6GSNS

Co-funded by the European Union

SAFE-6G

SAFE-6G

A Smart and Adaptive Framework for Enhancing Trust in 6G Networks

Deliverable D4.1: Cognitive coordinator, AI agents and User-centric functions

Date: 22/12/2025

Version: v1.1

DISCLAIMER

Grant Agreement	101139031
Document number	D4.1
Document title	Cognitive coordinator, AI agents and User-centric functions
Lead Beneficiary	IQBT
Editor(s)	Theodoros Ioannidis (IQBT) Kushal Mehta (IQBT)
Author(s)	Theodoros Ioannidis (IQBT) Kushal Mehta (IQBT) Iraklis Spyrou (INF) Evangelos Anagnostopoulos (INF) Georgios Koumaras (INF) Eugenia Vergi (INF) Guillaume Hébert (KEY) Henry Faures Geors (KEY) Van Hoan Hoang (KEY) Ilias Alexandropoulos (NCSRD) Georgios C Batsis (NCSRD) Spyridon Georgoulas (NCSRD) Harilaos Koumaras (NCSRD) Dimitris Uzunidis (UNIWA) Panos Karkazis (UNIWA) Stamatia Drampalou (UNIWA) Christos Betzelos (UNIWA) Gaëtan Pruvost (THALES) Andrés Anaya Amariles (TID) Rodrigo Sanz Sanz (TID) Apostolos Garos (SHG) Victoria Katsarou (SHG) Alejandro Fornés (UPV) Sonia Castro (EVIDEN) Joaquín Cáceres (EVIDEN) Nikolaos Zompakis (8BELLS) Zouzias Dimitris (eBOS)
Dissemination level	Public
Contractual date of delivery	30/06/2025
Status	Final
File name	D4.1_ SAFE-6G_v1.1.docx

Revision History

Version	
V0.1	Initial definition of TOC.
V0.2	Initial review of TOC.
V0.3	First round of Contributions.
V0.4	Second round of Contributions.
V0.5	Third round of contributions and internal review performed by IQBT.
V0.6	First review performed by THA and UNIWA.
V0.7	Version produced by IQBT based on the comments from the First Review.
V0.8	Second review performed by the Technical Steering Committee.
V0.9	Final draft and homogenization of content. Final review performed by the Project Coordinator and the Editor.
V1.0	Final version following the Quality check
V1.1	Updated version after mid-term review

GLOSSARY

Abbreviations/Acronym	Description
6G	Sixth generation
AI	Artificial Intelligence
ALE	Accumulated Local Effects
AMF	Access and Mobility Function
APIs	Application Programming Interfaces
BERT	Bidirectional Encoder Representations from Transformers
CA	Consortium Agreement
CC-BY	Creative Commons Attribution License
CEM	Contrastive Explanation Method
CFREU	Charter of Fundamental Rights of the European Union
cLoTw	Calibrated Level of Trustworthiness
CoCo	Cognitive Coordinator
CPU	Central Processing Unit
CV	Curriculum Vitae
DG RTD	Directorate-General for Research and Innovation
DIDs	Decentralized identifiers
DLoR	Desirable Levels of Resilience
DMP	Data Management Plan
DoA	Description of Action
DOI	Digital Object Identifiers
DPIA	Data Protection Impact Assessment
DPO	Data Protection Officer
EAB	Ethics Advisory Board
EC	European Commission
ECHR	European Convention for the Protection of Human Rights and Fundamental Freedoms
ENISA	EU Agency for Cybersecurity
ePD	ePrivacy Directive
ePR	ePrivacy Regulation
EU	European Union
FAIR	Findability, Accessibility, Interoperability, and Reusability
FL	Federated learning
GA	Grant Agreement
GDPR	General Data Protection Regulation
IPR	Intellectual Property Rights
KPIs	Key Performance Indicators
KVIs	Key Value Indicators
LoTw	Level of Trustworthiness
MANO	Management and Orchestration
ML	Machine Learning
MLOps	Machine Learning Operations

nApp	Network Application
NFs	Network functions
NIS	Network and Information Systems
nLoTw	Non-calibrated Level of Trustworthiness
NLP	Natural Language Processing
NSN	Network Service Nodes
NWDAF	Network Data Analytics Function
OIDC	Open ID Connect
ORDP	Open Research Data Pilot
PCF	Policy Control Function
PDP	Partial Dependence
PDV	Partial Dependence Variance
PII	Personally Identifiable Information
QoE	Quality of Experience
QoS	Quality of Service
RF	Resilience Function
SDP	Software Defined Perimeter
SDPCF	Software Defined Perimeter Core Function
SDPGW	Software Defined Perimeter GateWay
SMEs	Small and medium-sized enterprises
SMF	Session Management Function
SSI	Self-Sovereign Identity
TF	Trust Function
TF	Trust Function
TFEU	Treaty on the Functioning of the European Union
UE	User Equipment
UPF	User Plane Function
USN	User Service Nodes
vApp	Vertical Application
VCs	Verifiable Credentials
VGF	VPN Gateway Function
VPN	Virtual Private Network
WP	Work Package
WPLs	WP Leaders
XAI	eXplainable AI

EXECUTIVE SUMMARY

This deliverable presents the architectural design and preliminary implementation of the SAFE-6G cognitive trust management framework. It focuses on the Cognitive Coordinator (CoCo), eXplainable AI (XAI) module, and the five user-centric Trust Functions (TFs): Safety, Security, Privacy, Resilience, and Reliability.

The CoCo is implemented with a modular architecture, including a BERT-based regression model to interpret user intent and a resource-aware reasoning engine to calibrate trust levels in real-time. This module orchestrates the actions of the TFs based on semantic inputs and system constraints. The XAI module, developed in parallel, was designed to provide both local (contextual) and global (model-level) interpretability for decisions made by the Trust Functions.

Major technical progress has been achieved across all TFs, with each now featuring a defined architecture, multi-layer interaction model (AI Agent, vApps, nApps), and functional logic for enforcement, monitoring, and adaptation.

The Safety Function has delivered a modified Software Defined Perimeter (SDP) mechanism for secure resource isolation across the continuum.

The Security Function integrates local AI classification and forecasting with verifiable credentials and a Hyperledger Fabric blockchain for access control and auditability.

The Privacy Function enables fine-grained, session-level control using containerized nApps that modify 5G core behaviour according to dynamically computed privacy levels.

The Resilience Function introduces AI-driven fault tolerance using rule-based orchestration and telemetry feedback across OpenCAPIF and aerOS interfaces.

The Reliability Function maps Level of Trustworthiness (LoTw) tiers to tailored AI/ML pipelines and leverages multi-layer telemetry collection for Quality of Experience (QoE)-aligned prediction and mitigation.

This deliverable marks a key milestone by establishing the foundation for SAFE-6G's trust-aware orchestration layer. It prepares the ground for full-scale integration and validation activities, including the deployment of the CoCo, Trust Functions, and AI agents in immersive metaverse scenarios.

These components will be integrated into advanced 6G experimentation platforms and tested under diverse threat models and trust configurations. These activities are coordinated as part of the project's broader integration and pilot validation efforts, where system-level testing and use case execution will be carried out in the next phase and documented in the corresponding project deliverables.

KEYWORDS

Cognitive Coordinator, XAI, User-centric Trust Functions, Development status

TABLE OF CONTENTS

List of FIGURES

List of TABLES

1 INTRODUCTION

The SAFE-6G project introduces a paradigm shift in the establishment, evaluation, and enforcement of trust in 6G networks. The significance of a dynamic and adaptive trust framework becomes more evident as communication systems become more decentralized, intent-driven, and user-centric. The Cognitive Coordinator (CoCo), Local AI Agents, and the five User-Centric Trust Functions (TFs) (Safety, Security, Privacy, Resilience, and Reliability) are the primary focus of Deliverable D4.1, which is the foundational technical output of WP4.

This document presents how trust-related decisions can be automated and orchestrated through AI-driven reasoning, contextual inference, and explainability mechanisms. It outlines the architecture and operational roles of the CoCo and its interaction with AI Agents and external systems, the structural and semantic design of each TF, and the early integration status of the Explainable AI (XAI) module.

Additionally, the deliverable includes a development roadmap, partner responsibilities, and alignment with the SAFE-6G objectives and KPIs toward the final implementation milestones. The document targets and aligns with technical partners involved in system architecture (WP3–WP5), WP leaders responsible for integration, external reviewers monitoring progress against the Grant Agreement (GA), and researchers exploring intent-based trust systems.

The deliverable is structured as follows: Section 2 introduces trust requirements and the Trust Function concept; Section 3 explains the Cognitive Coordinator architecture; Section 4 focuses on the XAI module; Sections 5–9 describe each Trust Function in depth; and Section 10 outlines the implementation roadmap, maturity status, and objective alignment for WP4.

2 Overview of the Trustworthiness Management Framework

2.1 Trustworthiness in 6G: Challenges and Needs

As communication networks progress towards sixth-generation (6G) architectures, trust has transitioned from a peripheral issue to a fundamental design objective. Conventional methods—primarily reliant on fixed identities, perimeter-centric security, and implicit trust presuppositions—are insufficient for decentralized, intent-driven, and dynamic 6G environments.

6G will be characterized by the amalgamation of diverse resources, ultra-dense edge infrastructures, and user-centric orchestration frameworks. In such systems, trust cannot be predetermined or assessed solely during onboarding. It must be perpetually evaluated, contextually implemented, and user adaptive. The principal challenges that trust management must confront in 6G encompass:

- Trust gaps in heterogeneous infrastructure: 6G networks will integrate core and RAN components, along with edge nodes, devices, and dynamically instantiated virtualized services. Establishing and sustaining trust within such a diverse ecosystem is complex [1] [2] .
- Translating high-level user intents into actionable trust and privacy policies necessitates sophisticated parsing, semantic comprehension, and verifiable decision-making logic [3] [4] .
- Distributed enforcement at the edge: The transition to decentralized architectures necessitates the integration of trust enforcement mechanisms directly within edge devices, fog nodes, and lightweight service orchestrators—settings where computational resources and observability are frequently limited [5] [6] .
- The implementation of AI and Machine Learning (ML) in trust scoring and decision-making presents risks associated with black-box phenomena, necessitating explainability and transparency. Operators and users must comprehend and contest trust decisions, particularly when these decisions affect access, isolation, or prioritization [7] .
- Cross-domain trust governance: 6G environments frequently encompass various administrative domains and legal frameworks. Trust policies must be interoperable, auditable, and compliant with regulations across jurisdictions, especially regarding identity, consent, and data retention [9] .

To tackle these challenges, frameworks like SIX-Trust [2] and 6Blocks [6] advocate for multi-layer trust architectures that distinguish between sustainability, infrastructure, and service-level trust. Others implement zero-trust principles, enforcing ongoing verification and adaptive access control throughout the trust lifecycle. Blockchain-based trust anchors and federated reputation models offer supplementary components for verifiability and traceability in open 6G ecosystems.

The SAFE-6G project advances these cutting-edge approaches by integrating a cognitive trust orchestration framework that combines semantic user intent, dynamic TFs, and XAI into a cohesive, modular architecture. Trust is regarded not as a fixed characteristic, but as a dynamic, contextual process, implemented through the ongoing coordination of AI agents, vApps/nApps, and the Cognitive Coordinator (CoCo).

2.2 AI Cognitive Coordination in 6G Environments

The SAFE-6G CoCo is the intelligence core of the trustworthiness orchestration framework, enabling user-centric, intent-based coordination across heterogeneous, edge-cloud 6G environments. It is responsible for the coordination of AI enabled Trust Functions and the ongoing adaptation of trust provisioning strategies based on contextual data and feedback to convert high-level user objectives into actionable strategies.

It integrates a semantic reasoning engine with AI/ML-driven orchestration logic at its core. A structured NLP (Natural Language Processing) pipeline is employed to parse, classify, and map user intents—which are captured through natural language interfaces like the SAFE-6G chatbot—to one or more TFs. These inputs are employed to calculate a non-calibrated level of trustworthiness (nLoTw) for each trust dimension. Subsequently, the nLoTw is refined into a calibrated level of trustworthiness (cLoTw) by utilising established knowledge about the users as well as the possible conflicts that may occur during deployment on TFs side.

The CoCo's orchestration capabilities are data-driven and modular. It interacts explicitly or implicitly with all the SAFE-6G framework's components to maintain operational awareness. This closed-loop coordination enables CoCo to operate autonomously and adaptively, thereby ensuring that the system's behaviour is consistently in alignment with the user's trust objectives. The Cognitive Coordinator comprises three core components. First, the Regression Component leverages a BERT-based model to transform user intents into non-calibrated trust scores across five dimensions. Second, the Reasoning Engine contextualizes and calibrates these scores (cLoTw) based on resource availability, semantic rules, and user constraints. Third, the Cognitive Orchestration Module manages the deployment and coordination of TFs, using a message broker for scalable, asynchronous communication. Together, these components enable adaptive, intent-aware trust governance across the SAFE-6G ecosystem. Lastly, CoCo maintains connections with all key parts of the SAFE-6G ecosystem:

- The **Monitoring Framework**, to ingest telemetry data.
- The **MLOps system**, to retrieve model status and initiate retraining.
- The **CAPIF interface**, to manage service access and data flow.
- The **Trust Functions**, to deploy and monitor user-specific enforcement logic.
- The **Metaverse applications**, to contextualize trust enforcement in immersive, dynamic environments.
- The **Chatbot:** to enable user interaction and intent collection through natural language interfaces.
- The **5G/6G network:** to exchange control and user-plane data across domains, enabling real-time enforcement of trust policies.
- The **Edge-Cloud Continuum**: to support scalable, low-latency execution of services across distributed computing resources.

Overall, the Cognitive Coordinator serves as the intelligence core of SAFE-6G, enabling real-time, user-driven trust orchestration. By integrating semantic reasoning, AI-driven inference, and dynamic coordination, it ensures system behaviour aligns with evolving trust objectives. It lays the foundation for scalable, explainable, and adaptive trust management in next generation 6G networks.

2.3 Role of XAI in 6G Trust Coordination

Modern ML models achieve remarkable accuracy in many tabular data contexts, but their complexity often makes them opaque "black boxes." In high-stakes domains – from finance and healthcare to next-generation networks – this lack of transparency hinders trust and adoption. XAI addresses this challenge by providing human-understandable explanations for model decisions. XAI methods are increasingly recognized as crucial for ensuring transparency and regulatory compliance in AI-driven systems.

In the case of the SAFE-6G platform, the Trust Functions are managed by AI controllers, which decide about resource allocation and security. Yet operators may struggle to trust decisions they cannot understand. XAI techniques promise to bridge this gap by revealing why a model made a decision and which attributes it considered, enabling humans to interpret, validate, and even contest automated decisions. The XAI module will propose an interpretability layer between the decisions made by individual trust functions and operators. It will allow answering two categories of requests:

1. **Local explainability**: the XAI module helps in interpreting a given decision made by a trust function within a very specific context. For instance, the XAI module could explain why a given user was attributed a given Quality of Service (QoS) group by the Resilience TF. By analysing the context – i.e. the Resilience TF inputs – it may explain what parts of the context were crucial in making this decision or give counter-examples of how it could have been assigned a different QoS if the context was different.

2. **Global explainability**: the XAI module provides general rules of thumb on the influence of inputs on the predictions made by the TF. For instance, it may explain that the "*RequestType*" and the "*SystemPolicyMatch*" are, in general, crucial for Security TF to predict if the platform may or may not fulfil a query. Another use case of global explainability would be to ask the marginal impact of a given input feature with respect to a prediction.

Section 4.3 details the status of the development of this module.

2.4 Overview of the Five User-Centric Trust Functions

While each of the five Trust Functions (Safety, Security, Privacy, Resilience, Reliability) is designed to address a unique trust dimension, they are all based on a shared architectural model. Ensured by this unified structure are modularity, interoperability, and ease of orchestration under the Cognitive Coordinator. A multi-layered logic stack consisting of three core elements is used to implement each TF:

- **Local AI Agent**: A pluggable, lightweight AI component that is responsible for real-time inference, prediction, or classification in accordance with the logic of the TF. These agents can be dynamically updated or replaced and operate using models that are customized to their respective trust domains (e.g., assessing data exposure risk for Privacy, forecasting anomalies for Security). Models are provided by the SAFE-6G MLOps platform, which is seamlessly integrated with the AI Agent to deploy and serve the latest validated models in a containerized inference environment for each TF domain. The particularities of each integration are described in the Local AI Agent chapter of Sections from 5 to 9.

- **Vertical Applications (vApps):** are internal components of a Trust Function that operate at the Service/Application plane. They implement the specific trust dimension and translate trustworthiness requirements into service-level actions, such as access control, workload isolation, service configuration, or policy enforcement. vApps act on vertical applications by retrieving necessary information and/or applying trust-related actions within the application and service layer.
- **Network Applications (nApps):** are internal, network-level control components of a Trust Function, operating close to the 5G core and the edge–cloud infrastructure. They retrieve available information from the network and infrastructure planes and perform trust-driven actions directly on these planes, such as modifying data paths, configuring sessions, adjusting routing, or enforcing connectivity and transport-level policies. Their role is to either consume information from or apply low-level changes to the network and infrastructure required to realize trustworthiness decisions.

This layered model guarantees that trust enforcement can occur at the appropriate abstraction level, ranging from strategic intent interpretation (AI Agent) to execution (vApp) to infrastructure reconfiguration (nApp). It also enables the Cognitive Coordinator to coordinate the deployment, monitoring, and explainability patterns of all TFs.

SAFE-6G guarantees that each TF is self-contained and interoperable, capable of being integrated into more intricate trust workflows or functioning independently based on the context, by homogenizing their design. This architectural alignment streamlines lifecycle management, improves scalability, and minimizes integration overhead in a variety of deployment environments.

2.4.1 User-centric Safety Function

The Safety Function of the SAFE-6G framework correlates the requested level of trust to the corresponding resource isolation needed at the level of virtualised resources. The process of providing the requested level of trust within this concept starts by reassuring that the required cloud continuum resources that will be utilised are safely protected from unauthorised access and resource sharing with third parties.

To achieve this, the Software Defined Perimeter (SDP) stack technology is modified and integrated in the 6G infrastructure as part of the SAFE-6G Safety function. The developed SDP provides the protection of infrastructure nodes and ensures the availability of relevant microservices to specific users. In a typical 5G network, when a user tries to connect to the User Plane Function (UPF), mutual authentication will take place between the User Equipment (UE) and the AMF, with the location of the AMF exposed to the UE, leaving it vulnerable to attacks. SAFE-6G approach will make use of the SDP controller to exchange a mutual authentication package with the UE through a VPN connection, and as such, protect its location, making it susceptible to attacks. Upon successful authentication of the UE with the controller and the authentication host, a perimeter will be established for the user, which will contain all the necessary components. Additionally, microservices on the User Plane will be independently authenticated before being added to individual perimeters and deployed to the network.

The required Level of Trust is defined by a local AI Agent based on the scoring received by the Cognitive Coordinator. The classification output maps to one of the Safety Function flavours, each representing a different level of Safety. The expert scoring of Safety-related keywords was derived through a

structured, evidence-based process designed to translate qualitative safety concepts into quantitative trustworthiness levels. Initially, the scope of the Safety function was defined within the broader trustworthiness framework, agreeing that it encompasses all mechanisms and procedures ensuring that it provides the protection of infrastructure nodes and the availability of relevant microservices. Using this definition, a comprehensive list of Safety-relevant keywords was compiled, capturing both preventive and corrective aspects of safety (e.g., zero-trust authorization, encrypted connection, resource isolation).

Each keyword was then assessed, and each keyword was given a score on a scale from 0–100 according to the expected contribution of the corresponding mechanism to overall system safety keeping in mind how much the feature limits the probability of potential harm, the range of failure scenarios that can be addressed, and the degree of confidence that the safety mechanism will perform as intended. The individual assessments were averaged and normalized, leading to the final score clusters, aligned with increasing levels of trustworthiness. The first level provides fundamental protection mechanisms. It includes basic authorization, device validation, and entry-level resource isolation, focusing on establishing minimal but essential safety barriers. Systems at this level demonstrate baseline trust and basic segmentation to prevent accidental or low-skill threats. The second level introduces dynamic and role-based safety mechanisms. It expands on Level 1 by implementing standardized safety links and dynamic resource isolation. Finally, level 3 is the highest assurance level, integrating zero-trust principles, adaptive isolation, and comprehensive device validation.

Overall, the main characteristic of this approach is that the resource and tenant interactions are now decoupled by the SAFE-6G Safety function, with key elements being the Resource separation in terms of Visibility, Object footprint and Quotas, Configuration separation, and finally administrative separation.

2.4.2 User-centric Security Function

The user-centric security function handles real-time enforcement of security decisions based on trust levels, system metrics, and context provided by the Cognitive Coordinator. While it follows the same layered structure as the other trust functions; using a Local AI Agent, vApps and nApps, it focuses specifically on identity access management, security evaluation, policy enforcement and auditability.

At the core of the function is the local AI agent that processes incoming trust-score messages and determines whether a security response is required. A stacked ML model informs this decision by evaluating system state features like CPU usage, memory load, and trust score metrics. The classification output maps to one of three deployment levels or “flavors”, each representing a different level of enforcement. These can trigger anything from passive monitoring to partial restrictions or full access revocation with credential validation, depending on the assessed trust level.

In parallel, a forecasting component built on the fbProphet library [13] analyses historical trends to anticipate future resources demands. This allows the system to proactively adjust its posture before thresholds are exceeded, ensuring that enforcement mechanisms scale with predicted risk.

For access control, the function combines OAuth2 [8] (via an OIDC provider) with verifiable credentials (via a Self-Sovereign Identity (SSI) agent). This hybrid model supports both short-term token-based access and decentralized identity flows, where credentials are verified without relying on a central

authority. As a result, access decisions reflect both the user's current trust context and their verifiable attributes.

All major actions, such as credential issuance or flavour deployment, are recorded on a Hyperledger Fabric [13] blockchain via custom chaincodes. This provides a tamper-evident audit trail without revealing sensitive data, thanks to selective disclosure and scoped token access. It also ensures long-term traceability of trust-related decisions, supporting both compliance and post-event validation.

The security function brings decentralized, context-aware, and verifiable enforcement to the SAFE-6G stack. Its focus is on adapting security to live trust conditions and system behaviour, rather than relying solely on static identities or fixed rules.

2.4.3 User-centric Privacy Function

The User-Centric Privacy Function is designed to apply dynamic, fine-grained privacy controls directly within the 5G core network by executing predefined actions that modify the behaviour of network slices, sessions, and QoS configurations. These actions are implemented through lightweight, containerized components known as nApps, each of which encapsulates a specific network operation—such as adjusting pre-emption settings, altering throughput limits, or modifying slice attributes. The execution of these actions is triggered based on contextual requirements and policy decisions, which are mapped to vApps that group together one or more nApps aligned with a targeted privacy level (Low, Medium, High).

Once a vApp is selected, following a decision by the AI agent and confirmation from the CoCo, it invokes the necessary nApps, each of which interacts directly with the 5G core using authenticated API calls. These calls apply real-time configuration changes to entities such as user sessions, slice profiles, and QoS parameters. Examples include reducing the number of allowed UEs per slice to increase isolation or capping session-level throughput to obscure behavioural patterns. Each nApp operates independently, ensuring modularity and enabling reuse across multiple vApps depending on the privacy objectives.

This privacy enforcement mechanism is both reactive and auditable. Before execution, the proposed action plan is sent to the CoCo for approval. Once authorized, the orchestrator initializes the appropriate vApp, and the invoked nApps apply the changes to the network infrastructure. All actions are parameterized, reversible, and executed with minimal service disruption. This architecture enables the system to respond quickly to privacy-sensitive contexts while maintaining operational stability, offering a practical path to embedded, user-aligned privacy enforcement in real-world 5G deployments.

2.4.4 User-centric Resilience Function

The User-centric Resilience Function is a foundational component of the SAFE-6G framework, designed to ensure adaptive, intelligent, and trustworthy network behaviour across the edge-cloud continuum. It enables the dynamic orchestration of network resources and services tailored to individual user needs and intents, thereby enhancing the overall resilience and reliability of 5G/6G mobile networks.

The Resilience Function works in close coordination with the CoCo, which acts as the brain of the SAFE-6G trust system. CoCo is responsible for managing trust levels across the network by issuing

trustworthiness requests and collecting feedback from all Trust Functions, including the Resilience Function. More details about the design and definition of the CoCo can be found in Section 3.

In SAFE-6G, the Level of Trustworthiness (LoTw) is a system property that represents how much trust a service or session can provide within a 6G network. Instead of focusing only on traditional security, LoTw takes a broader view of trustworthiness, which includes five important dimensions: safety, security, privacy, resilience, and reliability. Each of these dimensions is handled by a dedicated TF in the SAFE-6G framework, with the Resilience Function playing a key role in ensuring resilience and reliability for users.

The Resilience Function integrates an AI Resilience Agent, which autonomously analyses real-time telemetry, user behaviour, and system performance metrics to drive informed decision-making. These decisions are guided by the LoTw and multiple Desirable Levels of Resilience (DLoR), ensuring that service configurations are both context-aware and user-aligned.

The Resilience Function operates as a Network Application (App) and interacts with various SAFE-6G components through standardized APIs, including OpenCAPIF [9] , Cumucore [10] and aerOS [11] enabling fine-grained control over user-specific strategies.

The key capabilities of the Resilience Function include:

- **Real-time assessment** of resilience feasibility via the CoCo.
- **Dynamic adaptation** of deployment templates and configuration based on evolving user intents.
- **Continuous monitoring** of performance, energy consumption, and infrastructure health.
- **Rule-based and AI-driven** actions to maintain service continuity, optimize QoS, and ensure fault tolerance.

