One issue must be created	OK	4/3/2024	dmorei...	
Domain	Error Handler	UT061	Local error	Raise exception, process local error, create issue	Raise exception	One issue must be created	OK	4/3/2024	dmorei...	

Figure 39: Unit Test Plan Preview

B.1.2 Integration Tests

Context	Source module	Target module	Test case	Test Name	Description	Preconditions	Expected results	Status	Execution Date	Assign	Evidences
E2E	Service LoT Manager	Orchestrator	IT010	Receiving and processing an unknown Service LoT	TC.SLOT receives an unknown service lot message and creates it in the database and requests a new calculation execution from the Orchestrator.	Simulate that the E2E PSM Integration sends to Service Lot Manager (TC.SLOT queue) a request with new service lot.	Service lot is created in the database. TC.TPS queue has a message with a e2e_tc_tps_do.	OK	22/03/2024	dmorei...	IT010
E2E	Service LoT Manager	Orchestrator	IT011	Receiving and processing a known Service LoT.	Service Lot Manager receives a known service lot message and updates it in the database and requests a new calculation execution from the Orchestrator.	Simulate that the E2E PSM Integration sends to Service Lot Manager (TC.SLOT queue) a request with an existing service lot.	Service lot is updated in the database. TC.TPS queue has a message with an e2e_tc_tps_do.	OK	22/03/2024	dmorei...	IT011
E2E	Trust Provider Score		IT012	Generating unknown provider LoT	Trust Provider Score receives an unknown execution lot message. It generates an execution object and a provider lot applying a specific formula.	Simulate that the Orchestrator sends to Trust Provider Score (TPS queue) a request with a new provider execution lot.	Provider lot is created in database. Execution is created in database.	OK	24/03/2024	dmorei...	IT012
E2E	Trust Provider Score		IT013	Generating known provider LoT	Trust Provider Score receives a known execution lot message. It generates an execution object and a provider lot applying a specific formula.	Simulate that the Orchestrator sends to Trust Provider Score (TPS queue) a request with a provider execution lot.	Provider lot is updated in the database. Execution is created in database.	OK	24/03/2024	dmorei...	IT013
E2E	Trust Provider Score	Error Handler	IT014	Provider LoT generation failed	Trust Provider Score receives an execution lot message. It generates an execution object, but it generates an unknow error.	Simulate that the Orchestrator sends to Trust Provider Score a request with new a provider execution lot.	Provider lot isn't created in the database. Execution is created with original payload in the "request" field. TC.ERROR queue has a message with an e2e_tc_error_do.	OK	24/03/2024	dmorei...	IT014
E2E	Trust Provider Score		IT015	Continue with Provider Lot retry	Trust Provider Score receives a known execution lot message. It gets the execution object and generates a provider lot applying a specific formula.	Simulate that the Orchestrator sends to Trust Provider Score a request with a known provider execution lot.	Provider lot is updated in the database. Execution is updated in database.	OK	24/03/2024	dmorei...	IT015
E2E	Error Handler		IT016	Create an issue	Error handler receives an Error object, it creates an issue object.	Simulate that any subcomponent sends an Error object to Error Handler.	Issue is created in database.	OK	25/03/2024	dmorei...	IT016
E2E	Error Handler	Orchestrator	IT017	Handling a retriable error	The error handler receives a retriable ExecutionError object and sends it to Orchestrator.	Simulate that any subcomponent sends an ExecutionError object to Error Handler.	Issue isn't created in database. TC.RETRIES queue has a message with an e2e_tc_retries_do.	OK	25/03/2024	dmorei...	IT017
E2E	Error Handler		IT018	Handling a non-retriable error	The error handler receives a non-retriable ExecutionError object because it raised max retries.	Simulate that any subcomponent sends an ExecutionError object to Error Handler.	Issue is created in database. TC.RETRIES queue hasn't a message.	OK	25/03/2024	dmorei...	IT018
E2E	Orchestrator	Trust Provider Score	IT019	Start new execution lot calculation	Orchestrator receives a provider id and propagates the signal to Trust Provider Score.	Simulate that Service Lot Manager received a service lot and send to Orchestrator the provider id associated.	TPS queue has a message with an e2e_tps_do.	OK	25/03/2024	dmorei...	IT019
E2E	Orchestrator	Trust Provider Score	IT020	Initiate a retry of the execution lot calculation.	Orchestrator receives a retry of execution lot and propagate the signal to Trust Provider Score.	Simulate a scenario where an execution lot fails, the Error Handler forwards the failure to the Orchestrator.	TPS queue has a message with an e2e_tps_do (retry case).	OK	25/03/2024	dmorei...	IT020
E2E	Orchestrator		IT021	Discard retry	Orchestrator receives a retry of execution lot, but timestamp is older than the last Provider Lot calculated.	Simulate a scenario where an execution lot fails, the Error Handler forwards the failure to the Orchestrator.	TPS queue doesn't have a message. Execution status must be updated to CANCELLED.	OK	25/03/2024	dmorei...	IT021