2.4.5 User-centric Reliability Function

The User-Centric Reliability TF represents a major shift from traditional network-centric approaches to a user-focused reliability framework within the SAFE-6G ecosystem. This function prioritizes Quality of Experience (QoE) by aligning network performance with individual user needs and requirements, moving beyond conventional packet loss rate measurements to examine subjective QoE-related metrics.

The reliability function architecture comprises three core components: the AI Agent, vApp, and nApp, all developed in Python and containerized for deployment within a microservice architecture across the heterogeneous edge-cloud continuum. The AI Agent serves as the decision-making core, receiving LoTw scores from the Cognitive Coordinator and mapping them to appropriate trustworthiness levels: Low (1-40%), Medium (41-80%), and High (81-100%). Based on these levels, the system dynamically selects and deploys appropriate AI/ML models tailored to specific user requirements.

The function operates through a structured workflow where the nApp interfaces with various data sources, including Prometheus and OpenCAPIF, collecting multi-layered monitoring data from infrastructure, network, and application planes. This comprehensive data collection enables the vApp to process input data and generate predictions, which inform the AI Agent's decisions for enhancing service reliability across the entire 6G edge-cloud continuum. The main capabilities of the Reliability TF include:

- **Dynamic LoTw-Based Model Selection**: the function maps LoTw scores into three tiers (Low, Medium, and High), with each tier deploying increasingly sophisticated AI/ML models from lightweight k-Nearest Neighbors algorithms to advanced federated learning (FL) with deep neural networks.
- **Multi-Layered Data Collection Framework**: the function operates in a continuous loop collecting real-time metrics across infrastructure (CPU, memory, network interfaces), network (5G/6G core functions, NWDAF, UPF), and application planes (QoS/QoE metrics, service latency), exploiting sources like OpenCAPIF, Prometheus, and RabbitMQ messaging.
- **User-Centric Approach with Cross-Layer Correlation**: the function focuses on subjective QoE-related metrics by analyzing how user characteristics (position, speed, device capabilities) directly impact radio access quality and infrastructure performance, enabling proactive identification of reliability related degradation patterns.
- **Proactive Security and Performance Management**: the function employs both service profiling (identifying infrastructure performance impacts) and security profiling (detecting anomalies) to trigger automated mitigation actions, including dynamic scaling, service migration, and intelligent system reconfiguration across the 6G edge-cloud continuum.

3 COGNITIVE COORDINATOR

The evolution toward 6G networks marks a critical shift in the telecommunications landscape, demanding a redefinition of trust in network systems. The SAFE-6G framework addresses this necessity through a user-centric, AI-native approach that prioritizes trustworthiness-encompassing safety, security, privacy, resilience, and reliability. At the heart of this framework lies the Cognitive Coordinator, an innovative, AI-powered orchestration component responsible for interpreting user trust intents and dynamically steering the system toward desirable, trustworthy operational states. The CoCo is designed to seamlessly translate high-level, semantic user expressions received through an AI-enabled chatbot-into actionable trust requirements. By coordinating five specialized TFs, the CoCo ensures that user-defined trust goals are systematically achieved in real-time, adaptive 6G environments. This orchestration leverages advanced AI/ML capabilities, XAI, and the intelligent integration of system and network metrics to deliver a highly responsive trust governance layer.

3.1 DATASET

To inform and calibrate the decisions made by the CoCo, a structured dataset categorizes various key terms according to five trust dimensions: Reliability, Privacy, Security, Safety, and Resilience. The Dataset is available at Zenodo: https://zenodo.org/records/15512671. Each keyword is mapped to a score (0–100), indicating its trust level, granularity, and relevance.

The data preparation process is as follows: Keywords related to each Trust Function are provided by domain experts. They are aligned with actual use cases and reflect the intents that vertical industry stakeholders, telco operators, system integrators or end users employ, in natural language, when describing specific telco needs. As far as the scores are concerned, a survey is conducted among the domain experts in which they are presented with the keywords-TFs pairs and their task is to assign a score from 0 to 100 that reflects their perceived relevance and expected trustworthiness level for the respective pair. The survey's results for each keyword are aggregated, yielding the final values per keyword.

The provided dataset sums up to 142 rows of "trust vectors" and their quantified associations (Intent - Trust Function Label – Trustworthiness Score), reflecting the most concrete and resourceful cases identified by the experts in a manner that it represents a high-quality, domain-curated, consensus-driven set rather than a large crowdsourced corpus. This is intentional as CoCo reasoning must be grounded in validated expert knowledge, not noise. The requirement for a much bigger dataset in order to train a deep learning classification model is tackled by augmenting the dataset with synonyms, utilising Princeton University's wordnet lexical database [14] and the nltk [15] library.

Such a dataset is able to guide the AI-enabled classification and reasoning engine within the CoCo to compute and refine both the initial (non-calibrated) and final (calibrated) Level of Trustworthiness (nLoTw and cLoTw, respectively).

In a deeper, explanatory dive per TF:

- **Reliability**: Keywords range from basic indicators such as "reliable application" and "fast algorithms" to advanced concepts like "federated learning (FL)" and "very complex algorithms." Higher scores reflect high complexity and performance-oriented capabilities.

- **Privacy**: The dataset includes basic data types (e.g., "user data," "cookie preferences") with lower scores and progresses to stringent privacy-preserving mechanisms like "end-to-end encryption," "zero-trust architecture," and "GDPR adherence."
- **Security**: Spanning from simple measures (e.g., "password protection") to robust solutions (e.g., "encryption keys," "intrusion prevention," and "penetration testing"), the security-related entries are graded to reflect escalating protection levels.
- **Safety**: Items are grouped from basic segmentation and authorization to advanced trust validation and encrypted communication, reflecting hierarchical safety provisions.
- **Resilience**: Starting from basic mechanisms like "load balancing" and "congestion monitoring," the resilience scores culminate in AI-powered strategies such as "predictive load distribution" and "automated service restoration."

3.2 Detailed Architecture of the Cognitive Coordinator

An overall architectural diagram of Cognitive Coordinator's internal, as well as neighboring, components is provided in the diagram below:

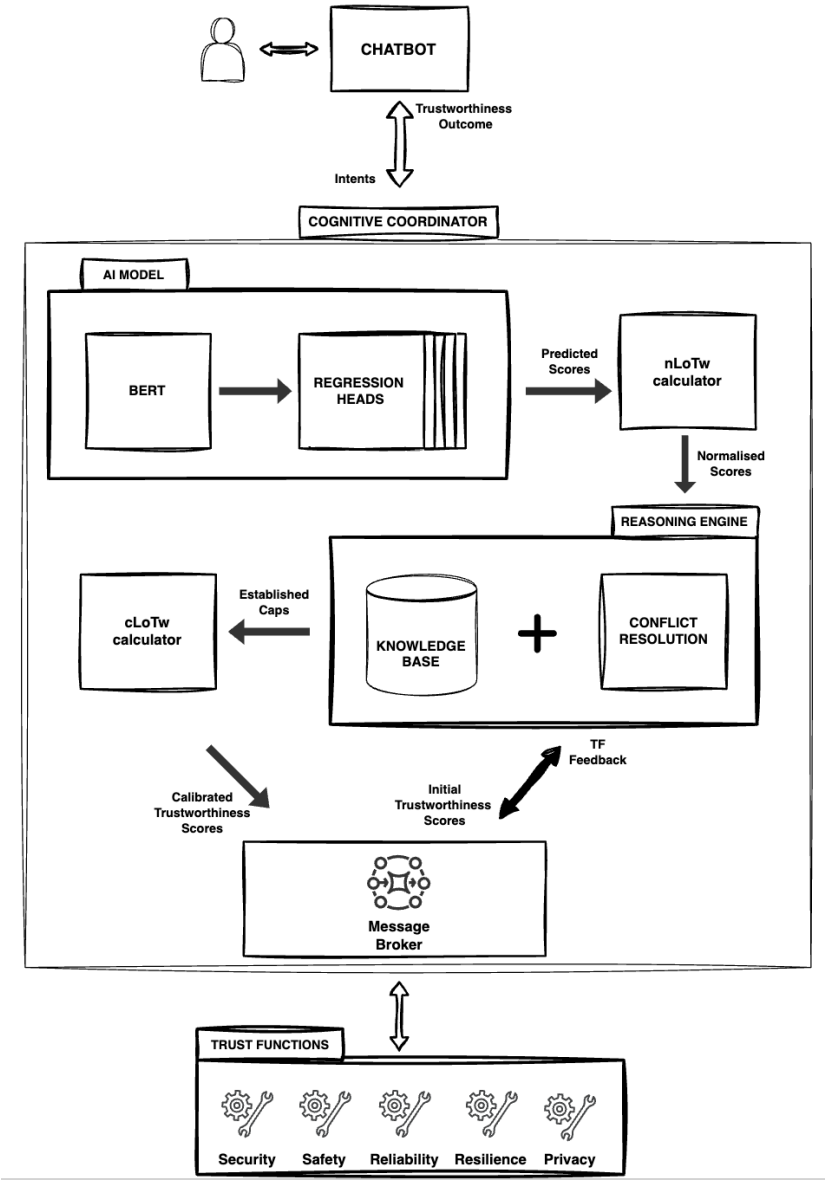

Figure 1: CoCo - Calculation of trust scores and communication with TFs.

In a deeper, explanatory dive, the Cognitive Coordinator itself comprises three core components:

1. **Regression Component:** The Regression Component serves as the initial processing stage within the SAFE-6G CoCo pipeline. It ingests semantically parsed user trust expressions provided by the AI Chatbot, which uses an NLP pipeline to convert raw natural language into structured representations aligned with the five trust dimensions: Safety, Security, Privacy, Resilience, and Reliability. Each user input sentence or keyword is associated with one or more TFs based on semantic similarity and classification logic. **¡Error! No se encuentra el origen de la referencia.**

To compute the preliminary nLoTw, this component employs pretrained **Bidirectional Encoder Representations from Transformers (BERT)** model, selected for its strong contextual embedding capabilities and generalizability across linguistic domains. On top of the shared BERT encoder, five

specialized regression heads-one per Trust Function-are fine-tuned for the SAFE-6G regression task. Each head outputs a continuous trustworthiness score $REF_{TF_j} \in [0, 100]$, corresponding to the TF-specific trust demand inferred from the input.

The number of sentences n_{TF_j} classified under each trust function TF_j, $for\ j \in \{1, 2, 3, 4, 5\}$ and the total number of sentences **N** is used to define a dynamic weighting mechanism that reflects the distribution of user concerns. Instead of using simple frequency ratios, the SAFE-6G model introduces a logarithmic-based weighting function that improves smoothness and mitigates sparsity effects:

$$W_{TF_j} = \frac{\log\left(1 + n_{TF_j}\right)}{\sum_{j=1}^{5} \log\left(1 + n_{TF_j}\right)}$$

This weight captures the proportional relevance of each TF in the user's intent profile while log transform is used to curb dominance by very large counts. The nLoTw is then computed as the normalized weighted average of the TF-specific regression outputs:

$$nLoTW = \frac{\sum_{j=1}^{5} W_{TF_j} REG_{TF_j}}{\sum_{j=1}^{5} W_{TF_j}} \cdot 100$$

The resulting score serves as the system's initial estimation of the overall trustworthiness requirement by the user. Moreover, the overall score is decomposed into per function contributions:

$$nLoTw_j = \frac{W_{TF_j} REG_{TF_j}}{\sum_{j=1}^{5} W_{TFj}}$$

So that:

$$nLoTw = \sum_{j=1}^{5} nLoTw_j$$

The modular design of this architecture allows for future extensibility-supporting the integration of domain-specific encoders, alternate transformer variants, or task-adaptive heads to improve generalization in evolving trust scenarios.

Performance Analysis of Regression Component:

In order to evaluate the Cognitive Coordinator's ability to perform inference on trustworthiness requirements from user provided intents, a BERT-based model with a regression head was fine-tuned. A labeled dataset that contained expressions in natural language was mapped to numerical trustworthiness scores provided by domain experts. The model was trained with the goal to predict a five-dimensional (Safety, Security, Privacy, Resilience, Reliability) trustworthiness vector scaled from 0 to 100. The dataset was split randomly into training and validation sets using a ratio of 80/20, ensuring that all trust functions were equally represented to maintain balance.

The training was conducted using a batch size of 32 with data augmentation enabled to increase robustness and generalization. The model was trained for up to 500 epochs, but epoch 14 was selected as the evaluation point based on the lowest validation loss and highest R^2 observed on the validation set, demonstrating high generalization performance without signs of overfitting. The validation loss, which was computed by the mean squared error on the evaluation set, reached 0.006, confirming the model's ability to generalize well outside the training data. The following metrics were computed on the evaluation script.

Mean Absolute Error (MAE)	**0.055**
Mean Squared Error (MSE)	0.006
Root Mean Squared Error (RMSE)	0.076
R^2 Score	0.907

Table 1 CoCo - Performance of the regression component.

The following plots further illustrate the learning behavior. On Figure 2, the training and validation losses (i.e. MSE) decrease in parallel without any significant divergence, showcasing good generalization and low risk of overfitting. On Figure 3, two additional error metrics (i.e. MAE, RMSE) exhibit a downward trend, indicating a reduction in the prediction error. Lastly, on Figure 4, the coefficient of determination increases in a steady trend, confirming improved fit to the underlying data distribution.

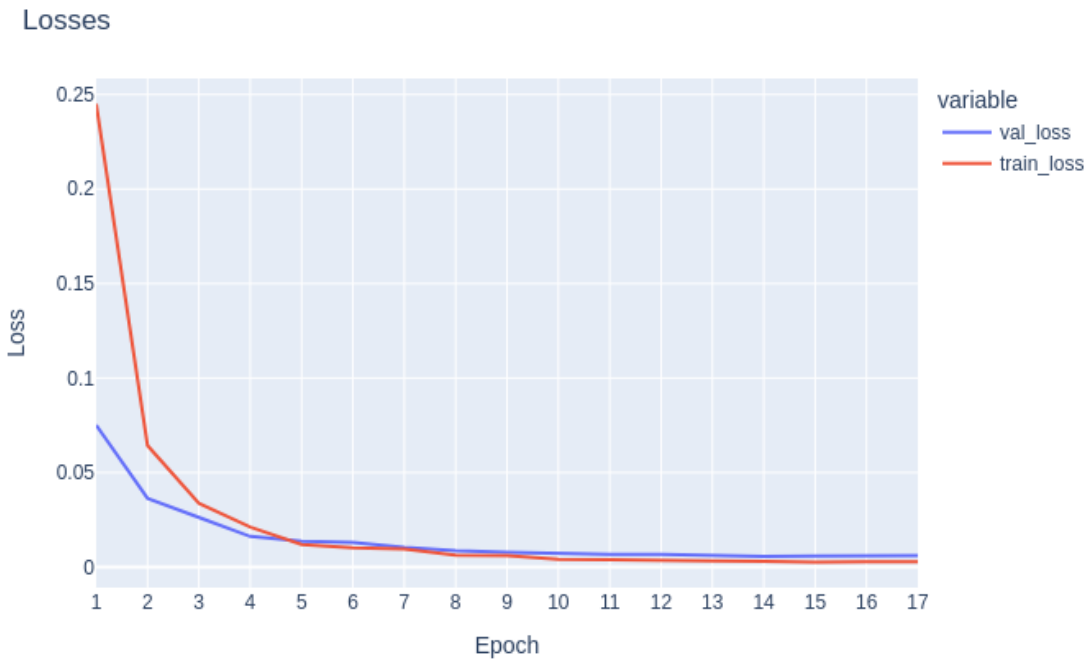

Figure 2: CoCo - Evolution of MSE during the training process.

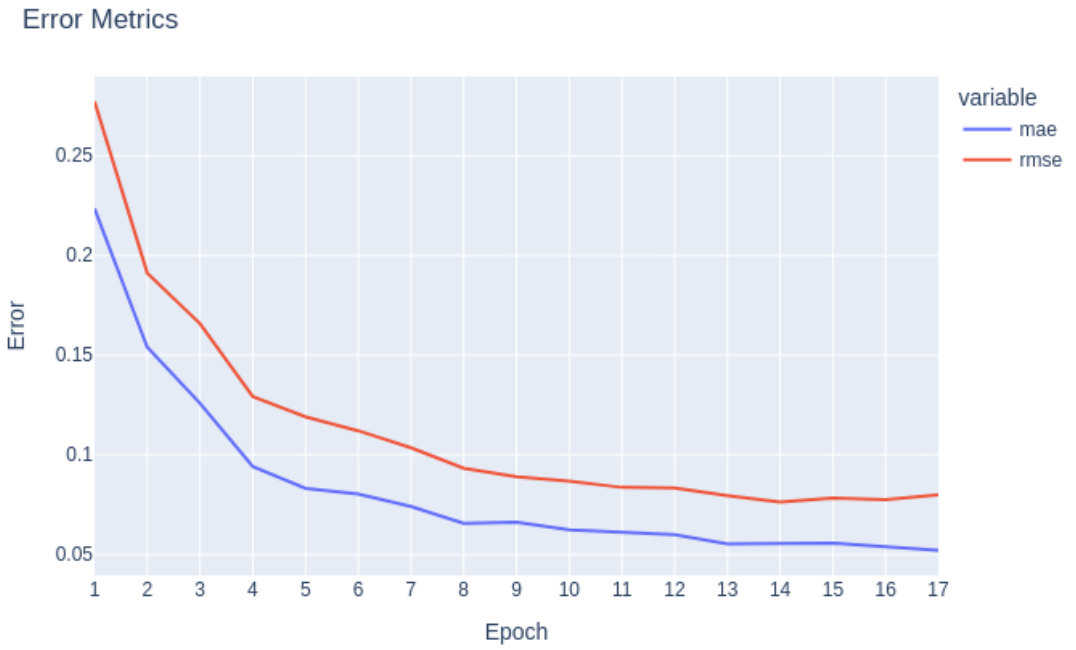

Figure 3: CoCo - MAE and RMSE evolution on the validation set.

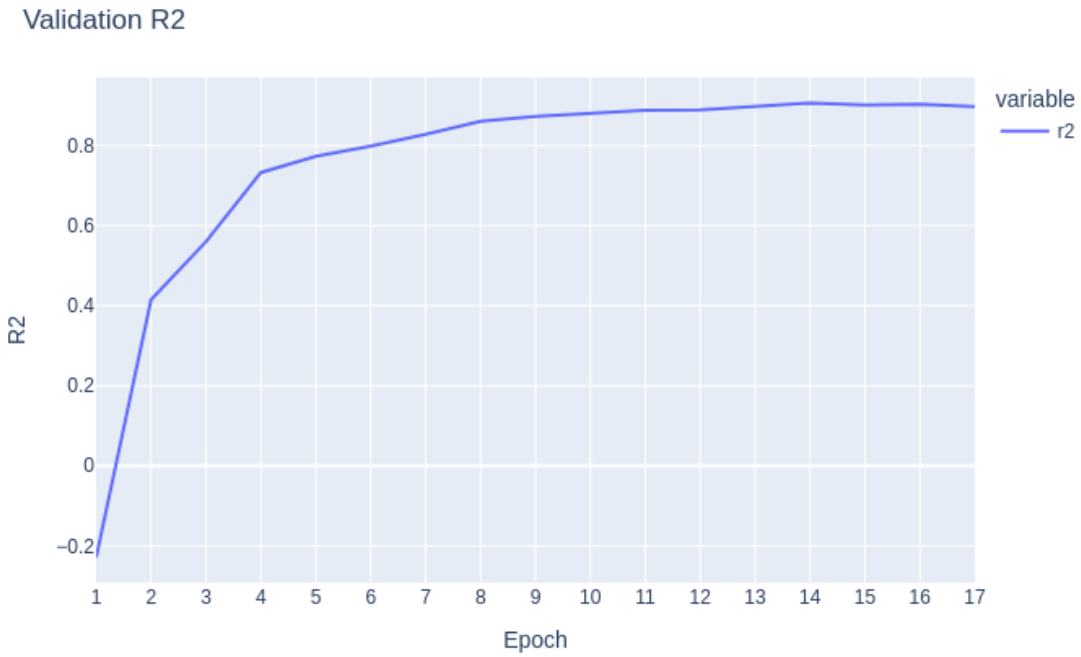

Figure 4: CoCo - R-squared evolution on the validation subset.

2. **Reasoning Engine:** The Reasoning Engine constitutes the context-aware calibration stage of the SAFE-6G trust pipeline. Its primary function is to refine the nLoTw into a realistic and deployable cLoTw by incorporating system-wide resource constraints into trust computation. It consists of a Knowledge Base and a Conflict Resolution Manager as presented in

3. Figure 1, which constitutes the fabric of the reasoning engine's adaptability to dynamic conditions. In a deeper dive, the knowledge base is the main force that drives the conflict resolution's outcomes and is composed of:
   - **Flavor knowledge:** Flavors specify the deployments to be executed at the nApp and vApp levels for each TF. They are organized into tiers—None, Low, Medium, and High—and mapped to trust-score thresholds, along with the corresponding actions required for deployment. Along with the flavors, hard rules are established, indicating the possible conflicts between those actions. In a more explanatory manner, high security may require a higher level of logging, while high privacy requires little to no logging, which produces a conflict.
   - **User knowledge:** Just like flavors, the tier approach is utilized here. Users may be defined as privileged or non-privileged primarily due to their placing in their organization or perhaps a paid subscription for specific access to services. Such cases govern the user's access to specific flavors as for example, high reliability flavor could not be provided to a non-privileged user.

The **Conflict Resolution Manager,** on the other hand, is a core component that validates and refines TF action plans prior to deployment. It evaluates the $nLoTw$ scores per TF against the knowledge base and, using Prolog-based inference, determines the maximum feasible trust level $TF_{cap_j} \in [0, 100]$, which represents the highest enforceable score per TF.

Once the maximum enforceable trust scores are established across all trust dimensions, the cLoTw is computed using the following formulation, utilizing the TF_{cap}:

$$cLoTw_j = \frac{\min\left(W_{TF_j} REG_{TF_j}, TF_{cap_j}\right)}{\sum_{j=1}^{5} \min\left(W_{TF_j} REG_{TF_j}, TF_{cap_j}\right)} \cdot 100$$

This formulation introduces a normalization step that adjusts the trust score based on what is achievable. It ensures that high trust demands from the user are honored only to the extent permitted by the system's operational state-preserving both trust integrity and infrastructure efficiency. The resulting cLoTw and the final set of actions are then used by the Cognitive Orchestrator to guide the deployment of each TF in alignment with real-world execution boundaries.

4. **Cognitive AI Orchestration:** This workflow acts as the central control and coordination hub within the SAFE-6G framework. It is responsible for managing the orchestration pipeline that connects both internal reasoning components (i.e., Regression component, Reasoning Engine, and XAI) and external execution layers (i.e., Trust Functions, the SAFE-6G Orchestrator, MLOps Framework, and the CAPIF interface).

A core enabling feature of this orchestrator is its embedded message broker, which establishes a decoupled, asynchronous, and scalable communication model across the SAFE-6G ecosystem. The message broker facilitates inter-component interactions by adopting a publish-subscribe and/or message-queue-based architecture. This ensures:

- **Decoupling of producers and consumers:** Components such as the Regression module can emit scores without requiring direct coupling to subscribers (e.g., TFs).

- **Fault tolerance and resilience:** Delays or failures in individual modules (e.g., TF deployment agents) do not affect upstream components, enabling robust pipeline execution.
- **Scalability and extensibility:** New modules or services (e.g., future Trust Functions or explainability plugins) can subscribe to relevant topics without architectural reconfiguration.

Through this brokered interaction model, the orchestrator collects partial results-such as the vector of regression scores, computed weights, per-TF capabilities, and context metadata-and synthesizes them into a consistent and actionable cLoTw.

3.3 Robustness Assessment of Cognitive Coordinator Language Model

The rapid evolution of 6G networks toward intent-driven, autonomous operation has underscored the necessity for cognitive systems that can reliably interpret and enact user-defined trust requirements. However, as these systems increasingly rely on large-scale language models—particularly transformer-based architectures—they become susceptible to adversarial manipulations that can undermine their capacity to deliver secure and trustworthy services. To this end, our work advances the systematic evaluation of adversarial robustness in BERT-based cognitive coordinators, with a focus on quantifying their resilience against targeted perturbations within the user-intent classification pipeline.

The experimental framework centered on a five-head regression model constructed atop a pretrained BERT encoder. Each regression head independently predicted the degree of compliance with a specific trust dimension—Security, Safety, Privacy, Resilience, and Reliability. The dataset used for training and validation was meticulously curated through expert annotation, encompassing diverse textual expressions of user trust requirements. Data augmentation strategies, including synonym substitution and normalization, were applied to improve generalization under varied linguistic formulations.

To characterize robustness under adversarial conditions, three complementary black-box attack strategies were implemented: TextFooler, BERT-Attack (BAE), and Probability Weighted Word Saliency (PWWS). These attacks, which systematically replace salient tokens with semantically plausible alternatives, were selected to evaluate the system's susceptibility to realistic manipulations that preserve surface coherence but induce significant prediction shifts. For each attack, four key indicators were measured:

- **Score Deviation:** quantified the average magnitude of prediction change induced by adversarial perturbations.
- **Mean Squared Error (MSE):** assessed the cumulative prediction error relative to unperturbed inputs.
- **Perturbation Coherence:** measured the semantic similarity between original and adversarial examples.
- **Success Rate:** reflected the proportion of adversarial inputs yielding prediction deviations above a specified threshold.

Empirical results revealed notable variation in model resilience across architectures. Among the evaluated models, RoBERTa-Base demonstrated the highest vulnerability, evidenced by both elevated Score Deviations and Success Rates under all attack modalities. In contrast, BERT-uncased exhibited the most balanced performance, combining lower sensitivity to perturbations with high semantic coherence preservation. ALBERT-Base, while efficient in parameter usage, showed mixed

robustness—performing well against TextFooler but exhibiting increased susceptibility to BAE and PWWS. These findings illustrate the critical influence of pretraining corpus composition and model sparsity on adversarial resistance.

Attack Type	*Model*	*Score Deviation*	*MSE*	*Perturbation Coherence*	*Success Rate*
TextFooler	BERT-uncased	0.102	0.019	0.896	14.79%
	RoBERTa	0.208	0.079	0.985	40.14%
	Albert	0.125	0.024	0.992	16.19%
	Electra	0.165	0.038	0.982	33.09%
BAE	BERT-uncased	0.159	0.044	0.617	33.10%
	RoBERTa	0.208	0.081	0.947	39.43%
	Albert	0.157	0.043	0.884	24.64%
	Electra	0.171	0.041	0.958	35.21%
PWWS	BERT-uncased	0.127	0.028	0.657	22.34%
	RoBERTa	0.228	0.097	0.802	43.66
	Albert	0.186	0.063	0.735	28.87%
	Electra	0.192	0.051	0.812	43.66%

Table 2 CoCo - Adversarial Resistance Evaluation

Importantly, the experimental analysis highlighted specific classes of adversarial transformations capable of inducing severe prediction inconsistencies. Subtle replacements of domain-relevant tokens (e.g., substituting "AI-driven" with "Army Intelligence-driven") led to marked prediction drift, underscoring an overreliance on isolated lexical cues rather than broader semantic context. This observation emphasizes the necessity of integrating contextual verification mechanisms, such as counterfactual consistency checks or adversarial training protocols, to mitigate model fragility in operational deployments.

Original Text	*Adversarial Text*	*Original Score*	*Adversarial Score*
Reliable application	dependable practical application	0.205	0.679
data retention policy	information keeping insurance policy	0.705	0.506
Full device trust validation	mobile device trust validation	0.855	0.364
AI-driven capacity planning	Army Intelligence - drive capacity planning	0.84	0.512
Protect against unauthorized modifications to network infrastructure	Protect against unauthorized modifications to network framework	0.82	0.54
Conduct safety reviews for digital transformation initiatives	carry out safety reviews for digital transformation initiatives	0.705	0.49

Provide secure and reliable infrastructure solutions	Provide protected and dependable framework solutions	0.755	0.511
Implement safe network segmentation strategies	apply safe system segmentation strategies	0.745	0.444
Create ontology of intent for resilience function	generate ontology of intent for robustness operation	0.717	0.408
Perform autonomous resource orchestration	execute autonomous asset orchestration	0.878	0.548

Table 3 CoCo - Adversarial Resistance Scores

The implications of these results are twofold. First, they substantiate the assertion that adversarial robustness must be treated as a first-class design objective when deploying cognitive coordinators in 6G networks. Second, they demonstrate the feasibility of applying systematic adversarial evaluation frameworks—originally developed in NLP contexts—to assess and harden the trustworthiness quantification pipelines integral to network orchestration. Future work should explore hybrid architecture combining transformer encoders with lightweight semantic filters, as well as domain-specific adversarial training regimes tailored to communication systems.

3.4 Role within the SAFE-6G System

The CoCo serves as the pivotal intelligence core within the SAFE-6G architecture, playing an integrative, mediating, and decision-enabling role across the framework. The coordinator begins its role by acting as the semantic interpreter of user intents. Through collaboration with the AI Chatbot and its NLP stack, it deconstructs natural language inputs into structured representations. These are then classified against the TF categories using the predefined dataset to determine their contextual trust significance. This upfront classification sets the trajectory for subsequent orchestration actions. One of the most essential responsibilities of the coordinator is the dynamic calculation of the Level of Trustworthiness. It begins with a regression-style extraction of the nLoTw-based purely on initial input-and progresses to a calibrated, realistic value (cLoTw) through reasoning and historical alignment. This enables the system to differentiate between aspirational trust levels and operational feasibility, avoiding overcommitment or underperformance.

Once trust requirements are determined, the coordinator directs the activation, configuration, and interaction of the five TFs. It ensures that each TF:

- Understands its trust quota.
- Has the resources and contextual insight to act.
- Engages in intelligent action loops.
- Communicates outcomes and challenges.

The system's ability to reactively and proactively adjust is governed by the coordination logic embedded in the Orchestrator module, further enriched by TF feedback and Monitoring Framework data.

The coordinator operates in a closed-loop fashion. Every trust function operates in deployment-feedback mode, returning intermediate results and alerts to the coordinator. Based on this feedback, the coordinator may revise cLoTw values, request model updates from MLOps, or issue reconfiguration commands. This cycle ensures continuous alignment with user-defined trust objectives. Beyond managing TFs, the CoCo acts as the central nervous system for all SAFE-6G intelligence components, maintaining direct or indirect communication with:

- **With MLOps Framework:** Supports structured development, deployment, and monitoring of AI models by enabling reproducibility, version control, performance tracking, and seamless integration into the overall system architecture.
- **With CAPIF:** Guarantees API-based connectivity and dynamic data provision.
- **With Monitoring Framework:** Maintains operational situational awareness.
- **With 5G/6G network:** Actions performed across the USN and NSN domains, and provision of relevant data to be used by the neighbouring components.
- **With Chatbot:** Enables user interaction and intent collection through natural language interfaces.
- **With Edge-Cloud Continuum**: Supports scalable, low-latency execution of services across distributed computing resources.
- **With Metaverse Application**: Facilitates immersive user experiences and context-aware interaction in virtual environments.

In this broader interoperability context, SAFE-6G ensures that the CoCo's intelligence can be effectively translated into end-to-end trust actions across WP3 and WP4 components. The exchange between the Chatbot and the CoCo is implemented through lightweight API calls, allowing intents generated from natural-language interaction to be forwarded seamlessly into the cognitive pipeline. Once processed, the CoCo communicates with the five TFs through an internal message-broker mechanism, enabling asynchronous, event-driven delivery of information and signals. This decoupled architecture allows TFs to operate autonomously while still remaining coherently aligned with the CoCo's global reasoning.

Each Trust Function then executes its actions across the three SAFE-6G planes—the edge-cloud continuum, the user-centric 6G core, and the application layer—by consuming the APIs exposed through the OpenCAPIF framework. Through CAPIF, TFs gain standardized access to network capabilities, telemetry, and service-level controls, ensuring interoperability with WP3 components and enabling consistent enforcement of trust decisions across distributed domains. Collectively, this architecture establishes an operational flow, meaning that the user intents enter through the Chatbot, are semantically processed by the CoCo, orchestrated via message-based coordination to the TFs, and finally applied across the continuum through CAPIF-exposed interfaces, achieving a seamless, closed-loop trust management cycle.

4 (X)AI AGGREGATOR

4.1 ROLE WITHIN THE SAFE-6G FRAMEWORK

As presented in Section 2.2 (Role of AI and XAI in 5G trust coordination), XAI aimed at addressing a better transparency of Machine Learning models to improve trust and adoption of AI. This is an important step in SAFE-6G project as the Trust Functions are managed by AI controllers, which decide about resource allocation and security.

In terms of operational context, the XAI module will need to interact with the user via the Cognitive Coordinator—primarily through its chatbot. Whenever the chatbot detects that a user utterance requires an explanation of a Trust Function decision, it will forward the request to the XAI module. As described in Figure 5, the XAI module requires additional information to propose an explanation that answers user's query.

The information exploited by the XAI assistant to make its decision is of two natures. First, it requires a domain description describing the context meaningful to evaluating the user query. For instance, if a user asks why, it can't access a given resource, the module will need to know that we are in the context of a SAFE-6G platform, that the security issues are handled by the Security TF, and that this TF relies on specific features to take its decision. All this information, given in natural language, will help the XAI module analyse the user's natural language query – with an LLM – to identify which TF is concerned and for which input features. This overall description of the context of operation is provided by each TF implementation based on implementation choices made in each TF AI submodule. It summarizes, for each TF AI model used, the domain-defining XAI characteristics as defined in 4.3.1. In addition, it specifies the role of each input/output feature of the AI model in natural language.

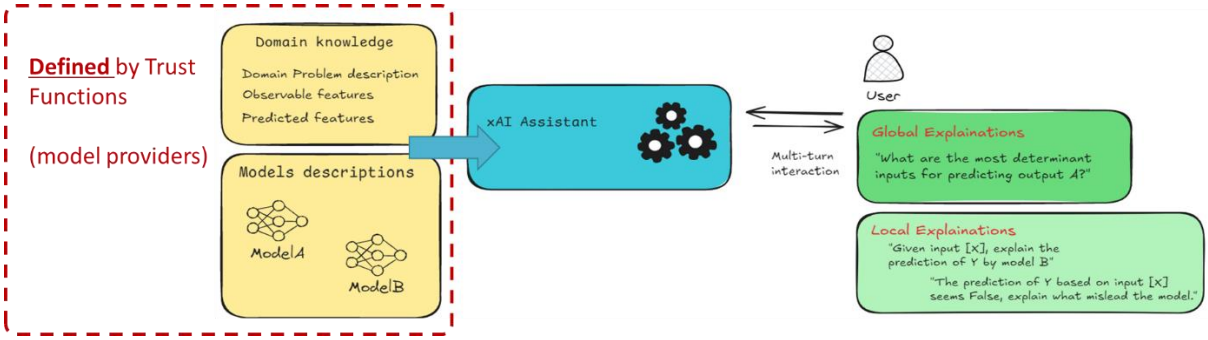

Figure 5: XAI - Overview of Operational Context.

4.2 ORCHESTRATION LOGIC

The orchestration logic of the XAI Module is described in the following Figure 6. This sequence diagram is high-level and requires further refinement to make API calls between components more concrete and instantiable.

On the high-level, the idea is that the XAI is triggered by the user when it asks for explanations on decisions taken by the framework. Thus, the first actor of this triggering is the Chatbot module: by detecting the user intention to get an explanation, it will trigger the CoCo. The XAI module will require contextual information to produce an explanation, for instance:

- What is the prediction to explain
- Who (i.e. which TF) made this prediction
- Contextual information about the TF who made the prediction

Thus, it is required that the chatbot goes through the CoCo, who is responsible for enriching the required context before triggering the XAI module.

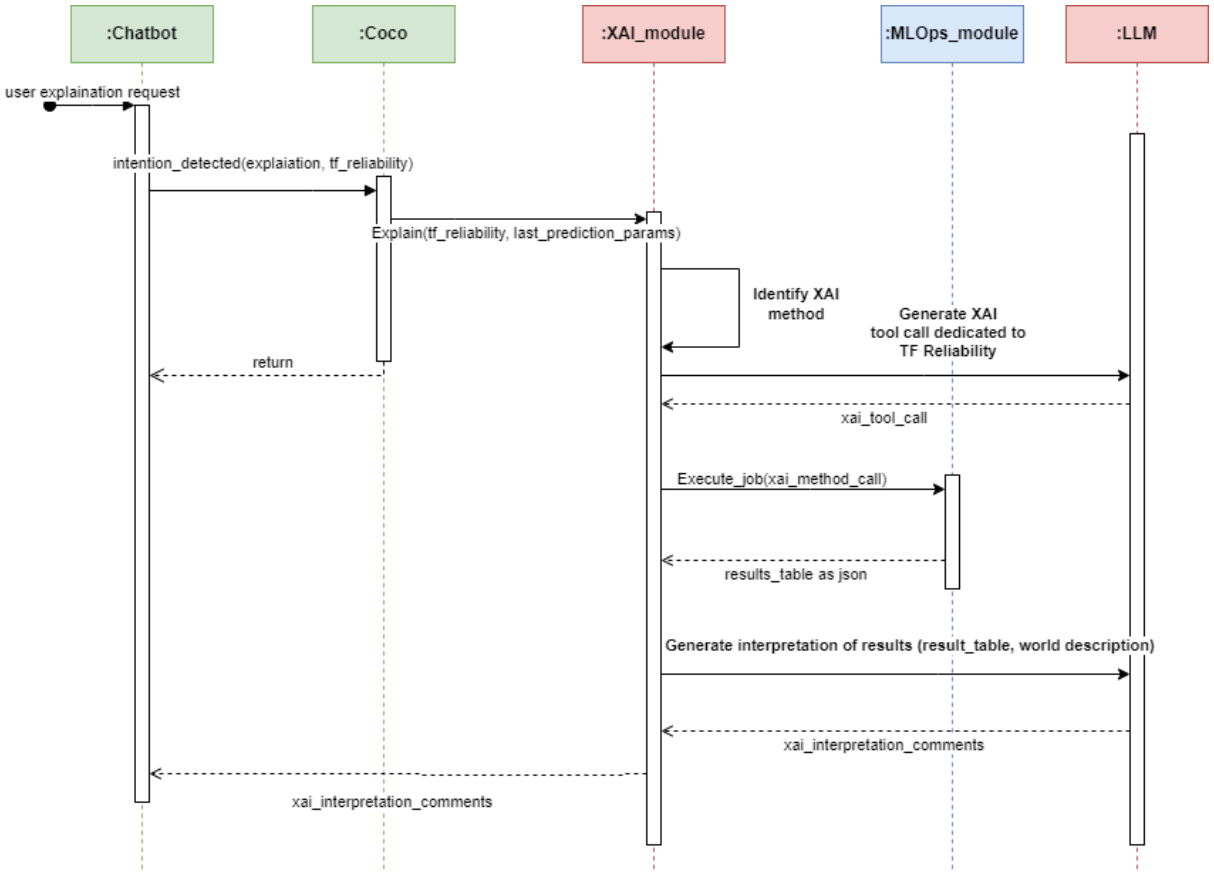

Figure 6: XAI - Orchestration Logic Sequence Diagram.

As can be seen, the XAI module will then have to make three different operations:

- **XAI method selection:** based on the user query and on the description of the TF AI, the module must identify which method(s) to use to generate an explanation.
- **XAI method application:** the module must trigger a concrete API call to launch computations of the selected XAI method(s). This relies on the MLOps offering the capability to trigger Jobs and wait for them to end.
- **XAI results interpretation:** XAI method results are often hard to interpret for a non-expert. Reusing an LLM's general knowledge and combining it with a description of the TF input features used for the prediction, this phase will generate a more understandable interpretation of the XAI technique results and deliver it as natural language to the chatbot.

Several inputs of the XAI Module are in natural language, in particular, feature descriptions & user queries. Thus, to provide an interpretable explanation of the prediction, the XAI module will make use of an LLM in at least the last two stages.

The next paragraphs demonstrate the current status of our exploratory work on each of these three steps of automation.

4.3 Development of Local and Global XAI Tools

In this section, the current advancements in the development of an XAI module designed to provide explanations regarding the predictions and decisions rendered by individual TFs within the 6G platform are discussed. The primary objective of this XAI module is to equip users with the ability to critically question and interpret the system's outputs. However, a significant challenge arises from the existing landscape of XAI research, which tends to be highly technical and intricately linked to ML algorithms. While these methodologies serve as proficient tools for ML experts seeking to decipher unexpected model outcomes, they create a pronounced semantic chasm between their technical and mathematical explanations and the more accessible, high-level interpretations necessary for the average user. Our aim with this module is to bridge this gap, facilitating non-experts' engagement with current XAI techniques without requiring an in-depth understanding of their complex mechanics.

To accomplish this, a repertoire of XAI techniques (Section 4.3.1) pertinent to explicating individual TFs were initially identified. Subsequently, an architectural framework (Section 4.3.2) that addresses the semantic and expertise divide between non-experts and the practical application of these techniques is proposed. Although this architecture remains in development, its potential applications through concrete examples in Section 4.3.3 will be illustrated. Finally, the conclusion will articulate the vision for the subsequent steps necessary to realize the proposed architecture.

4.3.1 Identifying a catalogue of XAI methods

A preliminary survey of XAI methods helped us identify the different characteristics that must be considered when applying explainability for the SAFE-6G platform. These characteristics are called the "***XAI applicability characteristics***" as they will determine the applicability of techniques to a given context:

- **Explanation types**: As explained in Section 2.2, the categories of explanations required by the user are traditionally divided into two categories: global or local. Different XAI techniques may be necessary depending on the nature of the explanation requested.
- **Input data type**: AI models can handle a wide range of input data types: images, time series, audio signals, tabular data, text, etc. Not all of these are adequately covered by appropriate XAI techniques. However, in the context of the SAFE-6G project, the data used by the TFs to make their predictions is of a tabular nature. Therefore, the focus will be on a selection of methods suitable for tabular data
- **Addressed Task**: AI models can perform tasks of different natures: classification, regression, anomaly detection, generation, etc. Current XAI methods are generally limited to classification and regression tasks, although recent work has started to address the explanation of generative models. In this study, the focus will be on methods for explaining regression or classification tasks.
- **Supported model nature**: Some methods require knowing the exact nature of the model as well as the values of the model's parameters. By exploiting the nature of the model—for

example, using the gradient of a neural network's loss function—one can interpret the model's predictions. These are called "white-box" models, as opposed to "black-box" methods, which provide explanations without manipulating the interior of the model. In black-box approaches, explainability is generally achieved by repeatedly stimulating the model with different inputs to draw conclusions from the averages. In this case, no restriction is imposed regarding the white-box or black-box nature of the approach, as systematic access to information about the model is assumed.

- **Compatibility restrictions**: The implementation of the methods used may impose compatibility restrictions with the framework used to implement the model (TF, Keras, PyTorch, SKLearn, etc.). This is especially true for "white-box" methods. Therefore, this parameter must be considered.

For each explainability request, we will rely on these characteristics to determine which method is most appropriate for the context. In terms of implementing these methods, various available libraries have been analysed and chosen to base our work primarily on the Alibi Python library, which encompasses a broad range of explainability methods applied to tabular data. The integration of new explainability methods from other libraries is not excluded (e.g., Captum, which is dedicated to explaining PyTorch neural networks) as the implementation choices for the TF models evolve.

Thus, the current XAI module integrates the following explainability techniques:

- **Accumulated Local Effects (ALE)**: ALE is a global explanation method that quantifies the effect of a feature on the model's prediction by accumulating local changes across its range. Unlike Partial Dependence Plots (PDPs), ALE accounts for interactions between features, providing a more accurate representation when features are correlated. This method is particularly useful for understanding the influence of individual features in complex models.
- **Partial Dependence (PDP)**: PDPs illustrate the relationship between a feature and the predicted outcome by averaging the predictions over the distribution of other features. This global method helps in visualizing the marginal effect of a feature, aiding in understanding how changes in that feature influence the model's predictions.
- **Partial Dependence Variance (PDV)**: Building upon PDPs, this method focuses on the variance in predictions as a feature changes, providing insights into the stability and uncertainty of the model's behaviour with respect to that feature. It is particularly useful for identifying features that cause high variability in predictions.
- **Permutation Importance**: This technique assesses the importance of a feature by measuring the decrease in model performance when the feature's values are randomly shuffled. A significant drop in performance indicates a high dependency on that feature, offering a global perspective on feature importance.
- **Anchors**: Anchors provide local, interpretable explanations by identifying high-precision decision rules that "anchor" a model's prediction. These rules are sufficient to ensure the same prediction for similar instances, making them particularly useful for explaining individual predictions in a human-understandable manner.
- **Contrastive Explanation Method (CEM)**: CEM offers local explanations by identifying the minimal set of features that lead to a particular prediction (pertinent positives) and those that should be absent to change the prediction (pertinent negatives). This contrastive approach provides a comprehensive understanding of the decision boundaries in the model.
- **Counterfactual Explanations**: Counterfactual explanations describe what minimal changes to the input would have resulted in a different prediction. This causal reasoning approach helps

users understand the factors influencing a model's decision and is particularly useful in domains requiring actionable insights.

- **Prototype Counterfactuals**: This method combines counterfactual explanations with prototype-based reasoning by identifying the closest examples in the training data that differ from the current instance. It provides context to the counterfactuals, enhancing their interpretability by relating them to representative instances.
- **Counterfactuals with Reinforcement Learning**: Integrating reinforcement learning with counterfactual explanations, this approach generates explanations by exploring the impact of different actions on the model's predictions. It is particularly useful in dynamic environments where decisions are sequential and interdependent.
- **Similarity Explanations**: This approach provides local explanations by identifying instances in the training data that are most similar to the current instance. By highlighting these similar examples, it helps users understand the model's decision-making process in a context they can relate to.
- **Integrated Gradients**: Integrated Gradients is a gradient-based method that attributes the prediction of a model to its input features by integrating gradients along a path from a baseline to the input. This technique ensures sensitivity and implementation invariance, making it suitable for deep learning models.
- **Kernel SHAP**: Kernel SHAP is a model-agnostic method that computes Shapley values to attribute the prediction of a model to its input features. By approximating Shapley values using a weighted kernel function, it provides both local and global explanations, making it versatile across different model types.
- **Tree SHAP**: Specifically designed for tree-based models, Tree SHAP efficiently computes Shapley values by leveraging the structure of decision trees. It provides exact explanations for individual predictions and is particularly useful for interpreting models like random forests and gradient boosting machines.

XAI method	Explanation Nature	Provided Explanation	Addressed Task	Input Datatype	Supported Model Nature	Compatibility Restrictions
Accumulated Local Effects (ALE)	Global	Feature Importance	Classification & Regression	Tabular	Black-Box	
Partial Dependence (PDP)	Global	Feature Importance	Classification & Regression	Tabular (+Categorical)	Black-Box	
Partial Dependence Variance (PDV)	Global	Feature Importance	Classification & Regression	Tabular (+Categorical)	Black-Box	
Permutation Importance	Global	Feature Importance	Classification & Regression	Tabular (+Categorical)	Black-Box	
Anchors	Local	Local Rules	Classification	Tabular	Black-Box	
Constative Explanation Method (CEM)	Local	Pertinent Features	Classification	Tabular	Black-Box (Differentiable)	TensorFlow
Counterfactual Explanations	Local	Prototype	Classification	Tabular (+Categorical)	Black-Box (Differentiable)	TensorFlow
Prototype Counterfactuals	Local	Prototype	Classification	Tabular (+Categorical)	Black-Box (Differentiable)	TensorFlow
Counterfactuals with Reinforcement Learning	Local	Prototype	Classification	Tabular (+Categorical)	Black-Box	
Integrated Gradients	Local	Feature Importance	Classification & Regression	Tabular	White-Box	TensorFlow

Similarity Explanations	Local	Prototype	Classification & Regression	Tabular (+Categorical)	White-Box
Kernel SHAP	Global & Local	Feature Importance	Classification & Regression	Tabular (+Categorical)	Black-Box
Tree SHAP	Global & Local	Feature Importance	Classification & Regression	Tabular (+Categorical)	White-Box

Table 4 XAI - Catalogue of available Explainability Techniques

The table above lists the techniques available in the XAI module catalogue and classifies them regarding each *XAI applicability characteristics*. The XAI module will rely on these characteristics to select the most appropriate technique to apply. The following section explains how the architecture of the XAI module supports this decision-making and instantiates the decision.

4.3.2 EARLY EXPERIMENTS WITH TF FUNCTIONS

To understand if our catalogue matches needs for the TFs and to understand the challenges of automating XAI, XAI techniques were applied to a couple of simple samples on a first experimental version of the Reliability TF.

In this experimental version, the Reliability TF provided several AI models trying to predict the CPU usage in a 5G video streaming application. The dataset used[16] to train these models was composed of the following features:

- Input Features
  - N: number of User Equipments that are currently downloading more than 100 kbps. No unit.
  - R [Mb/s]: aggregate downlink bitrate from the gNB (5G network node) to all the User Equipment. In Mb/s.
  - O [Mb/s]: outbound traffic at the pod's interface, transmitting the video(s) packets. In Mbps.
- Output/Predicted Feature
  - CPU[mc]: CPU usage at the server pod. In millicores (mc).

In this case, the task is a regression task. As can be seen in Table 4 it restricts the set of XAI techniques that can be used.

Five different model types were experimented by the Reliability TF owners, but it was decided to focus on the Random Forest model as it was the best statistical predictor and later added the DNN model (Deep Neural Network) for diversity. This way, both classical/statistical and neural architectures are explored.

Local explainability

Local explainability on the DNN model was first tackled, trying to answer the question: "Why is my predicted CPU load so high? ". An XAI expert might reformulate this question as "W*hy does my model make prediction of CPU=25.23 for the input [N=2, R=12.82, O=3.18]?"*.

Figure 7 shows the results of applying the Integrated Gradients method to the DNN model. This method attributes an impact score to each feature in the input. Both the amplitude and direction

(positive or negative) of the attribution score reflect the impact of the inputs. In this case, the most important input features seem to be both O and R. Their score indicates that the higher O and/or R, the higher the CPU load. On the contrary, the feature N seems to be negatively correlated to the prediction, indicating that the higher N (for given values of O and R), the lower the CPU load. However, the impact of N is 4 times less important than O or R.

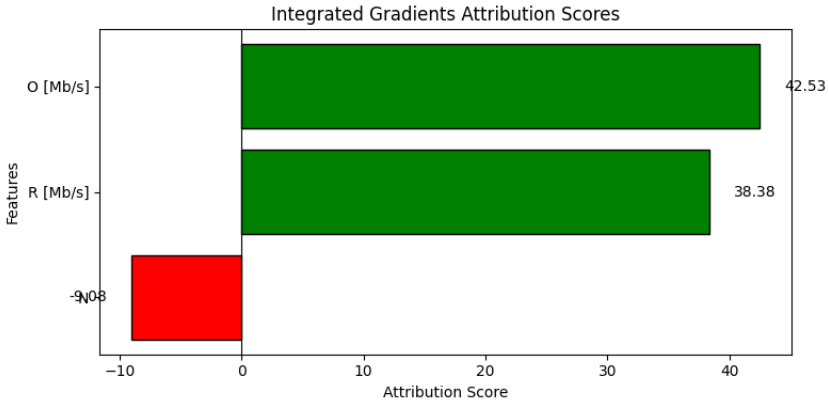

Figure 7: XAI - Feature Attribution Scores.