Figure 40: Integration Test Plan Preview

B.1.3 Workflow Tests

Context	Test Case	Test Name	Description	Preconditions	Expected results	Status	Execution Date	Assign	Evidences
E2E	E2E001	Enable Provider Trust Calculation	E2E Trust controller receives a trust SLA activation request for enabling provider trust calculation.	Mock PDL-Service Manager and NSSO. Deploy Database and PSM Integration microservices.	Return a 200 ok and e2e_providers_sc_activated_do. The provider is created in database.	OK	25/03/2024	dmorei...	E2E001
E2E	E2E002	Disable Provider Trust Calculation	E2E Trust controller receives a trust SLA deactivation request for disabling provider trust calculation.	Mock PDL-Service Manager and NSSO. Deploy Database and PSM Integration microservices.	Return 200 OK and e2e_providers_sc_activated_do. The provider is updated with INACTIVE status.	OK	25/03/2024	dmorei...	E2E002
E2E	E2E003	Generate Provider LoT	The E2E PSM sends a Service Lot to E2E PSM subcomponent with the goal of generating a Provider LoT.	Mock E2E PSM endpoint. Deploy Database, E2E PSM Integration, Service Lot Manager (SLOT) and Trust Provider Score (TPS) microservices.	Service Lot is created/updated. Provider Lot is created/updated with PENDING status. E2E PSM receives a 200 OK.	OK	26/03/2024	dmorei...	E2E003
E2E	E2E004	Generate Provider LoT Error and Retry	The E2E PSM sends a Service Lot to E2E PSM subcomponent with the goal of generating a Provider LoT but it fails in computation phase.	Mock E2E PSM endpoint. Deploy Database, E2E PSM Integration, Service Lot Manager (SLOT) and Trust Provider Score (TPS), Error Handler and Orchestrator microservices.	Service Lot is created/updated. Provider Lot is not created/updated with PENDING status. E2E PSM received a ko response.	OK	27/03/2024	dmorei...	E2E004
E2E	E2E005	Register Provider LoT	The Provider Lots sending job activates, finds three objects in PENDING, and sends them to the E2E PSM, which return ok, synchronizing states.	Mock E2E PSM endpoint. Deploy Database and E2E PSM Integration microservices.	Return 200 OK from E2E PMS. The provider lots with ACTIVE status must be updated to INACTIVE status. The provider lots with PENDING status must be updated to ACTIVE status.	OK	27/03/2024	dmoreira	E2E005
E2E	E2E006	Register Provider LoT Error	The Provider Lots sending job activates, finds one object in PENDING, and sends it to the E2E PSM, which return timeout. When job activate again (last chance) E2E PSM return timeout again, creating an issue.	Mock E2E PSM endpoint. Deploy Database, E2E PSM Integration and Error Handler microservices.	Issue is created. Provider Lots status remains unchanged.	OK	27/03/2024	dmoreira	E2E006

Figure 41: Workflow Test Plan Preview

Place here information about funding, FCT project, etc. in which the work is framed. Leave empty otherwise.