Reading this table of results is very technical and requires a deep understanding of both the features and the XAI technique used. Consequently, one would require the help of an AI expert to conclude that "*in this regime, the CPU load is mainly driven by both the incoming and outgoing traffic.*"

Note also that the same technique for the Random Forest model cannot be applied since the method is not differentiable, which is a requirement to apply IntegratedGradient. To conclude, the help of an XAI expert was required at **both ends** of the process. First, to select the technique most adapted to a given AI model architecture, and after execution of the technique, to interpret the results/graphics it produces.

Global explainability

Some **global explainability** techniques were also explored. Again, many of the techniques in this field focus on attribution of scores to the input features. However, in this case, they aggregate results from a representative set of data so that we can draw conclusions on general model trends. In our case, we tried to explain the general trend of the Random Forest model with the ALE method.

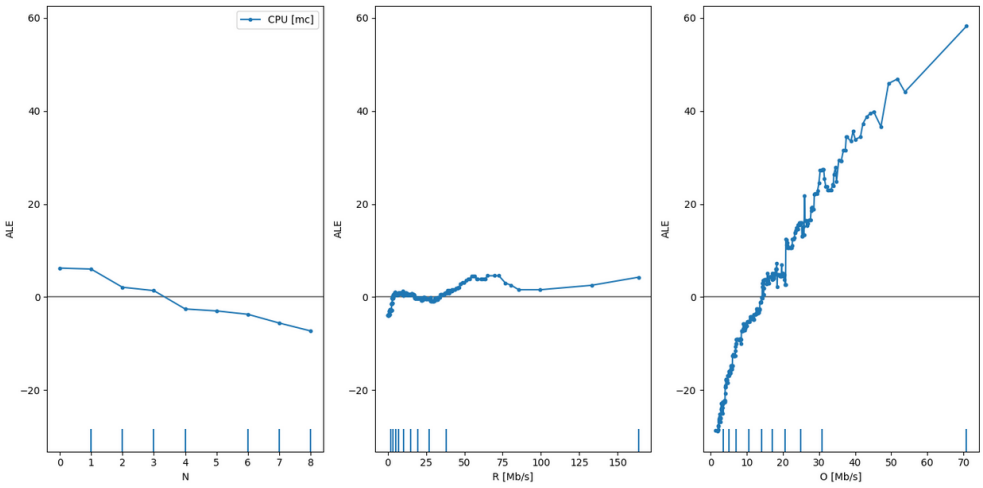

Figure 8: XAI - Features ALE scores.

Figure 8 illustrates the ALE scores of each of the three input features. Contrary to the local method presented above, we have here a summary of how features affect the model predictions. In addition, at the bottom of each diagram, we have a histogram of features values, giving an idea of how features are distributed in the evaluation dataset and consequently, which degree of confidence we may have in the diagram. For instance, the right parts of the diagrams for R and O seems unequally represented in the dataset, so conclusions derived from this part of the diagram are questionable.

The general conclusion of these diagrams would be that O seems to be the variable explaining most of the decisions made by the Random Forest model, with much fewer influence from N and R. Like in the local case, we also note that the N influence is negative, meaning that the CPU loads diminishes when N is higher. This could be either explained by real-world technical justifications – such as a better parallelism – or by a bias in the evaluation dataset - e.g. if all examples of high N correlate with low values of O whereas the reality is more varied.

We note once again that selecting the appropriate XAI method and interpreting the results requires expert knowledge and is not truly within the reach of the average user. We propose, in the next section, an architecture that tries to automate the application of such XAI techniques by tackling the integration of expert knowledge and decision-making from the user query to the generation of a user understandable interpretation of the AI models predictions.

4.3.3 Architecture definition of XAI module

As mentioned in Section 4.1, the XAI module will break down its activity into three different steps, each corresponding to a sub-module, as summarized in Figure 9. The three individual sub-modules share access to internal resources:

- A description of the application domain knowledge and of the TF models.
- The tools catalogue defined in Section 4.3.1.
- A set of callable APIs, one for each of the method of the tools catalogue. Individual XAI tools will be hosted and triggered through the MLOps module, as this module allows for instantiating computing resources on-the-fly and easy access to the AI models of the TF.

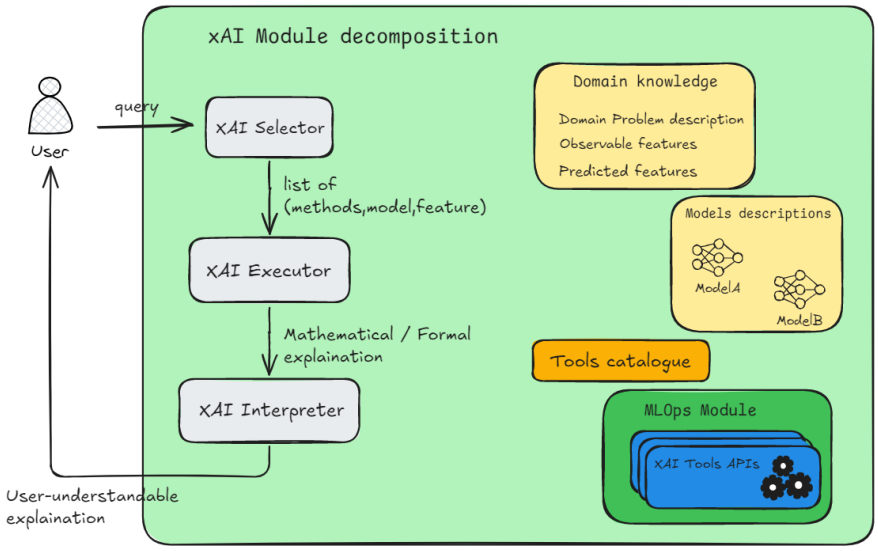

Figure 9: XAI - Decomposition of Sub-Modules.

The **XAI selector** sub-module is responsible for choosing one or several methods to run on the TF AI models. To make this selection, the XAI selector may have to reformulate the intent of the user to match it with the catalogue's description of the available tools. It also needs to rely on domain description to understand the role of individual features in the current application and match them with the query. The challenge of this module is that it requires matching both natural language – user query and domain description – while at the same time taking into account strict criteria – the *XAI applicability characteristics* –, which calls for a more formal/logical application. To achieve such hybrid reasoning, we currently focus on the design of a decision-tree where some decision nodes could be delegated to an LLM for intent-based matching. As an output, the sub-module will produce a list of triplets <xai_method, ai_model, [features_list]>. Each of these triplets describes the application of an XAI method on a given AI Model from a TF, and a list of features of interest to be explained. Most of the time, it is expected the list to restrict to one only method, but we keep this option open to combining multiple explanations in the future.

Once the XAI method selection is done, the job of the **XAI executor** is to trigger the execution of each triplet through the MLOps Module. The different XAI tools available in the catalogue will be exposed as a service by the MLOps Module through a REST API. The role of the XAI executor is to accurately build REST API queries, relying on the API documentation and on the domain description. Once built, the queries trigger execution of the appropriate explainability computation in the MLOps module. The XAI executor then waits for parallel computations to end and collects results. The format of results will be formal (JSON or CSV table), but its precise content depends on each tool.

Finally, the **XAI interpreter** will have the role to first interpret results of the selected XAI methods. To do this, it will rely on the tools description provided in the tools' catalogue and on the API return type description, we aim at providing a very technical explanation to address user's query. However, the explanation, as it builds upon the individual features used by the AI model, maybe cryptic to the final user. To help close the gap between the XAI method explanation and user's high-level understanding, the use of an LLM to reformulate the explanation into a plausible interpretation is proposed. This interpretation will be the result of exploiting the descriptions of the domain and of the observed and predicted features.

4.3.4 INTERACTION API WITH OTHER COMPONENTS

As mentioned above, the XAI module needs some piece of knowledge in order to perform selection and interpretation phases. Selection from the XAI Catalog requires some knowledge about the model's nature, whereas interpretation requires a natural language description of the features manipulated by the model.

More formally, to perform an explanation request, the module will need an object which summarizes this information. The schema this object must obey is specified in Table 5 below.

xai_parameter type		
Attribute Name	**Data Type**	**Description**
model_description	string	Natural language description contextualizing the usage of the model by the Trust Function. Used by the interpreter to formulate user friendly interpretations of the interpretability of models predictions.
model_location	URI	Needed to load the model in memory to perform explainability technique
model_architecture	string	Free string to describe the architecture (neural network, SVM, …). May be used by the LLM.
model_differentiability	bool	Differentiability is required for some XAI techniques
model_task	Enum['classification', 'regression']	Used by XAI selector to identify appropriate method in the catalog.
input_features	feature_description	Gives detailed description (name, type, description) of input features. Used by the executor to trigger appropriate XAI method and by the interpreter to provide contextual interpretation of XAI results.
output_feature	feature_description	Gives detailed description (name, type, description) of output feature. Used by the executor to trigger appropriate XAI method and by the interpreter to provide contextual interpretation of XAI results.
last_prediction_inputs	dict	Input features values for the last prediction made by the Trust Function model. Required for local explainability only.
last_prediction_output	dict	Output feature value for the last prediction made by the Trust Function model. Required for local explainability only.
feature_description type		
Attribute Name	**Data Type**	**Description**
name	string	Name of the feature. Used as a key to generate model prediction calls by XAI techniques.
type	type	datatype of the feature (int, float, string, enum…)

description	string	Natural language description contextualizing the usage of the feature. Used by the interpreter to formulate user friendly interpretations of the importance of this feature regarding models predictions.

Table 5 Trust function object schema as input for XAI module

Most of this object information is defined per Trust Function and is quite static for a given implementation of the platform. In order to simplify first integrations, we assume the model file and Trust Function's models' descriptions are statically accessible in the platform. Discussions are in progress to define more clearly the API to dynamically access this information.

4.3.5 Progress and future work on the XAI module

In this section, the status regarding the implementation of an XAI module capable of explaining/interpreting decisions made by individual TF's AI models has been presented.

The state-of-the-art, both in terms of Explainable AI and in terms of LLM-aided decision-making was first analysed. This allowed us to identify a catalogue of XAI methods meaningful in our context. Once the TF owners proposed their first AI models, we tried to manually explain some example predictions to get a grasp, on concrete examples, of what decisions the module needs to take and what knowledge in may rely on to achieve this. Based on this hands-on approach, a decomposition of the XAI module into sub-modules: selector, executor and interpreter is proposed. A high-level distribution of roles and an orchestration of these sub-modules was designed, as well as the description of input knowledge required by the XAI component. This formalisation of the input data required by the XAI component is a first step towards its integration. Further work is required to plan more precisely how this information will be retrieved by the XAI module from other components.

Among the three sub-modules, the XAI **executor** represents more of an integration effort than a real risk to the feasibility of the XAI module. So future work must focus on validating:

- the approach on the XAI **selector** as a combination of logical & intent-based matching techniques.
- the approach on the XAI **interpreter** as an interpretation/reformulation tool based on prompt engineering.

5 User-Centric Safety Function

The user-centric safety function within the SAFE-6G framework, leverages the Software Defined Perimeter (SDP) paradigm to create dynamic, individualized boundaries. By embedding this approach into the 6G Packet Core, the system ensures that access to critical resources is precisely aligned with user-specific requirements and operational context. Rooted in a zero-trust model, the function employs fine-grained micro-segmentation of permissions to minimize unnecessary exposure and maintain robust control. This enables a responsive, resilient network experience tailored to the intent and role of each user.

5.1 Architectural Components of the Function

The architectural components that comprise the safety function integrate various network elements to collect and analyse data, to enable decisions regarding user safety and trust. The core components of this architecture include the proposed Software Defined Perimeter Core Function (SDPCF), Software Defined Perimeter GateWay (SDPGW) and VPN Gateway Function (VGF) which are the three Network Functions that are dynamically deployed, taking into consideration the user request and needs. Also, the SAFE-6G's functional blocks named MLOps, DataOps along with several critical network functions (NFs), highly contribute to the user's level of safety as described below. The interaction between the architectural components and their position in the architecture can be observed in Figure 10.

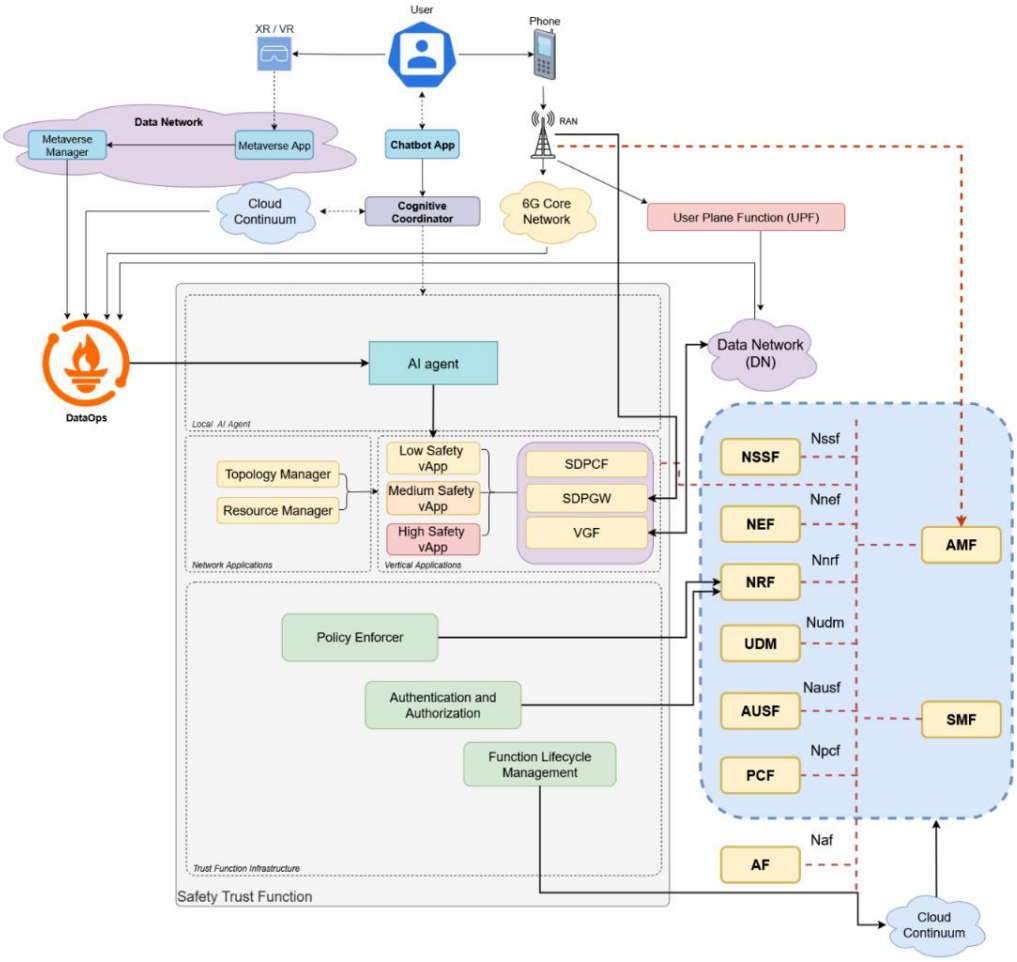

Figure 10: Safety Function - Architectural Components

Following is an overview of the system's architecture, including the roles of these components and how they interact with each other to form an end-to-end solution for safety in the network.

5.2 High-Level Architecture

At a high level, the system consists of multiple network elements that generate relevant logs and data, which are then collected, aggregated, and processed to decide the level of safety. These data are subsequently analysed by the AI layer of the safety trust function, which determines the safety levels of a particular user. The architecture can be visualized in Figure 10 includes the following elements:

- **Network Function Components:** Various Core NFs (SMF, AMF, PCF and UPF) to gather data and logs, including session, mobility, location, traffic patterns, and network policies.
- **aerOS:** Data on network health, resource allocation, and usage patterns.
- **DataOps Pipeline**: DataOps collects, pre-processes, and ensures that the data from these NFs is appropriately transformed, cleaned, and normalized before being fed into the ML pipeline.
- **MLOps Pipeline:** MLOps is responsible for the continuous training and refinement of ML models using the data processed by DataOps. These models generate decisions and classifications that inform the network's safety and trust-related actions.

The communication between these components creates a loop that ensures the safety trust function adapts to evolving user behaviours and network conditions, continuously refining its models and decisions.

5.3 Functionalities per Layer (Local AI Agent, vApps, nApps)

5.3.1 AI Layer of SAFETY Trust Function

The AI layer of the trust function consists of the ML model which will be the decision maker regarding the requested Level of Trust that can be provided. The parameters described above from the various planes will be considered to determine the safety level and what policies will be enforced to the Safety tunnel. Regarding MLOps, the continuous training of the model will assist in the change of the boundaries of those safety levels.

5.3.2 Components

1. **MLOps (Machine Learning Operations):**
   - Function: MLOps is a set of practices and tools that automate the end-to-end lifecycle of ML models. It ensures that the models used to evaluate safety are continuously trained, tested, and updated as new data flows in.
   - Role in the Architecture: MLOps receives processed data from DataOps and uses it to train ML models for decision-making. It ensures the seamless deployment of these models in a real-time environment to make accurate predictions.
   - Key Features:
     - Continuous training using fresh data.
     - Model monitoring and performance evaluation.
     - Version control and model deployment.
     - Automated retraining and tuning based on new data.

2. **DataOps (Data Operations):**
   - Function: DataOps is responsible for managing the data lifecycle, ensuring that data from various sources is efficiently collected, processed, and fed into the MLOps pipeline. It plays a crucial role in ensuring that data is consistent, clean, and formatted properly for analysis.
   - Role in the Architecture: DataOps acts as a mediator between the various network functions (CC, SMF, AMF, PCF, UPF) and the ML models. It collects logs and performance metrics, pre-processes the data (e.g., normalization, feature extraction), and ensures that the data is ready for use in model training and real-time decision-making.
   - Key Features:
     - Data ingestion from multiple network sources.
     - Data pre-processing (cleansing, normalization).
     - Data transformation for ML model consumption.
     - Ensuring data quality and consistency.

3. **Network Functions (NF):** The following network functions contribute valuable logs and data that feed into the system:
   - Session Management Function (SMF):

  - Function: SMF manages user sessions within the 5G network, including establishing, modifying, and terminating sessions. It also monitors traffic routing and session status.
  - Data Contribution: Provides data on session characteristics, traffic patterns, and user behaviour during network usage. This information is critical for understanding the nature of user interactions with the network and identifying unusual or potentially unsafe patterns.
- Access and Mobility Management Function (AMF):
  - Function: AMF is responsible for handling user access and mobility, including user authentication, mobility management, and access control.
  - Data Contribution: Provides data on user mobility (e.g., handovers, network access events) and authentication status. Mobility patterns and access behaviour contribute to assessing a user's trustworthiness and the potential risk they pose to the network.
- Policy Control Function (PCF):
  - Function: PCF makes policy decisions for the network, controlling things like Quality of Service (QoS), network slicing, and access restrictions.
  - Data Contribution: Provides information on the policies applied to users based on their behaviour and network context. This helps assess how user actions align with network policies and can indicate potential violations or abnormal behaviour.
- User Plane Function (UPF):
  - Function: UPF is responsible for forwarding user data traffic and performing packet inspection for QoE metrics.
  - Data Contribution: Provides performance metrics related to data throughput, latency, and packet loss, which are essential for determining the quality of the user's network experience. Any disruptions or anomalies in these metrics can trigger safety or trust assessments.

4. aerOS (Management and Orchestration):
   - Function: aerOS is responsible for managing and orchestrating network resources, including virtualized network functions and services. It monitors network performance, availability, and utilization.
   - Data Contribution: Provides data on network health, resource allocation, and usage patterns. This data helps evaluate overall network stability, which can affect user experience and trust in the network.

The data from these network functions flows through the system as follows:

- **Data Collection**: Each network function generates logs, performance data that are captured by DataOps.
- **Data Processing:** DataOps pre-processes and transforms this data into a format suitable for ML analysis. The data is cleaned, normalized, and structured to ensure it is consistent and usable.
- **ML Analysis:** The processed data is fed into the MLOps pipeline, where machine learning models are trained to generate insights about user behaviour, trustworthiness, and safety.
- **Decision Making:** Based on the ML analysis, the network makes informed decisions, such as granting or restricting network access, applying different levels of service, or triggering additional monitoring.

By integrating these components, the architecture ensures that real-time decisions about user safety and trust are based on comprehensive, data-driven insights. Furthermore, the continuous feedback

loop between MLOps, DataOps, and the network functions allows the system to adapt to changing user behaviour and network conditions over time.

5.3.3 TRUST FUNCTION LAYER

The Safety TF layer consists of the three functional blocks named AI Agent, vApp layer and NetApp layer. Firstly, the local AI Agent of the Safety Function is the decision maker and first touchpoint. It will decide based on information send by CC, which vApp to deploy (Figure 11).

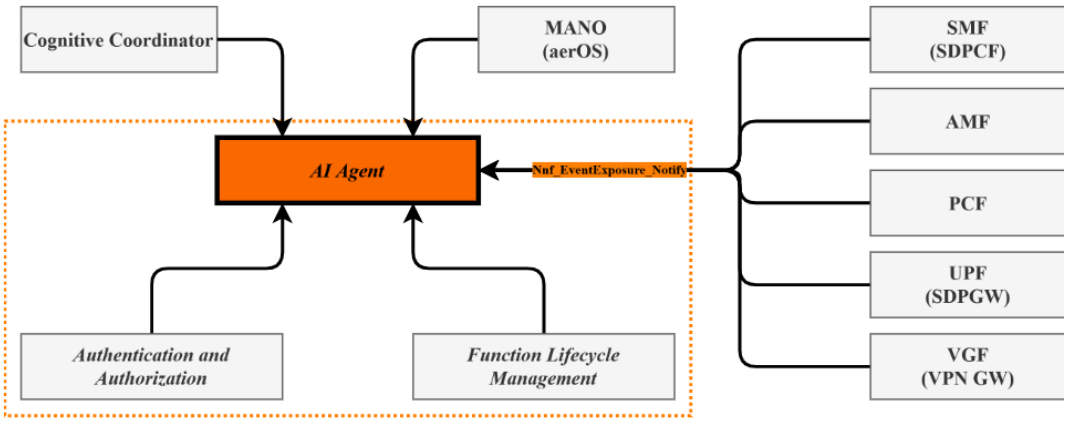

Figure 11: Safety Function - Local AI Agent interactions with the system components.

The vApp layer is responsible for communicating with other network elements to start the process of the Safety Function (Figure 12). It will deploy specific functions and carry out the actions for the creation of the Software-Defined Perimeter It is comprised of:

- SDPCF: It acts as an eSMF (enhanced SMF) for safer communications requests.
- SDPGW: Acts as a UPF and a safe gateway for target Applications.
- VGF: Acts as a VPN gateway that configures the tunnel between users and the requested end applications.

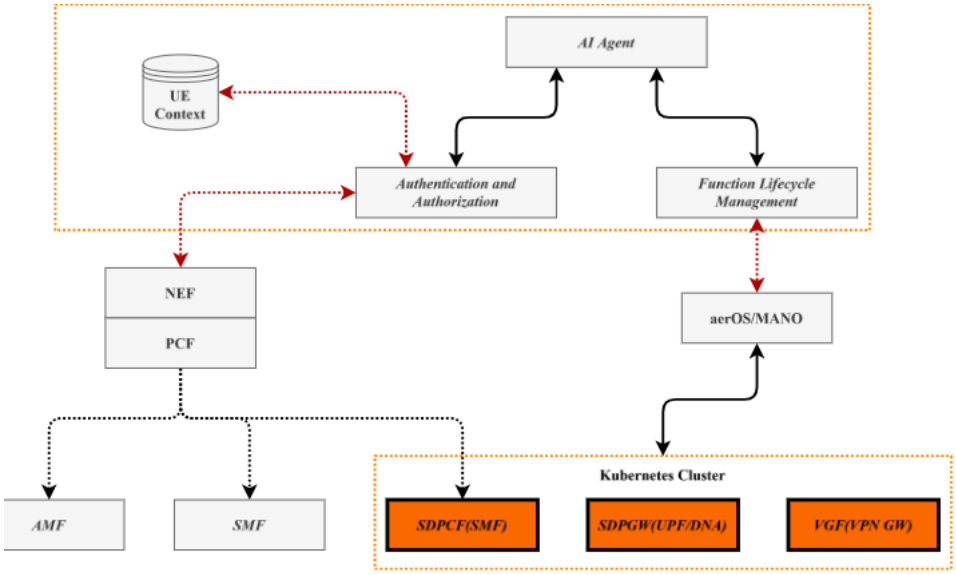

Figure 12: Safety Function - vApp Layer Overview

Lastly, the NetApp layer consists of the Network Apps that will assist the vApp layer and Local AI Agent with their decisions/actions. It includes various applications such as:

- **Topology Manager:** Calculates the path cost of each possible route across the continuum and generates information that help determine the overall safety level.
- **Resource Manager:** Tracks and manages system resources, comparing them against request overhead to support the AI agent's decision-making.

5.4 Design of the Secure SDP Stack

To support adaptive, intent-driven service access in next-generation networks, the user-centric safety function integrates an SDP stack purpose-built for dynamic trust enforcement across the 5G/6G core. This stack enables fine-grained, per-user micro-perimeters that encapsulate only the necessary service functions, reducing exposure and enhancing control.

The process is initiated when a UE, issues a safety request to CC for a target application. This request is routed to the safety function through CC and the AI layer of the function decides the next steps. The deployment of a safety level is initiated with the authentication of the UE and the associated user. The safety trust function engages the Authentication and Authorization Manager, which performs mutual verification of both the device and user credentials. Upon successful authentication, it returns a curated list of target applications and associated access policies, ensuring that only contextually relevant services are made available. Based on this input, the safety function coordinates with the Function Lifecycle Manager, which instructs the network orchestrator (aerOS) to instantiate the necessary dynamic functions (SDPCF, SDPGW, VGF). Once the environment is prepared, the SDPCF establishes communication with the SDPGW (SDP Gateway), which functions as a secure UPF and traffic moderator. The SDPGW ensures that data flows between the UE and the target applications remain confined within the authorized perimeter. At the same time, the VGF (VPN Gateway Function) is engaged to configure a secure tunnel between the UE and the gateway, ensuring that all communication is encrypted and isolated from external traffic. All subsequent traffic is routed through this secure tunnel, terminating at the VGF, which acts as the sole access path to the microservices.

The Main Controller of the core network remains entirely hidden from external exposure; only the SDPCF is momentarily reachable to facilitate initial onboarding. In case of compromise or failure, the system can seamlessly spawn a new SDPCF instance without disrupting ongoing services, thanks to the dynamic instantiation capabilities managed by the Function Lifecycle Manager.

This architecture enables real-time, policy-driven access with minimal external visibility, ensuring that each user's communication path is contextually defined, dynamically provisioned, and resilient by design.

6 User-Centric Security Function

The user-centric security function introduces a modular, distributed architecture that dynamically enforces security needs based on real-time user context, system performance metrics, and trust-level evaluation. The security function integrates with a local AI agent, SSI infrastructure, and blockchain-backed verifiable data registries to enable fine-grained, adaptive trust decision-making at the network edge. It operates in coordination with the Cognitive Coordinator to deploy appropriate security services (vApps) that match user-defined trust requirements, contributing directly to the SAFE-6G framework's goal of enabling personalized, trustworthy service delivery across the edge-cloud continuum.

6.1 Architectural Components of the Function

A high-level system view of the user-centric security function's components is illustrated in Figure 13, presenting the primary architectural components and their communication flows across the SAFE-6G infrastructure. These components span user-facing interfaces (e.g., XR/VR, mobile, chatbot), decision-making logic, and backend services, and include the function's Local AI Agent, vApps/nApps, and the backend trust infrastructure built on Hyperledger Fabric [12] and an SSI stack. The latter comprises the Verifiable Data Registry (VDR) and Security Audits smart contract, which together ensure data integrity and accountability in trust-related decisions.

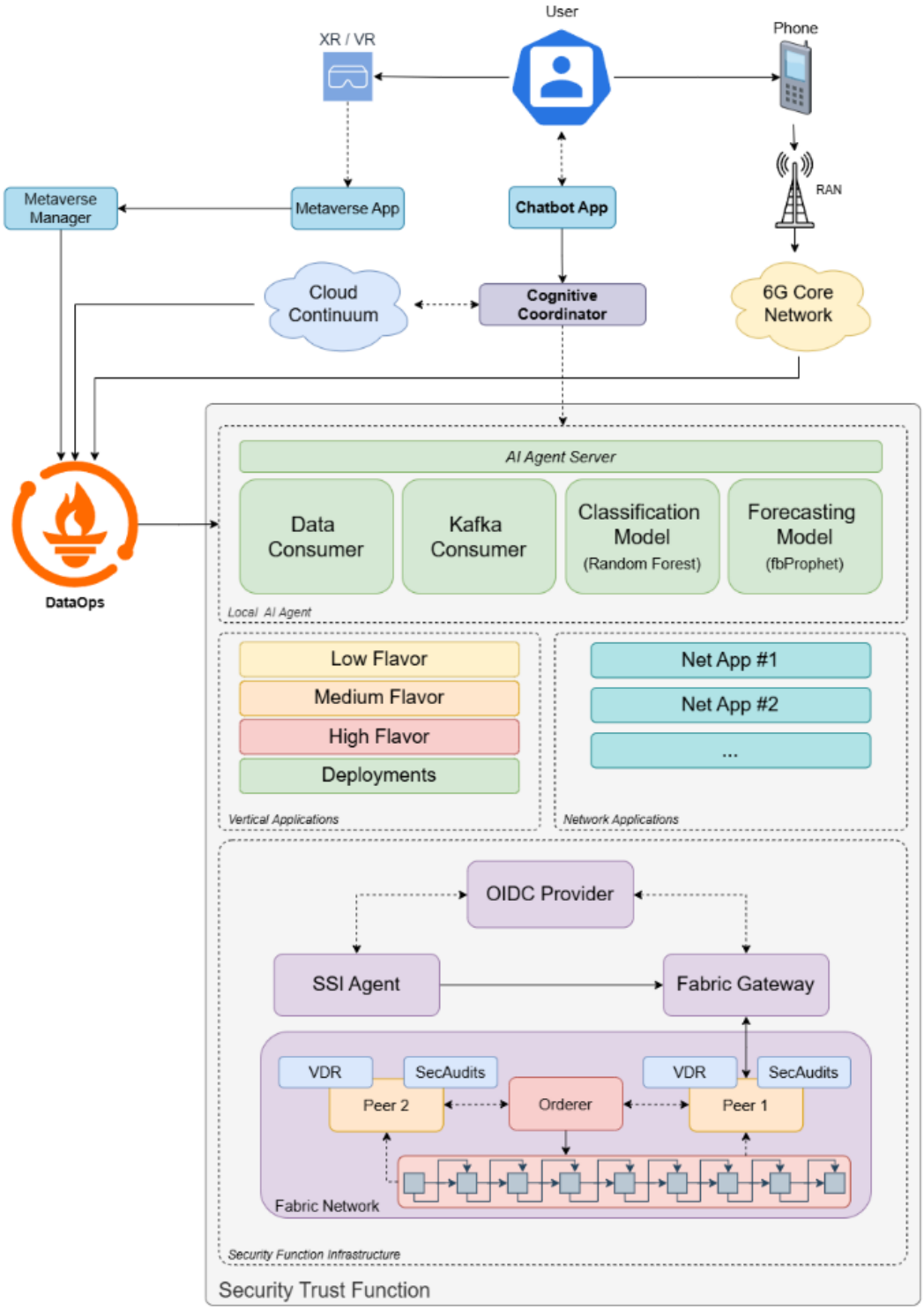

Figure 13: Security Function - Architectural Components.

The system flow begins with user interaction through multiple devices such as XR/VR gear and mobile phones, which connect the user to the Metaverse Application and Chatbot Application that serve as interfaces for capturing user trust intent. User input, particularly from the Chatbot App, is forwarded to the CoCo, which analyses the contextual information and maps the user's qualitative requirements to measurable trust scores. These trust scores are published to Kafka messaging topics, which are consumed by the User-Centric Security Function's Local AI Agent for further decision-making.

6.1.1 Local AI Agent

At the edge, the function's Local AI Agent operates as an intelligent, reactive component responsible for interpreting trust messages and real-time system telemetry to make localized security enforcement decisions. It comprises the following subcomponents:

- **Data Consumer**: Collects real-time infrastructure metrics such as CPU usage, memory consumption, and load averages using Prometheus.
- **Kafka Consumer**: Kafka client [17] subscribed to the CoCo's trust-score broadcasts (default topic: "trust-score").
- **Action Classifier**: A Random Forest-based model that predicts the required security enforcement level (e.g., no deployment or low/medium/high flavor vApp activation) based on current system status and user trust context. The Random Forest model can retrain and update via MLOps workflows based on evolving trust-score patterns and runtime system observations.
- **Decision Maker**: Consumes the outputs of the classifier and forecaster to determine the appropriate enforcement action. It also produces feedback signals to the Cognitive Coordinator, including updates to the TF cLoTw coefficient.
- **Demand Forecaster**: A forecasting model based on the Prophet time-series forecasting library [11] , this model estimates future security demand patterns (e.g., increased trust requirements during peak hours), enabling the system to pre-emptively scale and prepare for upcoming enforcement actions. The model is updated through an MLOps pipeline and enriched with contextual data streams provided by the DataOps layer.
- **Deployment Orchestrator**: Based on the decision outcome, this component triggers the corresponding vApp flavor deployment (e.g., low, medium, high), interfacing with the orchestration layer (e.g., Kubernetes) to instantiate the appropriate SSI-based service.

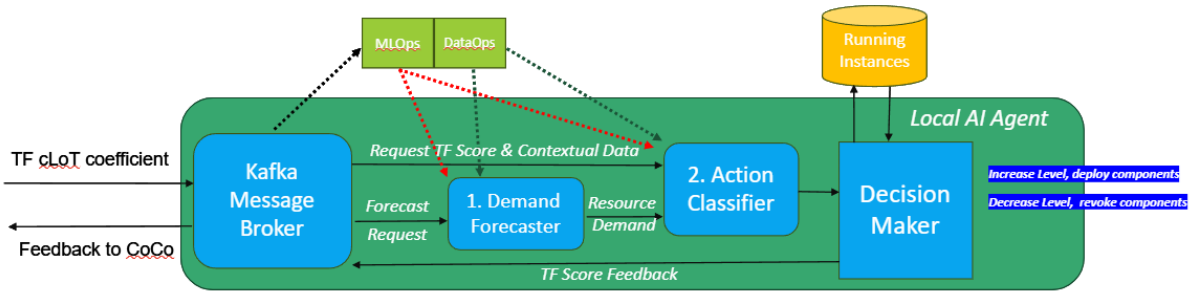

Figure 14: Security Function - Local AI Agent Workflow.

As illustrated in Figure 14 above, the Local AI Agent receives trust-score messages from the CoCo via Kafka and system metrics from Prometheus. The Demand Forecaster predicts future security demands based on historical workload patterns, while the Action Classifier evaluates the current system state and trust context to determine the required enforcement level. These outputs are consumed by the

Decision Maker, which determines the appropriate action and instructs the Deployment Orchestrator to activate or revoke the corresponding vertical application flavor. The Decision Maker also generates feedback for the CoCo, including updates to the TF cLoTw coefficient, enabling dynamic trust recalibration based on observed behavior.

6.1.2 Vertical & Network Applications

Depending on the AI agent's output, appropriate actions are triggered in either the Vertical Applications or the Network Applications layer:

- **Vertical Applications**: Include structured deployment "flavours" — Low, Medium, and High — that represent granular security responses or service adaptations.
- **Network Applications**: Encompass on-demand utility modules (e.g., monitoring scripts, context collectors) that can be triggered independently of vertical apps, extending their network capabilities.

This separation supports scalable security orchestration and enables dynamic adaptation to shifting network resources or user intent.

6.1.3 Function Infrastructure

The Security TF includes a foundational layer of digital identity and integrity for all system interactions. The entire trust function is designed to be extensible, secure, and verifiable — ensuring compliance with evolving privacy and regulatory standards. This digital identity layer is composed of the following subcomponents:

- **OIDC Provider**: This server enables OAuth2 secure authentication and access control via both "*client_credentials*", "*auth_code*" and "*preauth_code*" flows. This service is necessary for issuing access tokens that regulate interaction between the user and protected resources.
- **SSI Agent**: The SSI Agent manages decentralised identifiers and verifiable credentials using a custom DID method. It interfaces with the OIDC Provider to request Access Token for interactions with Fabric Gateway.
- **Fabric Gateway**: A REST-accessible proxy that acts as a secure bridge between off-chain components and the blockchain peers. It validates access requests using OIDC tokens or API secrets.
- **Fabric Network**: A permissioned blockchain implemented using Hyperledger Fabric. Peers host the chaincode (smart contract) logic and state, while the orderer manages transaction consensus and block generation.

6.2 Functionalities per Layer (Local AI Agent, vApps, nApps)

The user-centric security function follows a layered architecture that separates intelligence, execution, and utility roles across three operational layers: the Local AI Agent, the Vertical Applications (vApps), and the Network Applications (nApps). This design promotes modularity, clear separation of concerns, and allows each layer to be updated, scaled, or replaced independently without affecting the overall system integrity.

The interaction between the three layers is tightly coupled through a feedback-control loop. The CoCo provides a high-level trust score via the message broker. The Local AI Agent interprets these messages alongside system telemetry to make security decisions. Based on the output, corresponding vApps are deployed to enforce those decisions, while nApps continuously supply monitoring data to inform future actions.

This structure enables both anticipatory and reactive security responses, driven by live telemetry and trust-policy coordination — all while maintaining modular separation between intelligence, execution, and support functionalities.

6.2.1 Local AI Agent

The Local AI Agent is the cognitive engine of the security function, responsible for transforming raw telemetry and trust signals into actionable security decisions. Deployed as a Python-based FastAPI service, the agent performs real-time reasoning to determine the most appropriate response to changing system conditions and user-driven trust intents. Its core functionalities include:

1. **Trust-Score Consumption**: Subscribes to Kafka messages published by the CoCo and parses incoming trust-score payloads.
2. **System Monitoring**: Retrieves runtime system metrics via Prometheus from the DataOps layer, including current CPU usage, RAM consumption, available memory and system load.
3. **Security Classification:** Applies a Random Forest-based classifier to assess whether a security response should be triggered. The current model outputs a binary decision (deploy or not deploy), with a plan to expand to enumerable outputs (e.g., levels 1-6) for more nuanced response mapping.
4. **Security Forecasting:** A forecasting model based on Prophet [13] estimates future security demand over a defined time horizon, enabling proactive resource scaling and pre-emptive enforcement.
5. **Threshold Logic:** Internally maps classification outcomes and demand forecasts into predefined flavours of deployment (e.g., low, medium, high), each reflecting a security trust level or response intensity.

6.2.2 Vertical Applications

vApps are the contextual execution components triggered directly by the AI agent's decision-making process, designed to be modular and composable, allowing vertical-specific security logic to be deployed flexibly across use cases. They encapsulate response logic that adapts system behaviour based on the trust level and operational context (e.g., smart manufacturing, metaverse environments). Each vApp corresponds to a specific security posture or operational mode:

- **Low Flavour**: Passive monitoring or soft policy enforcement with minimal system intervention.
- **Medium Flavour**: Elevated monitoring, partial access control changes, or intermediate access rules.
- **High Flavour**: Full enforcement, including access revocation, active defense mechanisms and enforced credential validation.

6.2.3 Network Applications

Network Applications, or nApps, provide auxiliary or support functionality, often focused on monitoring, telemetry collection, or network-level actuation. While the current implementation embeds some nApp behaviour inside the Local AI Agent, future iterations will externalize these into distributed agents. Some of the functionalities planned are listed below:

- **Data Collectors**: Scripts or lightweight agents that collect external or local system state and transmit it to the Local AI Agent.
- **Edge Monitors**: Placeholder for future agents that may track QoS metrics, mobility patterns, or anomalous flows.
- **Action Hooks**: nApps may also serve as programmatic endpoints that respond to AI-driven decisions (e.g., isolate a node, apply a firewall rule).

6.3 Architecture for Data Security

The architecture of the user-centric security function incorporates privacy and data protection mechanisms across all functional layers. Designed with privacy-by-design principles, the system enforces data minimization, selective disclosure, and secure data exchange in line with regulatory frameworks such as GDPR, eIDAS, and NIS2. By combining decentralized identity, local AI processing, and fine-grained access control, the architecture supports secure, privacy-respecting operation in dynamic 6G environments.

At the core of runtime decision-making, the local AI agent performs inference on system telemetry, including CPU usage, RAM availability, and load statistics, all collected locally via Prometheus. The AI agent does not store any personally identifiable information, focusing only on aggregated or operational metrics. Future versions of the models are planned to incorporate differential privacy during offline training phases, ensuring that even indirect leakage of sensitive user information is mitigated.

Access to sensitive services and protected components is controlled through an OIDC provider, which issues access tokens based on OAuth2-compliant flows. Backend services such as the Fabric Gateway and the SSI Agent authenticate using "*client_credentials*" grants, while interactive users leverage the "*authorization_code*" flow. Tokens are scoped to enforce least-privilege principles, with resource-based access logic governed by token claims. Token storage is short-lived and ephemeral, further reducing exposure.

For identity management, the system adopts an SSI model, centered around the Veramo framework and a custom DID method. Through this approach, users and agents operate under decentralized identifiers (DIDs) that avoid the need for centralized identity providers. Verifiable Credentials (VCs) are issued and exchanged selectively enabling a user, for example, to prove only their role or authorization without disclosing unrelated identity attributes. This credential layer is also integrated into the OIDC flow, allowing for seamless authentication based on SSI-backed identity without persistent user tracking or centralized credential resolution.

Privacy is further reinforced by secure, permissioned on-chain data management. The system utilizes Hyperledger Fabric to store both identity-related records and future audit logs. The VDR (Verifiable Data Registry) chaincode maintains DID documents compliant with the W3C DID specification, while

the SecAudits (Security Audits) chaincode stores security-triggered events, or trust score changes and is planned to track system actions such as credential issuance. All blockchain entries are immutably logged, cryptographically signed, and access-controlled via X.509 certificates.

Overall, the system emphasizes data minimization, edge-based processing, and user sovereignty. Data collected by the Local AI Agent is transient and never linked to identifiable users. All identity interactions are mediated through verifiable and revocable credentials. By design, the architecture limits exposure, centralization, and long-term data retention, ensuring privacy and data security remain first-class concerns throughout the security function lifecycle.

6.4 Blockchain, Chaincodes, and Tokenization

The user-focused security feature uses a permissioned blockchain system built on Hyperledger Fabric to provide unchangeable audit logs, reliable identity management, and safe handling of credentials throughout their lifecycle. The blockchain layer is tightly integrated with the AI agent, SSI stack, and access control logic, forming a decentralized trust backbone for SAFE-6G. Two specialized chaincodes have been developed to address the core blockchain functions:

- **Verifiable Data Registry** (VDR) chaincode serves as the on-chain repository for DID documents. It implements the core CRUD operations necessary to store, resolve, and manage custom DID method entries, ensuring consistent identity resolution across services. Each DID document includes associated public keys, authentication methods, and service endpoints, compliant with the W3C DID v1.0 specification.
- **Security Audits** (SecAudits) chaincode acts as a ledger for key security events such as credential issuance, vApp deployment triggers, trust score changes, or user access requests. Each entry is designed to be timestamped, signed, and optionally include cryptographic hashes of the verifiable credentials involved—enabling zero-trust auditing without exposing sensitive data.

Chaincode events are emitted on critical transactions (e.g., DID registration, credential exchange), and these can be intercepted by the Fabric Gateway, which acts as policy-aware middleware between off-chain systems and the blockchain network. This event infrastructure supports future extensions such as real-time alerting, trust synchronization between edge agents, or external compliance reporting hooks.

Tokenization within the architecture focuses primarily on access control and event traceability. Tokens issued by the OIDC provider may be used to represent:

- Access rights (via OAuth2 scopes)
- Trust classifications (e.g., encoded into JWT claims to match classification outcomes)
- Delegated authority (to authorize one component to act on behalf of another)

While these tokens are symbolic in nature—not tied to economic value—they provide a machine-verifiable layer of control that complements the immutable state tracked in the blockchain. Over time, these tokens may also be cross-referenced in on-chain audit entries, allowing cryptographic traceability of security-relevant actions across users, agents, and coordinating components.

Together, the audit and tokenization infrastructure enable a decentralized, tamper-evident, and privacy-preserving model for security assurance in complex, multi-stakeholder 6G environments. The modular chaincode design allows additional registries or attestations to be layered in as new use cases emerge (e.g., policy revocation logs, device trust registries), providing a foundation for future extensibility and integration.

6.5 Self-Sovereign Identity (SSI) Implementation

6.5.1 SSI Context

Self-Sovereign Identity (SSI) is an innovative approach to identity management that empowers individuals with greater control over their personal data. As the demand for more secure and private identity solutions grows, various decentralized identity platforms, SSI wallets, and SSI protocols have emerged. These platforms enable users to create, manage, and verify their digital identities using distributed ledger technology and decentralized mechanisms. SSI wallets serve as secure repositories for storing credentials and managing identities, while SSI protocols establish the technical standards necessary to implement SSI solutions, facilitating trusted digital identities for individuals, organizations, and even devices.

This section of the document explains the contributions to the Self-Sovereign Identity layer of the Security Function, particularly its implications for the 6G layer. SSI is designed to allow users to safely store, and exchange digital credential known as VCs, allowing them to authenticate themselves with relying parties to gain access to specific resources with the minimum required data or justify pieces of their identity to another entity.

The goal of this element of the User Centric Security function is to provide users with a tool to manage their identity by giving them access to a Self-Sovereign Identity Agent (SSI-Agent), which will act as their personal VC and identifier storage. The user will then be able to perform Attribute based Authentication Control (ABAC) by creating Verifiable Presentation of their credential upon request and sending them to a relying party which could be for example the authentication portal of a factory.

In the SSI ecosystem, three primary roles are present: issuer, holder, and verifier:

- The issuer issues credentials about a given entity, referred to as the subject.
- The holder, on the other hand, stores and manages these issued claims in their digital wallet, akin to storing ID cards in a physical wallet.
- The verifier, often called the relying party, can be considered as a service provider that requires verification for their services.

Together, these roles establish what is commonly called the SSI Trust Triangle. With this foundational understanding of the roles within the SSI ecosystem, we can now delve into the case studies that illustrate the practical applications and implications of Self-Sovereign Identity in real-world scenarios like SAFE-6G project.

6.5.2 SSI STUDY CASE

Before implementing SSI within SAFE-6G project, the first step is to understand what SSI is and how it should be implemented within the project.

With insight based on prior projects and internal research on how to implement SSI to perform issuance and verification of attributes, multiple components were developed and reviewed to refine the global understanding of the paradigm.

Key technologies were identified to create a solution in adequacy with latest digital identity regulation, including:

- Adherence to the standards for DIDs and VCs.
- Compliance with the standard protocols governing SSI like W3C and Decentralized Identity Foundation.
- Conformance to VC formats as established by W3C and IETF.

6.5.3 SSI TRUST LAYER

DIDs are a foundational component of SSI, providing a new way for individuals and entities to establish and verify their identities without relying on a central authority. DIDs are globally unique identifiers that enable secure and private interactions between parties. They can be created, managed, and resolved through various decentralized mechanisms. There are created by the users (holder of credential, issuer or relying party) and are based on a keypair, Figure 15 describes the composition of a DID.

Figure 15: Security Function - W3C DID Structure (https://www.w3.org/TR/did-1.0/).

While the private key is kept by the user, the public part can be retrieved using the DID Method-Specific Identifier and is either computed from the identifier or used as a way to retrieve the public key on a registry. All the mechanism related to DID operation are listed in a method specification document, which act as the base of the interoperability between actors within the same ecosystem.

To create common ground for the communication of all actors using the SSI Implementation, a DID SAFE-6G specification has been produced and is available on SAFE-6G project repository. The document allows implementors to know how to create the identifier, which key type to use and how to produce the object that stored the public crypto material (DID Document).

The implementation of a specific method for SAFE-6G was needed instead of using an existing one to allow the project to have sovereignty over the network used and how the registry would be created to better fulfil the SAFE-6G project requirements. Table 6 Security Function - DID State of the Art describes some common DID methods that are currently used within the SSI ecosystem, whether they use an external registry or not.

DID Method	*Registry*	*Type of registry*	*Generation of the identifier*
did:web	Web Server	Web Server	Particle is the address of your web server minus http particle (for google it would be did:web:google.com).
did:key	Ledgerless	No registry	Multibase base58-btc encoded value that is a concatenation of the Multicodec identifier for the public key type and the raw bytes associated with the public key format.
did:cheqd	Cosmos	Public Blockchain	base58 encoding of the first 16 bytes of the SHA256 of the first Verification Method Ed25519 public key OR UUID.
did:ethr	Ethereum	Public Blockchain	Ethereum address or a compressed secp256k1 publicKey.

Table 6 Security Function - DID State of the Art

The SAFE-6G DID Registry is set to be hosted on a Hyperledger Fabric Blockchain, a blockchain also used within the security function to host the audit chaincode. The document describes the two workflows to interact with the registry: DID Creation and DID Resolution.

Figure 16 shows the workflow to create an identifier. This workflow can be implemented by every SSI Entity (Issuer, Holder, Relying Party) that has access and is trusted by the vAapp Gateway, a REST API developed to interact with the SSI DID Registry. In SAFE-6G project, Issuer and Relying Party entities are external components provided by a partner.

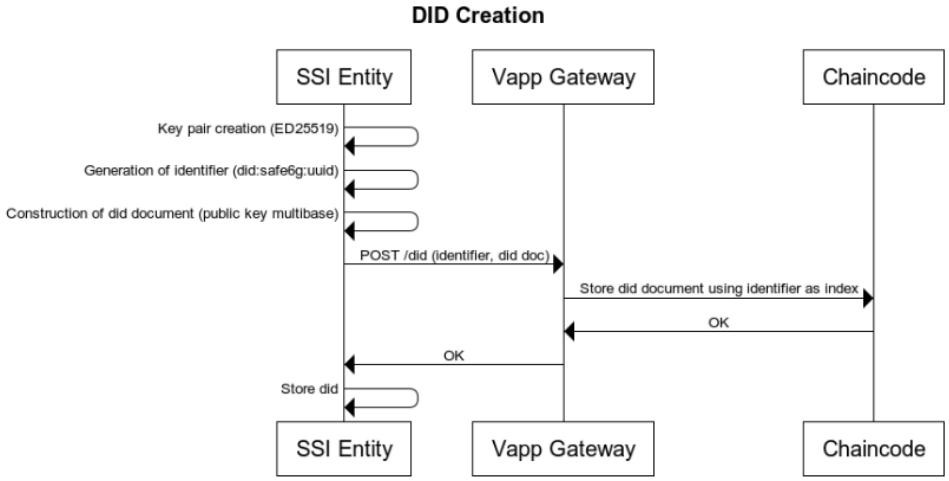

Figure 16: Security Function -DID Creation Workflow.

Figure 17 describes the DID Resolution, a process mostly used in authentication processes that allows an entity to retrieve the public crypto material associated with an identifier.

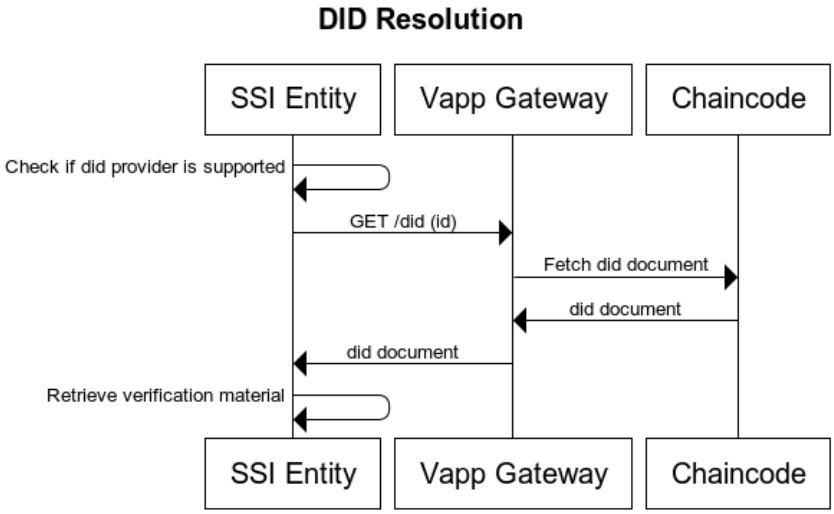

Figure 17: Security Function -DID Resolution Workflow.

To help implementors understand and use the SSI Implementation, a developer guide will be provided in the future, which will describe everything needed to execute an issuance and verification workflow.

6.5.4 SSI Implementation

Within the SAFE-6G project, one of the main aspects of the security function is to showcase how the 6G layer could integrate decentralized identity to authenticate users and grant them permission using Verifiable Credential and DID.

To do so, an implementation has been made of a DID Provider which implements the DID Specification to allow Issuers to create identifier based on SAFE-6G Trust layer and use them to sign credentials. The credential is received by an identity wallet using standard protocols. Currently, the wallet is an open-source solution, but it could be swapped to use SAFE-6G SSI-Agent latter in the project.

The Provider is also used during the authentication of a credential holder to retrieve the crypto material used to sign the VCs they are holding.

Since the provider is used for both issuing and verification workflow, current implementation allows a full demonstration of a traditional SSI Exchange based on SAFE-6G Trust layer.

6.6 Legal and Regulatory Compliance

The SSI stack aims to follow the latest versions of protocols and specifications defined by the European Digital Identity Wallet Architecture Reference Framework (ARF), ensuring alignment with European interoperability guidelines. The following highlights some of our accomplishments in accordance with established standards, recommendations, and regulations:

- **Verifiable Credential Issuance**: OID4VCI (OpenID for Verifiable Credential Issuance) has been adopted as a standard for credential issuance.
- **Verifiable Credential Verification**: Our stack supports OID4VP and SIOPv2 protocols (OpenID for Verifiable Presentation and Self-Issued OpenID Provider), ensuring standards-based verification workflows aligned with EU Digital Identity Wallet initiatives.
- **Verifiable Presentation**: the Decentralized Identity Foundation Presentation Exchange v2.0 specification is adopted, supporting flexible credential presentation across formats.
- **Verifiable Credential Lifecycle Management**: Our implementation incorporates both the W3C Verifiable Credential Status List 2021 and the Bitstring Status List to enable revocation and status checking throughout the credential lifecycle.
- **Verifiable Credential Formats**: We support multiple credential formats, including IETF SD-JWT VC, W3C Verifiable Credential Data Model v1.1, and ISO/IEC 18013-5 mDL/mDoc, in alignment with eIDAS v2-recognized formats and cross-border wallet pilot requirements.
- **Trust Models**: A comparative analysis of three trust models, OpenID Federation, EBSI, and TRAIN has been conducted, resulting in a concrete understanding of their trade-offs in terms of security, interoperability, and implementation practicality. This analysis has informed the trust policy design and credential validation logic of the SAFE-6G architecture.

7 User-Centric Privacy Function

7.1 Architectural Components of the Function

Below is a block diagram of the architecture of the Privacy TF.

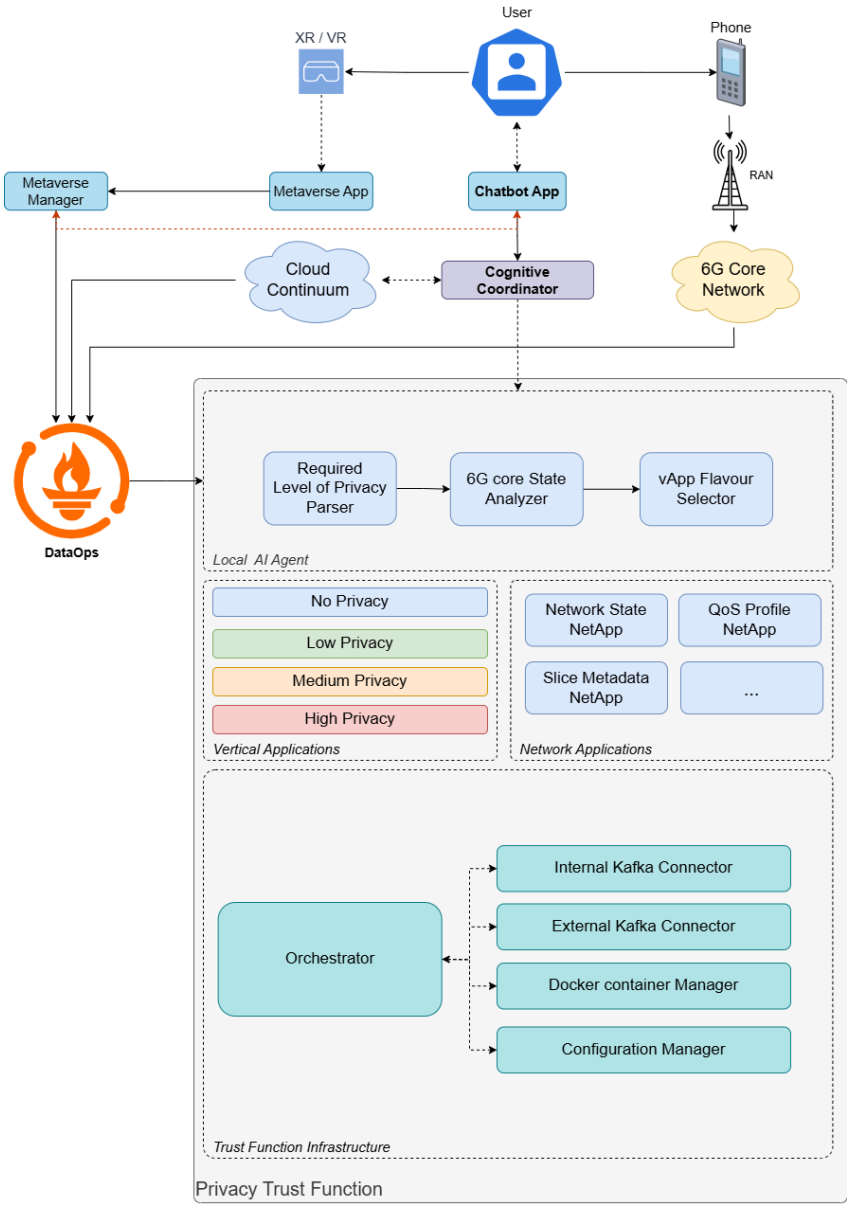

Figure 18: Privacy Function - Architectural Components.

As illustrated in Figure 18, several components (described in the next section) must coordinate to execute the Function's workflow. To ensure robustness and maintainability, an orchestrator service governs the process by managing inputs, invoking the appropriate services at each step, and aggregating the final output. Each auxiliary service encapsulates a specific functionality, designed for efficiency and a degree of deterministic behaviour. A detailed breakdown of these components is provided below.

7.1.1.1 ORCHESTRATOR

The orchestrator serves as the central coordinator of the system. It manages the end-to-end execution by receiving input, determining the required workflow, invoking the appropriate services, and aggregating the result. It ensures that each component is triggered in the correct sequence and monitors their execution to maintain system stability.

7.1.1.2 AI AGENT

The AI agent supports decision-making by considering the required privacy level along with the context of the session. It helps choose the most suitable vApp to run, making sure the system applies the right actions to protect user privacy.

7.1.1.3 VERTICAL APPLICATIONS

Each vertical application (vApp) encapsulates a predefined set of actions corresponding to a specific privacy policy (Low, Medium, High), allowing the system to dynamically adapt its behaviour to the required privacy level. Upon invocation, the vApp triggers the appropriate nApps, oversees the execution of their actions, and returns the outcome to the orchestrator.

7.1.1.4 NETWORK APPLICATIONS

Network applications (nApps) are responsible for performing actions directly on the 5G core network. These actions may include retrieving network information or applying specific privacy-enhancing measures, such as traffic shaping or access control. Each nApp is designed to be atomic, meaning it focuses on a single, well-defined function and is reusable across different vApps.

7.1.1.5 MESSAGE BROKER CONNECTOR

The system includes two primary message brokers: one for internal communication between the components of the Privacy Function, and another for external communication with the Cognitive Coordinator. The connector manages the production and consumption of messages, ensuring reliable data exchange for all required operations.

7.1.1.6 CONTAINER MANAGER

For deployment, all vApps and nApps are packaged as Docker container images. The container manager is responsible for handling these images and provides an interface that allows other components to launch, manage their execution, and consume their results.

7.1.2 FLOW

The orchestrator component is responsible for the communication of the Privacy Function with external systems and the coordination of internal services. When a new trigger is received from the CoCo (along with relevant information), the following process is executed:

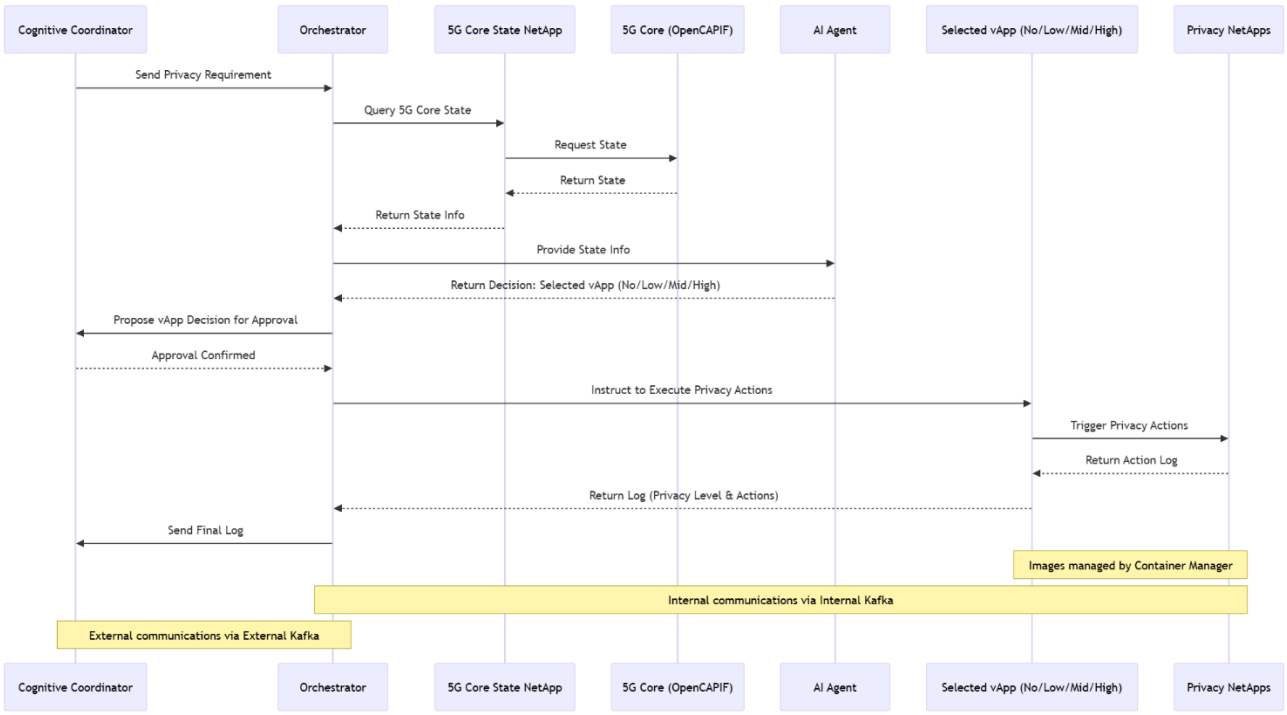

Figure 19: Privacy Function - Component Workflow.

All the vApps and nApps (which are containerized applications) are managed by the Container Manager. Communication between modules is handled using **Kafka**, with a clear separation between **external interfaces** (via *External Kafka*) and **internal orchestration logic** (via *Internal Kafka*).

The workflow consists of the following steps:

- **Privacy Requirement Initiation:** The CoCo initiates the workflow by sending a numeric privacy requirement (0-100) to the Orchestrator. This requirement typically reflects user needs that need to be enforced during the session.
- **5G Context Acquisition:** The Orchestrator needs contextual network information to decide on the appropriate privacy strategy. It issues a query to the 5G Core State NetApp, which is a specialized nApp for retrieving network state. The 5G Core State NetApp then sends a request to the 5G Core (via OpenCAPIF) to access the most up-to-date network state (e.g., user mobility, service usage, edge resource status).
- **State Propagation:** The 5G Core responds with the requested data. This state information is propagated back through the 5G Core State NetApp to the Orchestrator, which parses and structures it into a format consumable by internal agents.
- **AI-Based Privacy Assessment:** The Orchestrator forwards the contextual state to the AI Agent. This agent uses predefined policies and logic to assess the privacy requirements and determine the necessary response. Based on this evaluation, it selects the most appropriate vApp flavor (Non, Low, Mid, or High), each representing an increasing degree of privacy enforcement mechanisms.

- **Decision Communication:** The AI Agent returns the decision to the Orchestrator, specifying which vApp to activate.
- **Approval Workflow:** Before execution, the Orchestrator compiles the vApp plan and sends it to the CoCo for approval. This step ensures policy compliance and allows for higher-level decision logic to intervene if necessary.
- **Approval Confirmation:** The CoCo evaluates the plan and sends back approval confirmation. If approval is withheld or delayed, the process pauses, avoiding actions with conflicting consequences with other Functions.
- **Action Plan Execution:** Once approved, the Orchestrator instructs the selected vApp to launch and begin executing its privacy actions.
- **Triggering Privacy Mechanisms:** The selected vApp invokes specific Privacy nApps, which implement the actual technical mechanisms. These nApps are modular and reusable across different privacy contexts.
- **Execution Logging:** The vApp aggregates the logs and returns them to the Orchestrator, which validates and formats the report. It sends the log to the CoCo, closing the loop.

7.2 Functionalities per Layer (Local AI Agent, vApps, nApps)

7.2.1 Local AI Agent

The Local AI Agent operates as the decision-making core within the Privacy Function. It is responsible for analysing contextual data, such as user preferences, session metadata, and current network state, to determine the most appropriate privacy policy to apply. This decision is informed by a module configured to balance utility with privacy protection. Its primary functionalities include:

- **Input analysis**: Ingests data received from the CoCo and the 5G core (via the core state nApp).
- **Policy evaluation**: Determines the appropriate privacy policy level which corresponds to the vApp that will be deployed (No, Low, Medium, High). The way this decision is made is through a combination of rules and symbolic AI. More approaches will be tested on the second part of the project.
- **vApp selection**: Maps the selected policy level to a corresponding vertical application. For example, a Medium Privacy Policy level means the Medium vApp will be deployed.

7.2.2 vApps

A vertical application (vApp) is a containerized orchestration unit designed to implement a specific set of privacy-preserving actions. Each vApp corresponds to a predefined privacy level - None, Low, Medium, or High - and encapsulates the control logic required to enforce it. Upon activation, the selected vApp is responsible for invoking the appropriate nApps, configuring them according to the selected privacy policy, monitoring their execution, and collecting logs to report back to the Orchestrator. This modular design allows the Privacy Function to dynamically adapt to varying contextual requirements (e.g., user sensitivity, network state, service risk) while preserving a transparent and auditable flow of actions.

The High Privacy vApp adopts a maximum isolation strategy, aiming to minimize data exposure by placing users in highly controlled environments. It ensures strict separation from public or shared

network resources and configures the network infrastructure to eliminate contention and interference. This strategy is well-suited for sensitive use cases where confidentiality, integrity, and exclusivity are paramount.

The Medium Privacy vApp applies a resource-isolated strategy, maintaining the use of shared infrastructure while ensuring dedicated throughput and limited access. It configures virtualized resources in a way that balances performance with privacy, reducing the likelihood of data leakage or service interference while maintaining efficient resource utilization. This approach is ideal for users who require elevated but not absolute privacy protection.

The Low Privacy vApp follows a best-effort privacy strategy, leveraging standard public network resources while applying minimal constraints. It prioritizes scalability and performance over strict isolation, offering a lightweight privacy enforcement layer that supports high user density and general-purpose services. This level is appropriate for everyday applications where data sensitivity is low and operational efficiency is critical.

The None Privacy vApp does not make any configuration changes, since it is assumed that all vApps operate within a containerized execution environment, allowing seamless deployment, scaling, and lifecycle management under the supervision of the Container Manager. The decision of which vApp to activate is made dynamically by the AI Agent, based on contextual input (e.g., user policy, network state) and subject to approval by the Cognitive Coordinator. This model supports transparent, auditable, and adaptive privacy enforcement that can evolve with emerging user and network needs.

7.2.3 nApps

An nApp is a self-contained, stateless component that performs a specific, predefined action on the network. Each nApp is designed to be self-contained, meaning it encapsulates a single function and operates independently of other components.

For instance, a **Max Throughput nApp** executes the following logic upon activation:

- Loads the required configuration parameters (e.g., the target maximum throughput value in Mbps).
- Issues an HTTP request to the designated API endpoint to apply the configuration.
- If the request is successful, it returns a success message and terminates gracefully.
- If unsuccessful, it initiates a retry mechanism, looping back to the HTTP request step.

Each action defined within a vApp is delegated to exactly one nApp. Consequently, a single vApp may invoke multiple nApps, depending on the privacy policy it enforces. Based on the current design scope, the system is expected to include up to 12 unique nApps, corresponding to the complete set of actions required across all privacy levels. This modular approach ensures scalability, ease of maintenance, and functional clarity.

7.3 Privacy Score and Assessment System

The Local AI Agent acts as the decision-making core of the Privacy Function. Its main role is to select one of the predefined vApps based on a combination of context-aware inputs. Each vApp corresponds to a fixed privacy policy (None, Low, Medium, or High) and encapsulates the concrete actions required

to enforce it. The agent itself does not perform any network operations or enforce policy; it simply determines which vApp should be activated.

The agent ingests two main categories of input:

- **Privacy Score (0–100):** A numerical score received from the CoCo, designed to reflect the privacy sensitivity of the current session. This score is based on multiple upstream factors, including user privacy preferences, service classification (e.g., messaging vs. video streaming), and sensitivity of the data being processed. The Privacy Score provides the primary signal for determining the required level of privacy enforcement.
- **Network State Snapshot:** A set of selected 5G core network parameters that represent current operating conditions. These include:
  - Slice profile metadata, such as resource sharing level and isolation guarantees.
  - User-specific QoS parameters, such as PreemptionCapability and PreemptionVulnerability.
  - Real-time infrastructure metrics, such as CPU and interface utilisation.
  - Session-level attributes (e.g., number of active UEs, DNN usage, and throughput limits).

These inputs are evaluated against a symbolic rule base that encodes the mapping from input conditions to privacy policy selection. This logic is designed to be fully transparent and auditable, ensuring that each decision path can be reconstructed from the rules triggered and the conditions they matched.

The agent also applies internal safety checks to delay or prevent execution when network capacity is critically constrained.

All rules are written as declarative clauses to maintain consistency and traceability. This is particularly advantageous in the early deployment phase, where statistical models are not yet feasible due to limited telemetry. Future iterations will incorporate statistical calibration based on real Privacy Score distributions and system load patterns, with the potential to introduce learned components under strict guardrails.

This architecture ensures that privacy enforcement decisions are reliable, explainable, and aligned with both user requirements and infrastructure limitations.

7.4 Decision Support and Mitigation Suggestions

The Privacy Function defines a set of predefined actions that can be executed on the 5G Core to enhance user privacy. These actions are applied through the nApps and are grouped based on their expected privacy impact. This categorization allows the Local AI Agent to reason about trade-offs and recommend an appropriate level of intervention.

Privacy-enhancing actions are divided into three categories:

- **High Impact**: These actions make substantial changes to session handling, slice isolation, or routing. They offer the strongest privacy guarantees but may significantly impact service performance or availability.

- **Medium Impact**: These actions moderately increase privacy by altering configuration parameters that reduce user traceability or exposure.
- **Low Impact**: These actions make minor adjustments to improve privacy with minimal effects on performance.

A detailed description of the available actions is provided below. Each action is atomic and managed by one nApp.

7.4.1 High Impact Actions

1. **Adjust 5GQoS Preemption Settings:** Updates the preemptCap and preemptVuln parameters within the 5G QoS profile to reduce the risk of session preemption. By disabling the ability to preempt and increasing vulnerability to being preempted, this configuration ensures that user sessions are less likely to be forcibly terminated or redirected. This protects user data during moments of network congestion or reallocation.
2. **Reduce RAN Slice Resource-Sharing Level:** Sets the resourceSharingLevel to its minimum value, enforcing stricter resource isolation in the RAN. This prevents co-mingling of traffic between different slices, which in turn reduces the risk of side-channel attacks and cross-slice traffic inference.
3. **Limit Maximum Number of UEs per Slice:** Applies a hard cap on the number of UEs allowed in a given slice by updating the maxNumberofUEs parameter. Smaller UE groups lower the statistical likelihood of inter-device correlation, making it more difficult for adversaries to track or profile users based on aggregated slice behaviour. Since this action does not have a good privacy/performance ratio, it will only be available in cases where extremely high privacy is required.
4. **Change the Data Network Name:** Alters the data network name field in the service profile to route traffic through a dedicated and controlled private network. This ensures that data traverses paths with elevated access controls and reduced exposure, enhancing traffic confidentiality. Since this action does not have a good privacy/performance ratio, it will only be available in cases where extremely high privacy is required.

7.4.2 Medium Impact Actions

1. **Constrain PDU Session Count per Slice:** Limits the number of concurrent PDU sessions allowed within a slice subnet. By reducing the session volume per user or per group, this configuration minimizes the surface area available for correlation across multiple flows or applications.
2. **Switch Subscriber's Group for Shared AMBR Settings:** Moves a subscriber to an alternative QoS group, such as group2, which may have more restrictive AMBR (Aggregate Maximum Bit Rate) settings. This enables the enforcement of different traffic profiles and limits bandwidth observability by reducing variance between users.

7.4.3 Low Impact Actions

1. **Modify Guaranteed Uplink Throughput:** Updates uplink throughput values in the service profile to reduce guarantees and cap maximum throughput. This makes traffic patterns less predictable and deters adversaries from performing fingerprinting based on flow size or frequency.

2. **Restrict Session Aggregate Maximum Bit Rate:** Caps the session AMBR parameters for both uplink and downlink directions. Lowering bandwidth ceilings limits the ability to distinguish user behaviour based on traffic intensity or burstiness.
3. **Limit expDataRateDL in eMBB Requirements:** Reduces the expected downlink data rate (expDataRateDL) in the enhanced Mobile Broadband (eMBB) slice profile. This diminishes the formation of distinct, high-throughput profiles and lowers exposure to profiling in high-traffic use cases.
4. **Increase priorityLevel in 5G QoS Profile:** Adjusts the priorityLevel field to raise the priority of user sessions. This reduces the likelihood of service interruption or redirection, mitigating risks associated with forced teardown, rerouting, or transient exposure.

All actions are designed to be parameter-driven and reversible. The decision to activate a specific action is taken by the Local AI Agent, in alignment with the selected vApp. Before enforcement, the full plan is shared with the CoCo for approval.

This design enables dynamic, policy-driven privacy mitigation within the 5G core network, without requiring constant manual oversight.

8 User-Centric Resilience Function

8.1 Architectural Components of the Function

The Resilience Function (RF) is designed to deliver intent-aware, adaptive resilience as part of the SAFE-6G trust framework. Its architecture is modular and distributed, enabling it to operate effectively across heterogeneous 6G environments while responding dynamically to user-centric intents. The RF architecture is structured into two main functional layers:

- The AI Layer, which hosts the decision-making intelligence, and
- The nApp/vApp Layer, which executes resilience actions by orchestrating network and application resources.

These layers interact through standardized APIs and a messaging framework to maintain seamless communication with the CoCo and other SAFE-6G components, including DataOps, MLOps, OpenCAPIF, and aerOS.

At a high level, the AI Layer ingests user intents, telemetry, and contextual information, analyses them using machine learning models and policy rules, and determines the most appropriate resilience strategy (or flavour) to apply. The nApp/vApp Layer then implements the selected strategy by applying configuration changes, reallocating resources, and ensuring service continuity across the edge–cloud continuum. The figure below illustrates the High-Level Design and data flow between components.

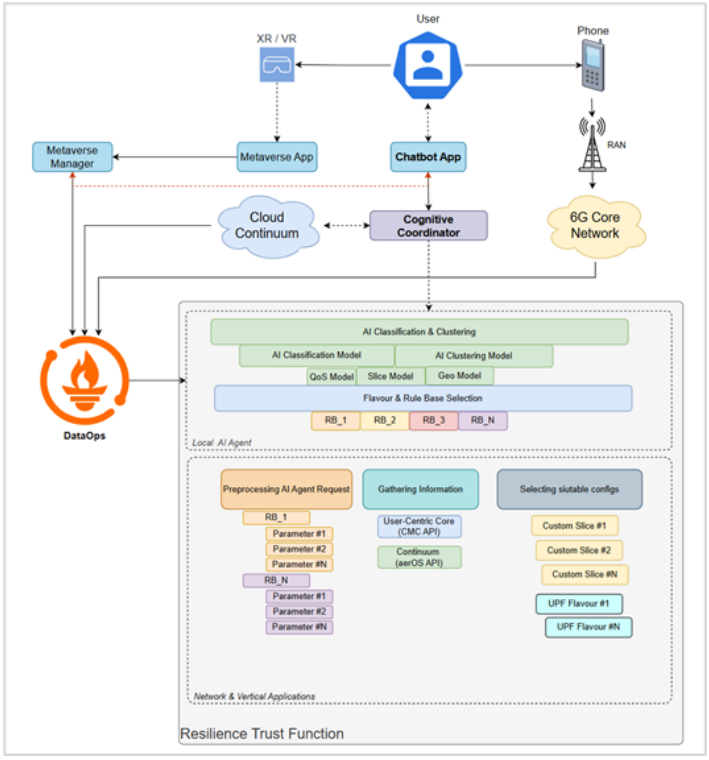

Figure 20: Resilience Function - High-Level Designng of Resilience Function.

8.2 Functionalities per Layer (Local AI Agent, nApp, and vApp Layer)

The RF has two main layers – the Local AI Agent and the nApp/vApp Layer – each provides distinct but complementary functionalities.

8.2.1 Local AI Agent (AI Agent)

The Local AI Agent is the intelligent core of the RF, implemented as a containerized microservice within the User Service Node (USN). It is responsible for analyzing user intents and operational context to determine and recommend appropriate resilience actions, aligning system behavior with the specified trustworthiness objectives.

The AI Agent receives LoTw requests from the CoCo via an asynchronous messaging framework, such as Kafka. These requests include user or session metadata along with the desired trust objectives, which serve as the primary input to decision-making process for the RF.

Once a request is received, the AI Agent collects real-time telemetry and contextual metrics from the infrastructure through DataOps and other monitoring components. This data provides visibility into the current state of the network, including performance indicators, risk signals, and QoS metrics.

The collected data is then preprocessed and normalized to generate session-level context indicators, such as risk profiles and QoS classifications. This preprocessing step ensures that the data is clean, consistent, and suitable for use by the downstream decision-making modules.

The AI Agent applies trained machine learning models (e.g., classification and clustering algorithms) along with rule-based logic to assess the session state and select the most appropriate resilience flavor. These flavors represent predefined groups of policies that encapsulate specific configurations and actions tailored to operational conditions.

Finally, the selected flavor is translated into a structured, actionable configuration plan that can be executed by the nApp and vApp layers. The AI Agent also sends feedback to the CoCo on the outcome of the enforcement and updates its internal models through integration with the MLOps platform whenever retraining or recalibration is needed.

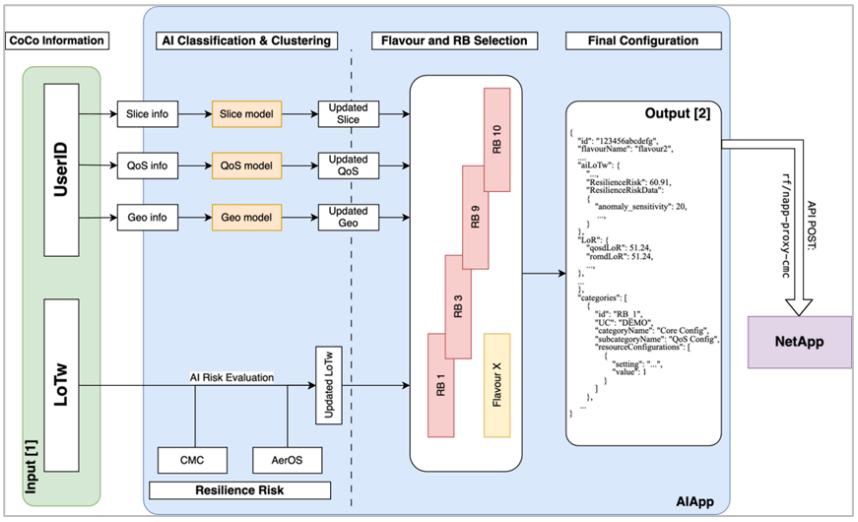

Figure 21: Resilience Function - AI Resilience Layer.

The AI Agent integrates a range of machine learning models and is designed to work seamlessly with the SAFE-6G MLOps platform for model lifecycle management. The models employed include supervised classifiers (e.g., XGBoost, KNN) and sequence models (e.g., LSTM) trained to predict risk profiles, classify sessions, and recommend resilience flavors based on contextual metrics.

Training datasets are generated by collecting historical telemetry and contextual data from the SAFE-6G infrastructure, including QoS metrics, slice configurations, and observed resilience outcomes under different scenarios. These datasets are preprocessed and augmented to improve model robustness and generalization.

The ML pipeline follows modern MLOps principles, with continuous monitoring of inference performance, automated retraining when performance drifts, and controlled deployment of new model versions. The AI Agent connects to the MLOps platform to retrieve the latest validated models, which are deployed in the containerized inference environment (e.g., KFServing). This pipeline ensures that the decision-making logic remains adaptive to evolving network conditions and user requirements while maintaining operational stability.

8.2.2 nApp and vApp Layer

The nApp & vApp Layer operates in direct coordination with the AI Resilience Layer, serving as the executor of the AI Agent's decisions. Once a structured action plan is generated by the AI Layer, it is delivered to the nApp & vApp components for operational enforcement within the SAFE-6G infrastructure.

Figure 22 illustrates the internal workflow of the nApp and vApp Layer, highlighting its integration points, processing sequence, and interaction with the AI Resilience Layer and infrastructure services.

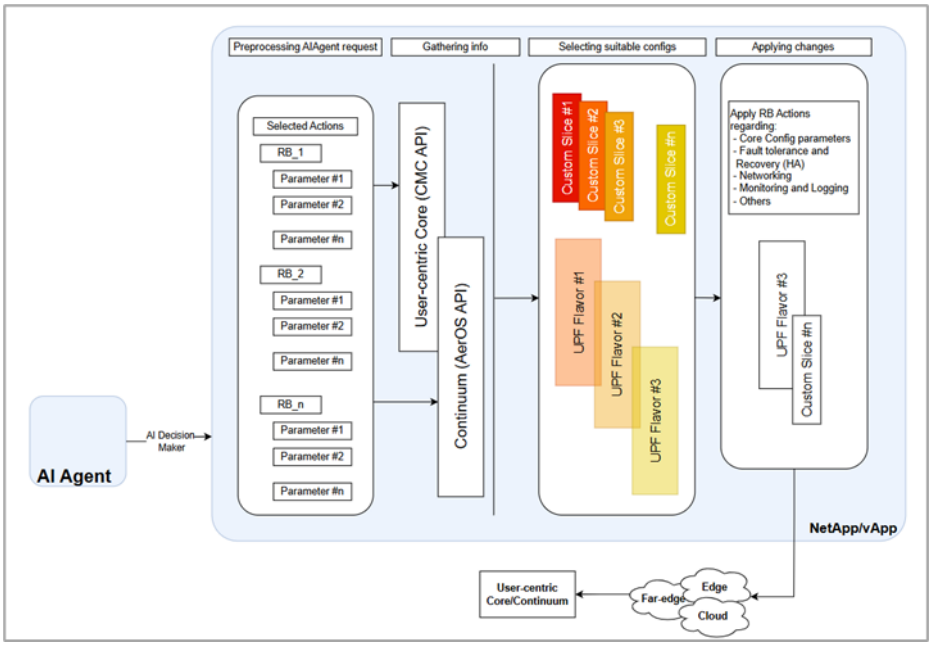

Figure 22: Resilience Function - nApp & vApp Resilience Layer.

From the figure above. The process begins with parsing the resilience flavor-based configuration received from the AI Resilience Agent.

A resilience flavor represents a predefined group of resilience policies (Section 8.3) suited to specific operating conditions. The AI agent selects the most suitable flavor and maps it to a concrete set of network actions. These may include creating or modifying network slices, rerouting traffic through a different UPF, or adjusting QoS parameters such as 5QI and AMBR.

With the resilience flavor selected, the vApp layer parses the incoming policy and contextual inputs, prepares parameterized deployment templates that match user-specific requirements, and incorporates dynamic variables such as service types, slice parameters, QoS levels, and energy awareness.

Once prepared, the nApp proceeds to interface with orchestration and control platforms (e.g., aerOS, CNC, OpenCAPIF) to enforce the defined actions. After execution, the system tracks the outcome and operational status of each action and results in being consumed via API to the rest of SAFE-6G components.

8.3 Resilience Policy Engine

The Resilience Policy Engine is a core component of the Resilience Function, responsible for applying rule-based decision logic that covers how resilience flavor is executed across the 6G network and continuum infrastructure. It bridges the AI Agent's intent-driven decisions with executable configurations by leveraging predefined rules, the Policy Engine ensures that resilience strategies are consistently aligned with the current LoTw, contextual conditions, user intent, and AI Agent classifications. The Policy Engine operates using two main categories of rules:

8.3.1 Core Configuration

This category focuses on rules that govern the configuration of core network functions and user-specific parameters. Actions in this category include:

- Allocating or modifying network slices (e.g., medium/high slice selection).
- Adjusting QoS parameters such as 5QI, GBR, and AMBR.
- Switching subscription profiles or UE group memberships.
- Redirecting user traffic to optimal UPF instances.
- Prioritizing traffic flows based on trust and intent.

These rules ensure that the network is reconfigured dynamically to maintain resilience and trust while optimizing resource utilization.

8.3.2 Fault Tolerance and Recovery Actions

This category covers resilience mechanisms that respond to service degradation, failures, or predictive anomalies. Rules in this category enable the system to:

- Instantiate redundant core functions (e.g., AMF/SMF replication).
- Trigger failover or re-establishment procedures.
- Migrate services across edge nodes to restore trust levels.
- Initiate self-healing operations based on observed KPIs.
- Log events for explainability and incident tracing.

The detailed rule definitions that drive the Policy engine information are not included in this section due to the scope of this document.

8.4 Integration and Interfaces

The RF is designed to operate as an integrated component of the SAFE-6G trust framework, interacting seamlessly with both internal modules and external network infrastructure. Its modular design and use of standardized APIs and messaging frameworks enable interoperability across heterogeneous domains, from the 6G Core Network to the edge–cloud continuum.

This section outlines the main integration points and interfaces of the RF with other components in the SAFE-6G ecosystem.

8.4.1 Interfaces with SAFE-6G Components

Beyond CoCo, the RF interacts with several other components in the SAFE-6G framework through well-defined interfaces:

Core Network APIs
The RF enforces resilience actions at the control and data plans of the 6G Core Network. It connects to APIs such as:

- **OpenCAPIF**: Serves as the secure, policy-compliant exposure layer to discover and invoke core network services and analytics.
- **Core Network Control (CNC)**: Used for applying network-level configurations like QoS adjustments, slice reconfiguration, and resource allocation.
- **NWDAF & NEF**: It is used to retrieve network analytics and enforce edge policies as required.

Edge–Cloud Continuum APIs
The RF also interfaces with orchestration layers managing the distributed infrastructure:

- **aerOS**: Provides orchestration, monitoring, and scaling of edge/cloud resources.
- **OpenCAPIF**: Used for secure access to orchestration and telemetry services at the continuum level.

Other SAFE-6G Modules
- **DataOps**: Supplies real-time telemetry and contextual metrics for decision-making.
- **MLOps**: Manages the lifecycle of ML models used by the AI Agent, including retraining and deployment.
- **Monitoring Framework:** Provides operational situational awareness and feeds back performance metrics.

The next table summarizes the integration of the RF with the rest of the SAFE-6G components:

Component	*Interface Type*	*Purpose*
Cognitive Coordinator	Messaging Broker (Kafka)	Receive intents and send feedback

OpenCAPIF	REST API	Control plan and analytics access
CNC/NWDAF	REST API	Core Configuration and monitoring
aerOS	REST API	Edge-cloud orchestration
DataOps	REST API	Telemetry and metrics ingestion
MLOps	REST API	ML Model management

Table 7 Resilience Function - Integrations

The current design and integration of the Resilience Function establish the foundation for intent-driven, user-centric resilience within the SAFE-6G framework. Further implementation details, performance evaluations, and extended functionality will be developed and documented in subsequent deliverables dedicated to each trust function.

9 User-Centric Reliability Function

9.1 User-Centric Reliability In 6G

9.1.1 State-of-the-Art in 6G Reliability

The 6G network architecture is inherently distributed, leveraging technologies such as Multi-Access Edge Computing (MEC) and Edge AI to meet the demands of low latency, high reliability, and availability for critical services. MEC reduces latency by processing data closer to end users, enabling real-time applications like gaming and vehicular communications [17] -[18] . Edge AI further enhances 6G by bringing computational power to the network edge, improving performance through real-time data analysis and predictive maintenance. This approach minimizes service disruptions by anticipating potential issues, thereby increasing network reliability [17] , [19] .

However, the distributed nature of Edge AI introduces challenges, such as asynchronism among servers, which requires robust synchronization mechanisms to maintain efficiency. Real-time data exchange regarding memory and processing capabilities is essential to ensure seamless operation [17] . The reliability of 6G networks, therefore, hinges on efficient coordination of computing nodes and robust communication protocols capable of handling substantial data traffic from storage and retrieval processes.

9.1.2 Defining Reliability in 6G Networks

Reliability in 6G networks can be defined as the probability of a system functioning correctly over a specified period under given conditions. This is critical in distributed 6G architecture, where reliability depends on the effective coordination of computing nodes and dependable network infrastructure. According to[17] , robust communication protocols are vital to manage the high volume of data traffic, ensuring consistent performance. The Ultra-Reliable Low Latency Communications (URLLC) capabilities in 6G enhance current 5G standards by achieving success probabilities of transmitting packet of different size within the maximum allowed latency at a certain channel quality ranging from $(1 - 10^{-5})$ to $(1 - 10^{-7})$, supporting mission-critical applications [20] -[21] . AI-powered predictive maintenance further bolsters reliability by enabling real-time communication between machines, anticipating failures, and optimizing performance in industrial settings [19] .

9.1.3 User-Centric Reliability in SAFE-6G

The SAFE-6G framework adopts a user-centric approach, prioritizing QoE by aligning network performance with user needs. QoE encompasses user satisfaction, trust, and seamless interaction with applications, which is critical for real-time data processing in applications like industrial automation and autonomous vehicles [22] . This approach for reliability is different than those proposed in [20] -[21] , which measure reliability in terms of packet loss rate, while in SAFE-6G we want to focus on the users' perspective and their unique needs/intents by examining their subjective QoE-related metrics. By focusing on user requirements, SAFE-6G can enhance latency reduction, resource allocation, and overall network dependability, fostering trust in automation and reinforcing security, privacy, and resilience [1].

At the application layer, SAFE-6G ensures reliability by maintaining sufficient resource availability for Virtual Network Functions (VNFs), which must remain operational throughout the entire service

lifetime [23] . Intelligent scaling mechanisms balance performance and energy efficiency, anticipating traffic demands while mitigating server failures without excessive resource activation. This approach creates a robust and efficient communication environment that users can rely on confidently [23] .

9.2 Architectural Components of the Function

9.2.1 Reliability Function Architecture

Reliability TF is designed to ensure that the network maintains its overall trustworthiness by dynamically managing and continuously optimizing the reliability of the service based on the provided LoTw. It acts dynamically and based on the user's intent, leverages AI-driven mechanisms to adapt to changes in the network environment and infrastructural capabilities. It manages reliability throughout the entire lifecycle of network services, from deployment to decommissioning. This lifecycle management ensures that the desired LoTw is maintained as the network evolves and changes.

Reliability TF consists of three main components. The AI Agent, the vApp, and the NetworkApp. These components have been developed in Python and have been containerized for deployment within a micro-service architecture over the heterogeneous and distributed edge-cloud continuum. The AI Agent is responsible for selecting and deploying the appropriate vApp based on the LoTw value received from the CoCo via an event streaming platform. It supports three LoTw ranges. The "Low" LoTw, which is between 1 and 40%, the "Medium" LoTw, which is between 41 and 80%, and the "High" LoTw, which is between 81 and 100%. The vApp component, also developed in Python and containerized for scalability and modularity, focuses on the inference of different AI/ML models with separate instances for each network service. Notably, its models are stored within container images using, e.g..joblib and .h5 file extensions, ensuring modularity and ease of deployment. The vApp reads input monitoring data from the NetworkApp through a message broker and returns predictions to the AI Agent using the same messaging system. Meanwhile, the NetworkApp, similarly developed and containerized, is designed to interface with various data sources, such as Prometheus and OpenCAPIF, with one instance per component. A high-level overview of the reliability TF is illustrated in the following figure. In addition, a video developed for the purposes of SAFE-6G project and that presents the operation of the Reliability TF, can be found at this link.

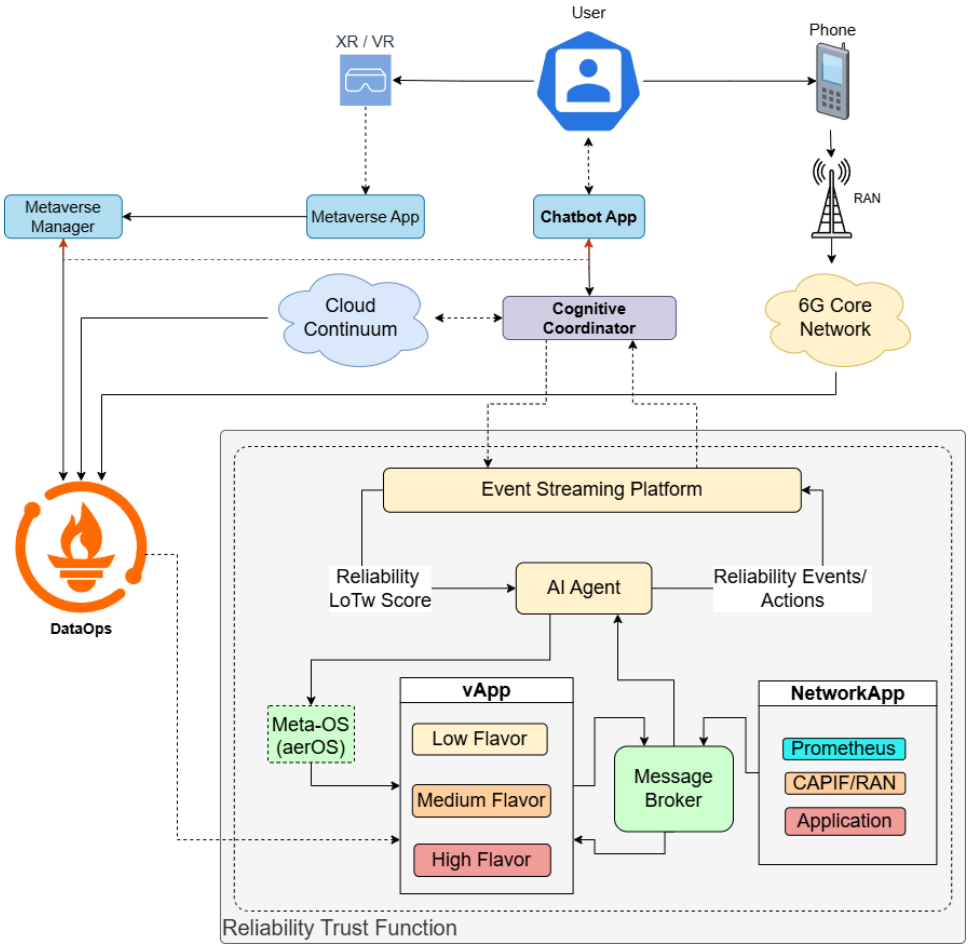

Figure 23: Reliability Function - Architectural Components.

9.2.2 REQUIREMENTS

The Reliability TF requirements have been documented in the SAFE-6G Deliverable D2.1 and are briefly presented here for the sake of completeness. At the core is the requirement for seamless communication with the cognitive SAFE-6G coordination component (REQ-FREL-NET-F-M-01), which ensures that reliability events detected by the function can be transmitted in real time for system-wide, intent-driven orchestration. Given the complexity of tasks and the amount of data, the provision of HPC infrastructure for ML (REQ-FREL-RES-NF-O-02) is critical. This requirement reflects the necessity to expedite both training and inference operations, minimizing training and inference time for real-time reliability estimations and making it feasible to handle large-scale, feature-rich datasets drawn from the different planes of the 6G network. Relatedly, the ability to select different ML models based on varying levels of trustworthiness (REQ-FREL-AI-NF-M-03) introduces the required intelligence adapted to the user intents.

Next, the interaction with the MLOps (REQ-FREL-NET-NF-R-04) for post-deployment evaluation ensures that deployed AI models are rigorously assessed for performance, accuracy, and speed, thus embedding continuous improvement and accountability into the operation of the reliability TF. Security is fundamentally addressed by requiring that reliability operations take place within secure, isolated environments restricted from public network access (REQ-FREL-NET-F-R-05), protecting both the AI pipeline and sensitive monitoring data from exposure or compromise. Another requirement for

interoperability (REQ-FREL-RES-NF-M-06) demands that Reliability TF can seamlessly consume streaming monitoring data concerning infrastructure and service status, reflecting the need for real-time data-driven adaptation. The training and inference of reliability AI models (REQ-FREL-AI-NF-M-07) is another key requirement, mandating that the function predicts both service-breaking points and security threats, for proactively detecting anomalous and/or malicious behaviors.

Next, Reliability TF shares common requirements with the other TFs within SAFE-6G system. The most notable requirements include the access to performance data across the entire cloud continuum (REQ-COM-DES-F-M-13), the 5G/6G core (REQ-COM-DES-F-M-14), and from the user application itself (REQ-COM-DES-F-M-15) to ensure that reliability models are trained and operationalized on various layers capturing the full context of 6G network operation. The function's scalability (REQ-COM-DES-F-M-16) and fault tolerance (REQ-COM-DES-F-R-17) are required to ensure that reliability mechanisms are robust under dynamic scaling and resilient to failures, essential for mission-critical and high-availability services. Furthermore, requiring a standardized, robust interface with the MLOps framework (REQ-COM-DES-F-R-19) reinforces both the intelligence and maintainability of the system. Near real-time monitoring through a publish-subscribe approach (REQ-COM-DES-F-R-20) is crucial for supporting rapid analysis and mitigation actions, while the requirement REQ-FREL-DES-NF-R-21 mandates the user applications to support management actions—scaling, migration, or otherwise—to uphold service continuity in direct response to reliability related events.

Further, REQ-COM-RES-NF-M-23 emphasizes the need for reliable data persistence, ensuring that sufficient storage is available for data, algorithms, and trained models in distributed deployments, a necessity for operational AI systems. REQ-COM-RES-F-M-25 targets the development of energy-efficient AI models to address the considerable resource consumption posed by large-scale, distributed AI, thereby promoting sustainability in 6G. Finally, REQ-COM-RES-F-M-27 guarantees adequate data availability for training, which is vital for achieving accurate and adaptable AI models capable of meeting SAFE-6G's trustworthiness objectives.

9.2.3 Operation

Reliability TF is designed to uphold overall system and network trustworthiness by dynamically managing and continuously optimizing service reliability in response to the provided LoTw. The operation follows a structured sequence to ensure reliable and adaptive performance, as illustrated in the figure below. The main steps are as follows:

1. The AI Agent receives the LoTw score from the CoCo through an event streaming platform.
2. Then, it maps the desired LoTw score received by the CoCo to the appropriate trustworthiness level (low, medium, high).
3. Based on the LoTw level, the AI Agent selects the appropriate AI/ML model and identifies the required NetworkApp(s).
4. The AI Agent deploys the selected vApp and nApp(s) via the Meta-OS orchestrator.
5. The nApp(s) retrieves input data from data components (e.g., CAPIF/RAN, Prometheus, Application).
6. The vApp processes the input data from the nApp(s) and generates predictions.
7. The vApp sends the predictions to the AI Agent.
8. Based on the predictions, the AI Agent determines the appropriate actions or outputs across the whole 6G edge-cloud continuum to enhance service reliability.
9. The AI Agent transmits the related reliability actions or outputs to the CoCo for further processing.

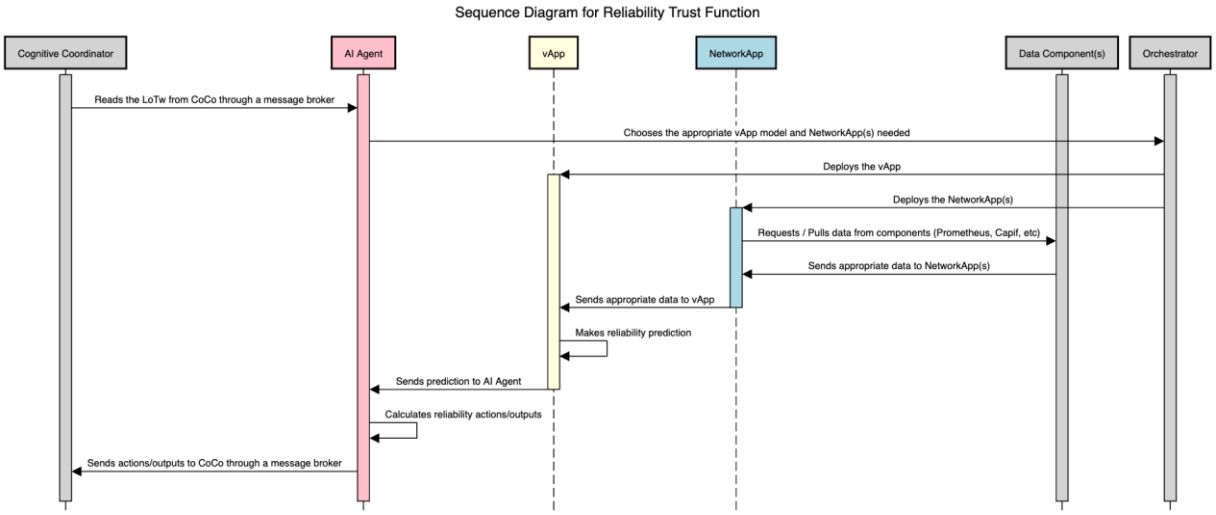

Figure 24: Reliability Function - Architectural Component Workflow.

9.3 Functionalities per Layer (Local AI Agent, vApps, nApps)

9.3.1 Local AI Agent

The local AI agent acts as the decision-making core of Reliability TF, responsible for AI/ML model selection and orchestration. It interacts with the cognitive SAFE-6G coordination component through an event streaming platform, enabling the real-time transmission of detected reliability events to support system-wide, intent-based orchestration. The first important responsibility of the AI agent is to map the desired trustworthiness score received by the cognitive coordinator (e.g., 15%, 50%, 80%, etc.) to the appropriate trustworthiness level (low, medium, high). Based on this LoTw level, the system selects the most suitable vApp flavor (and the related nApps) for deployment from among the available options. The deployment of the vApp and nApp(s) is achieved through the Meta-OS orchestrator. The AI agent communicates with the vApp and receives its predictive outputs via a message broker. According to these predictions, it determines the necessary actions across the 6G edge-cloud continuum to enhance service reliability. This could involve tightening security protocols, reallocating resources, migrating services, or mitigating emerging risks. In the end, it reports and alerts back to the cognitive coordinator, by providing the final succeeded score and these actions. The AI agent, in collaboration with the vApp and the nApp, continuously monitors the state of the network and system, dynamically adjusting reliability through real-time analysis and actions to ensure the application trustworthiness.

9.3.2 Different vApp Flavors Tailored to the LoTw

The Reliability TF in 6G networks is designed to ensure dependable and adaptive performance by tailoring vApp flavors to the LoTw. The LoTw, determined by the CoCo, dictates the computational complexity, accuracy, and security features of the deployed ML models. Reliability TF adopts a stratified approach to vApp deployment based on LoTw, to ensure that 6G networks deliver user-centric reliability, balancing performance, efficiency, and security according to the requested by users trust levels.

9.3.2.1 vApp Flavors Tailored to Level of Trustworthiness

Reliability TF employs distinct vApp flavors that scale in computational complexity, accuracy, and security features based on the LoTw value, which ranges from 1 to 100%. These flavors are categorized into three tiers: Low, Medium, and High LoTw, each addressing specific user and system requirements.

Low LoTw (1–40%)

In environments with low trust (LoTw below 40%), the system prioritizes speed and resource efficiency. Lightweight ML approaches, such as k-Nearest Neighbors (k-NN) algorithms and simplified neural networks, are deployed to ensure sufficient accuracy while minimizing computational overhead. This configuration supports core operations, including basic device monitoring, data analysis, and maintaining adequate QoS. Key functionalities include:

- Fast algorithms for rapid processing.
- Device monitoring for operational oversight.
- Guaranteed QoS to ensure reliable service.
- Sufficient accuracy for basic reliability needs.

This tier is optimized for scenarios where trust is limited, focusing on fundamental reliability without advanced features.

Medium LoTw (41–80%)

At moderate trust levels (LoTw between 41% and 80%), the system transitions to more sophisticated federated learning (FL) implementations using deep neural networks. These models balance resource utilization with enhanced performance, delivering high accuracy while monitoring multiple network planes and devices. The increased computational complexity ensures seamless operation and improved QoE. Key functionalities include:

- Seamless operation across network components.
- Deep neural networks for advanced processing.
- Multiple alerts for proactive monitoring.
- High accuracy without excessive computational burden.

This tier supports applications requiring consistent performance and enhanced user experience.

High LoTw (81–100%)

In high-trust environments (LoTw exceeding 80%), the system employs advanced FL methods with robust security enhancements. These sophisticated implementations provide exceptional accuracy and can enable detection of malicious activities, such as Distributed Denial-of-Service (DDoS) attacks and intrusions. The High LoTw tier introduces advanced capabilities, including:

- Malicious actions detection for proactive threat mitigation.
- Security profiling to enhance network resilience.
- Very complex algorithms for superior performance.

This tier is designed for mission-critical applications, ensuring secure and reliable performance under stringent trust requirements.

This categorization enables 6G to deploy increasingly sophisticated ML approaches proportionate to established trust relationships: from basic service guarantees to advanced security threat mitigation.

9.3.3 nApps and proposed actions

The nApp operates in a continuous loop collecting, processing, and distributing metrics in near real-time across the entire edge-cloud continuum. It gathers data periodically from multiple heterogeneous sources, such as OpenCAPIF for service exposure framework integration, core network infrastructure components for fundamental network telemetry, Prometheus monitoring systems for comprehensive resource metrics collection, and direct application-level data sources that provide granular insights into user service performance. The gathered data undergoes intelligent filtering and preprocessing mechanisms that transform raw telemetry data into standardized formats optimized for consumption by vApps, ensuring seamless interoperability and efficient resource utilization throughout the reliability assessment pipeline. To deliver the processed data to the vApp, the nApp leverages advanced message queuing protocols, utilizing message brokers, such as RabbitMQ, to establish reliable, asynchronous data publishing channels that maintain system responsiveness while handling high-volume metric streams. This communication infrastructure ensures that Reliability TF can maintain continuous awareness of system state changes across all operational layers and planes, enabling proactive identification of potential reliability degradation patterns and facilitating rapid response mechanisms for maintaining service quality objectives.

9.4 Multi-Layered Monitoring And Data Collection Framework

The SAFE-6G multi-layered monitoring and data collection framework is designed to feed Reliability TF with data from all planes of the 6G ecosystem: the infrastructure (edge-cloud continuum), the network (core 5G/6G functions), and the application/user service plane. From the infrastructure plane, metrics can be gathered from physical devices, hypervisors, virtual machines, containers, and orchestration platforms, spanning hardware-level resources such as CPU, memory, network interfaces, and energy usage. In practice, this can be realized by deploying Kubernetes as the orchestration platform along with tools such as Cilium for networking, Kepler for energy metrics, Prometheus for resource monitoring, Grafana for visualization, and k9s for Kubernetes management. Devices like UEs can be connected to a portable network server (hosting both RAN and core functions), enabling telemetry from both physical and virtual resources. The network plane offers rich data from sources such as NWDAF and UPF, including analytics on UE mobility, network load, session management, traffic class differentiation, and slice usage. Finally, the application/service plane provides detailed workload characteristics, such as number of concurrent requests, user-application traffic, service latency, and QoS/QoE metrics, directly reflecting user experience.

An important feature of Reliability TF is its ability to interrelate these data streams across planes. For example, changes in a user's position or device capabilities (application plane) can directly affect radio access quality (network plane) and, by extension, resource consumption, and performance at the infrastructure plane. By collecting metrics like Channel Quality Indicator (CQI), Signal-to-Interference-plus-Noise Ratio (SINR), end-to-end delay, and traffic volume, and correlating them with user-centric QoE measures (e.g., frame rate in metaverse applications or real-time latency), the Reliability TF not only can observe how lower-layer phenomena propagates to the user experience but also identify early indicators of service degradation or abnormal behavior. This integrated, correlated view enables more accurate training of AI/ML models, which can then predict reliability related issues or performance-breaking points to provide timely alerts and trigger proactive mitigation actions, such as scaling, migration, or reconfiguration.

10 Implementation Roadmap and Timeline

The implementation of the SAFE-6G user-centric trust functions, Cognitive Coordinator, and supporting AI components is being executed following a modular and agile plan, allowing for parallel development and late binding of components via well-defined APIs and shared orchestration logic.

The development activities follow a phased roadmap, with inter-WP coordination points serving as synchronization anchors. These anchors allow each partner to progress independently while aligning integration and validation efforts through shared data formats, brokers, and lifecycle hooks.

10.1 Partner Roles and Responsibilities

Component	Lead Partner	Responsibility
Cognitive Coordinator	NCSRD	Regression + reasoning engines, orchestration broker, chatbot integration
Explainable AI Module	THA	Selector, executor, and interpreter for Trust Function decisions
User-centric Trust Function: *Safety*	EBOS	SDP stack, secure perimeter deployment, microservice isolation
User-centric Trust Function: *Security*	IQBT	SSI/OIDC, classification engine, blockchain VC + audit trails
User-centric Trust Function: *Privacy*	SHG	Dynamic nApp control, privacy score engine, mitigation planner
User-centric Trust Function: *Resilience*	TID	AI-driven resilience agent, deployment templates, fault-tolerant execution
User-centric Trust Function: *Reliability*	UNIWA	LoTw-based model switching, cross-layer telemetry processing

Table 8 Roadmap - Component Responsibilities Matrix

10.2 Component Development Maturity Matrix

Development is coordinated through the following principles:

- **Shared Orchestration Framework**: All Trust Functions and AI Agents interact through a message broker, enabling asynchronous command-and-control via LoTw signals.
- **Pluggable AI Models**: Trust Functions support modular ML models, allowing iterative replacement without an architectural refactor.
- **Closed-Loop Execution**: All TFs return telemetry and outcome states to the Coordinator, enabling dynamic policy revision and model retraining.

The following maturity matrix summarizes the current maturity level of each key WP4 component based on architectural readiness, implementation status, and integration progress.

- ✅ **Yes** – Implemented or near production-ready
- ⚠️ **Partial** – Under development or partially integrated
- ❌ **Not yet** – Planned but not started or conceptual only

Component	*Architecture Defined*	*Implementation Started*	*Integration in Progress*	*Early Validation/Test*
Cognitive Coordinator	✅ Yes	✅ Yes	⚠️ Partial (Orchestrator hooks)	⚠️ Limited
XAI Module	✅ Yes	✅ Yes	⚠️ Conceptual (selector pipeline)	✅ Yes
Safety TF (EBOS)	✅ Yes	✅ Yes	⚠️ Edge Core SDK under integration	❌ Not yet
Security TF (IQBT)	✅ Yes	✅ Yes	✅ Yes	✅ Yes
Privacy TF (SHG)	✅ Yes	✅ Yes	❌ Not yet	❌ Not yet
Resilience TF (TID)	✅ Yes	✅ Yes	⚠️ Partial (CAPIF hooks)	❌ Not yet
Reliability TF (UNIWA)	✅ Yes	✅ Yes	✅ Yes	✅ Yes

Table 9 Roadmap - Component Responsibilities Matrix

10.3 Alignment with Objectives & KPIs

This deliverable contributes directly to the core technical objectives defined in the SAFE-6G GA, particularly Objective O.2 (design and implementation of the CoCo and AI-driven orchestration logic) and Objective O.3 (development of five user-centric TF).

The status of implementation is tracked against Key Performance Indicators (KPIs) and Key Value Indicators (KVIs) defined in the proposal. The table below summarizes the alignment of each major WP4 component with the relevant project objectives and associated KPIs:

Component	*Objective(s)*	*KPI / KVI*	*Description / Target*	*Current Status*
Cognitive Coordinator	O.2	KPI 2.1	Initial implementation of intent parsing and LoTw computation	✅ Implemented (regression + reasoning)
XAI Module	O.2	KPI 2.2	Ability to generate local/global model explanations	⚠️ Conceptual stage; early experiments only
Chatbot-CoCo Interface	O.2	KPI 2.3	Integration of natural language trust input via chatbot	✅ Functional (UI + API integrated)
Security TF	O.3	KPI 3.1	Trust score-based security policy enforcement	✅ Integrated with SSI, blockchain, classifier
Safety TF	O.3	KPI 3.2	Secure SDP enforcement and deployment flavors	✅ Implemented

Privacy TF	O.3	KPI 3.3	AI-driven privacy policy enactment on 6G core	⚠️ Architecture complete, implementation started
Resilience TF	O.3	KPI 3.4	Dynamic trust-driven orchestration of deployment templates	⚠️ Model design done; integration pending
Reliability TF	O.3	KPI 3.5	QoE-based model selection and action orchestration	✅ LoTw model logic and telemetry covered
MLOps Pipeline	O.2	KPI 2.4	Trust-function-aware model lifecycle integration	⚠️ Kubeflow integration in progress
Broker-based orchestration	O.2, O.3	KVI 1.1	Use of a modular, decoupled pub/sub system	✅ Redpanda-based broker operational

Table 10 Roadmap - Component KPI Alignment

KPI category	***KPI name***	***KPI Target Value***	***Current Status***
Latency	Tunnel latency	<15 ms	Preliminary testing indicates latency and jitter are within acceptable limits. Formal benchmarking will be conducted in Release B to confirm compliance under full load conditions
Latency	Tunnel Jitter	<2 ms	
Capacity	Tunnel Throughput Efficiency	>85%	Early indications suggest throughput efficiency meets target levels. Detailed performance measurements and multi-user load testing are planned for Release B
Operational Network	Tunnel Packet Loss	<0.1%	Functional validation successful, no quantitative reliability data yet collected. Endurance and stability tests will be executed in Release B to gather precise metrics.
Operational Network	Tunnel establishment success	>90%	
Operational Network	Tunnel establishment failure	<10%	
Safety Trust Function	Provision of Level of Safety	< = 3 min	Initial profiling indicates resource use within design targets. Comprehensive scalability and performance tests will be completed in Release B.
Scalability	Resources per Tunnel	=3Mi CPU, =60MB RAM	

Table 11 Roadmap – Safety Function KPI Alignment

KPI category	*KPI name*	*KPI Target Value*	*Current Status*
Blockchain	Peer Count	>= 5 Peer	❌ Utilizing 3 peers for component development, KPI target is for integrated system
Latency	Transaction Latency	<2 seconds for transaction endorsement, <2-3 seconds for final confirmation.	✅ 1.2s for transaction endorsement, 2s for final confirmation
Blockchain	Block Finalization	< 5s	✅ 1-3s for block finalization.
Blockchain	Transaction Throughput	100 - 200 TPS	✅ 300 TPS achieved
Blockchain	Network Uptime	> 99 %	⚠️ Will be measured in integrated system

Table 12 Roadmap – Security Trust Function KPI Alignment

KPI category	*KPI name*	*KPI Target Value*	*Current Status*
Latency	Privacy Score Calculation Latency	≤ 3 seconds	❌ Currently using calibrated Level of Privacy from CoCo, not calculated by the TF
Latency	Privacy Action Recommendation Latency	≤ 3 seconds	✅ Rule base implementation takes 70ms
Reliability and Availability	Privacy Function Uptime	≥ 99.9%	❌ Cannot meassure it until all systems are integrated
Performance	Privacy vApp flavor selection accuracy	≥ 80% satisfaction	✅ Confirmed via manual audits and log analysis
Privacy Trust Function	Level of Privacy	< = 3 min	⚠️ Integration in progress

Table 13 Roadmap – Privacy Trust Function KPI Alignment

KPI category	*KPI name*	*KPI Target Value*	*Current Status*
Latency	Resilience Score Calculation Latency	≤ 1 second	✅ Calculated the score takes 90ms

Latency	Resilience Action Recommendation Latency	≤ 3 seconds	✅ Rule-based implementation takes 90ms
Reliability and Availability	Resilience Function Uptime	≥ 99%	❌ Cannot measure it until all systems are integrated
Performance	AI Orchestrator Satisfaction Rate	≥ 80% satisfaction	⚠Need to be confirmed via manual audits
Resilience Trust Function	Level of Resilience	<= 3 min	⚠Integration in progress

Table 14 Roadmap – Resilience Trust Function KPI Alignment

KPI category	***KPI name***	***KPI Target Value***	***Current Status***
Performance and Accuracy	Number of metrics	≥ 10	✅ Achieved. More than 10 metrics were collected and processed by the TF.
Performance and Accuracy	Type of alerts	≥ 3	⚠️ Ongoing. 1 type of alert has been produced by the TF (for scaling up or out of resources).
NFV energy efficiency	NFV green deployment	≥ 15%	⚠️ Ongoing. Integration with aerOS in progress. AerOS can provide the option to support the service deployment into an infrastructure powered by renewable energy sources.
ML energy efficiency	Energy-aware ML models	≥ 20%	⚠️ Ongoing. Low complexity models are already trained that lead to lower energy consumption during training/inference, compared with more complex models for the same task.
Reliability Trust Function	Level of Reliability	≤ 3 min	✅ Achieved. Measured, and it is less than one minute. The measured time includes data collection, model inference time, and communication-related delays between TF and the Cognitive Coordinator.

Table 15 Roadmap – Reliability Trust Function KPI Alignment

10.4 Development Phases and Integrations

The development of WP4 components is conducted in a phased approach that enables the gradual integration of system-wide features with early functional prototyping. The following timeline is designed to account for the technical interdependencies between components, the progressive readiness of orchestration and infrastructure layers (e.g., CoCo, aerOS, Cumucore), and the evolving maturity of each TF. While not all integration milestones are rigidly fixed, the phases below reflect the

general progression path adopted by the consortium, with particular focus on leveraging modular communication (via Redpanda/Kafka) and standardized orchestration (via OpenCAPIF and aerOS).

Phase 1 – Function Prototyping (M6–M18): During this phase, the core logic for each TF was implemented in isolation, with a focus on:

- Independent development of Trust Function ML models and internal AI agent flows.
- CoCo regression and reasoning engines implemented using BERT-based models and semantic scoring.
- XAI experiments conducted using tabular inference data (e.g., CPU usage), validated via ALE and PDP.
- Initial interactions tested via a Kafka-compatible broker (Redpanda) for publishing trust-score messages.
- Chatbot UI completed; tested with API-based LoTw query delivery to the CoCo.
- Minor integration with external systems occurred during this phase — efforts were primarily internal, targeting interface definitions, core loop stabilization, and component-level testing.

Phase 2 – Controlled Interaction (M18–M27): This phase focuses on bringing together the previously siloed components through orchestrated data and message flows. The main goals include:

- Integration of the **Security** and **Reliability** Trust Functions with the SAFE-6G **6G Core testbed (Cumucore)** and **aerOS** platform.
- Connection of all Trust Functions to the **Kafka trust-score topic**, enabling consumption of nLoTw/cLoTw values and policy reaction.
- Deployment of **CoCo-Orchestrated vApp flavors** in response to synthetic or test-driven trust scores.
- Use of **OpenCAPIF** to handle function discovery, authorization, and exposure of APIs to the core.
- Activation of the **Chatbot→CoCo pipeline**, now functional and publishing user intents to the regression engine via RESTful API.
- Focused effort on **integration scripting** for data wiring, orchestration simulation, and triggering of TF reactions using Cumucore telemetry.
- Early version of **MLOps→GitLab→Kubeflow CI/CD loop** under design, with initial manual retraining triggers in place.
- No unified explanation delivery to users yet; the XAI module is being integrated with one function at a time for testing.

Phase 3 – Full Integration and Feedback Loops (M27–M33): This phase aims to validate the system end-to-end with real-time flows, user interaction, and feedback-enabled learning. Key highlights:

- All TFs will be deployed as reactive agents across the edge-cloud continuum via aerOS, subscribing to trust-score topics and applying corresponding actions through OpenCAPIF APIs.
- Feedback from TFs will be returned to the CoCo, triggering dynamic recalibration of LoTw, conflict resolution, and re-orchestration.
- XAI outputs will be injected into the user chatbot interface (formatting and LLM-based reformulation pipeline under development).
- MLOps workflows will be automated, enabling GitLab commits or performance drifts to trigger model updates across TF AI Agents.

- Monitoring across all TFs will be active, with plans to extend to structured feedback logging and eventually a dedicated feedback listener (TBD).

The phased development strategy outlined above directly aligns with SAFE-6G's milestones for WP4, including the final releases of the CoCo (D4.2) and each user-centric Trust Function (D4.3–D4.7), all due at Month 33. Intermediate integration will feed into the SAFE-6G testbed validation activities scheduled in WP5, particularly D5.2 (Integrated Ecosystem Final Release, M35) and D5.3 (System Level Verification and KPI Analysis, M36).

To facilitate smooth integration, all components are developed as modular, loosely coupled components using pluggable AI models and exposed orchestration hooks (e.g., OpenCAPIF, REST APIs). These will be progressively deployed on the aerOS platform and validated through the SAFE-6G testbed in collaboration with WP5.

As part of pilot readiness, all TFs and orchestration modules will be delivered in containerized form, with standardized APIs and configuration manifests. Initial deployment tests will be conducted in local integration environments during Phase 3, with final handover to WP5 planned for M32–M33 to support the SAFE-6G pilot campaigns.

11 CONCLUSION

This deliverable marks the first consolidated technical output of WP4, establishing the architectural foundation and early-stage implementation of the SAFE-6G trustworthiness framework. Through the Cognitive Coordinator, five modular Trust Functions, and an evolving Explainable AI component, the project introduces a novel approach to managing trust in 6G environments—moving from static enforcement models to dynamic, user-centric, and intent-driven orchestration.

All TFs share a common layered architecture comprising Local AI Agents, vApps, and nApps. This modularity ensures that trust decisions can be computed, explained, and enforced at the appropriate system layer—supporting flexible deployment across heterogeneous infrastructures. The CoCo complements this architecture by translating user intents into actionable trust policies through semantic reasoning, ML-based prediction, and contextual feedback loops.

A phased development roadmap has been defined, with clear alignment to the SAFE-6G GA milestones. During Phase 3, each component will mature into an integration-ready module. These will be delivered in containerized format with standardized APIs and orchestration hooks, facilitating their deployment within the SAFE-6G platform. Handover of the production-ready components to WP5 is scheduled for M32, enabling final validation in the project's pilot campaigns. The upcoming deliverables D4.2 through D4.7 will report on the complete implementation and evaluation of each TF, while D5.2 and D5.3 will demonstrate system-level integration and KPI compliance in real-world 6G testbed environments.

12 REFERENCES

[1] E. S. Taleghani, R. I. M. Valencia, A. L. S. Orozco and L. J. G. Villalba, "Trust Evaluation Techniques for 6G Networks: A Comprehensive Survey with Fuzzy Algorithm Approach," Electronics, vol. 13, no. 15, p. 3013, 31 July 2024.

[2] Y. Wang, X. Kang, T. Li, H. Wang, C.-K. Chu and Z. Lei, "SIX-Trust for 6G: Toward a Secure and Trustworthy Future Network," IEEE Access, vol. 11, p. 107657–107668, 2023.

[3] Alonso-Lupez, J. A., L. A. M. Hernández, S. P. Arteaga, A. L. S. Orozco, L. J. G. Villalba, A. Pastor and D. L. García, "Level of Trust and Privacy Management in 6G Intent-Based Networks for Vertical Scenarios," 6GNet 2022 Proceedings, p. 1–4, 2022.

[4] S. F. Drampalou, D. Uzunidis, A. Vetsos, N. I. Miridakis and P. Karkazis, "A User-Centric Perspective of 6G Networks: A Survey," IEEE Access, 2024.

[5] Y. Yin and H. Fang, "A Novel Multiple Role Evaluation Fusion-Based Trust Management Framework in Blockchain-Enabled 6G Network," Sensors, p. 6751, 2023.

[6] P. Bhattacharya, S. Arpit, T. Sudeep, K. Neeraj and S. Ravi, "6Blocks: 6G-Enabled Trust Management Scheme for Decentralized Autonomous Vehicles," Computer Communications, vol. 191, p. 53–68, 2022.

[7] G. Tripi, A. I. L. Rinieri and M. Prandini, "Security and Trust in the 6G Era: Risks and Mitigations," Electronics, vol. 13, no. 11, p. 2162, 2024.

[8] "oAuth net," [Online]. Available: https://oauth.net/2/.

[9] "ETSI Software Development Group OpenCAPIF," [Online]. Available: http://ocf.etsi.org.

[10] "The Mobile Private Network 5G Core," [Online]. Available: https://cumucore.com.

[11] "The aerOS project, Horizon Europe No 101069732," [Online]. Available: https://aeros-project.eu.

[12] "Hyperledger Foundation," [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.5/.

[13] "fbProphet," [Online]. Available: https://pypi.org/project/fbprophet/.

[14] Wordnet documentation: https://wordnet.princeton.edu/

[15] nltk library documentation: https://github.com/nltk/nltk/tree/develop

[16] Dataset opensource and available at: https://zenodo.org/records/7858064

[17] F. Rodriguez, I. Ahmad, J. Huusko and K. Seppänen, "Towards dependable 6G networks," TechRxiv, 07 July 2022.

[18] K. Gunasekaran, S. Dhanasekaran, R. V. Kumar and S. Aswath, "Advanced Beamforming and Multi-Access Edge Computing: Empowering Ultra-Reliable and Low-Latency Applications in 6G Networks," International Journal of Communication Systems, vol. 38, no. 5, 31 October 2024.

[19] R. Chataut, M. Nankya and R. Akl, "6G Networks and the AI Revolution—Exploring Technologies, Applications, and Emerging Challenges," Sensors, vol. 6, no. 24, p. 1888, 15 March 2024.

[20] S. M. Mohammed, A. Al-Barrak and N. T. Mahmood, "Enabling Technologies for Ultra-Low Latency and High-Reliability Communication in 6G Networks," Ingénierie des Systèmes d'Information (ISI), pp. 1195-1208, 2 June 2024.

[21] SAFE-6G, "A Smart and Adaptive Framework for Enhancing Trust in 6G Networks," Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101139031, 2023. [Online]. Available: https://safe-6g.eu/.

[22] S. F. Drampalou, D. Uzunidis, A. Vetsos, N. I. Miridakis and P. Karkazis, "A User-Centric Perspective of 6G Networks: A Survey," IEEE Access, vol. 12, pp. 190255 - 190294, 12 December 2024.

[23] J. Ortin, P. Serrano, J. Garcia-Reinoso and A. Banchs, "Analysis of Scaling Policies for NFV Providing 5G/6G Reliability Levels With Fallible Servers," IEEE Transactions on Network and Service Management, vol. 19, pp. 1287 - 1305, June 2022.

[24] D. Krummacker, B. Veith, D. Lindenschmitt and H. D. Schotten, "DLT Architectures for Trust Anchors in 6G," Annals of Telecommunications, vol. 78, no. 9, p. 551–560, 2023.