CASTOR

Building Trust
in the Computing Continuum

D6.1
CASTOR Integrated Framework, Use Case Analysis & Interim Report on PoC Verification Results

Project number:	101167904
Project acronym:	**CASTOR**
Project title:	Continuum of Trust: Increased Path Agility and Trustworthy Device and Service Provisioning
Project Start Date:	1st October, 2024
Duration:	36 months
Programme:	HORIZON-CL3-2023-CS-01
Deliverable Type:	Other
Reference Number:	HORIZON-CL3-2021-CS-01-101167904/ D6.1 / v1.0
Workpackage:	WP6
Due Date:	31st March, 2026
Actual Submission Date:	8th April, 2026
Responsible Organisation:	ORO
Editor:	Ioan Constantin
Dissemination Level:	Sensitive
Revision:	1.0
Abstract:	This deliverable presents the foundational groundwork and early integration designs toward the first version of the CASTOR integrated framework. To facilitate proactive testing and technical alignment, the document specifies a series of validation ”warm-up sessions” structured as Proof-of-Concept (PoC) narratives, which serve as the initial validation phase within a common PoC routing environment. Furthermore, the document details the overarching CASTOR validation and evaluation framework, establishing a comprehensive, multi-testbed infrastructure capable of supporting diverse experimental requirements—ranging from CASTOR-enabled trust extensions to realistic network workload needs. Finally, it reports on the development and validation activities towards shaping the use case evaluation environments for the upcoming testing activities.

Funded by EU's Horizon Europe programme under Grant Agreement number 101167904 (CASTOR) and supported by the European Cybersecurity Competence Centre. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Cybersecurity Competence Centre. Neither the European Union nor the European Cybersecurity Competence Centre can be held responsible for them.
This work has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI). Funded by UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee.

Copyright Notice

Project Funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable	OTHER*	
Dissemination Level		
PU	Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page)	
SEN	Sensitive, limited under the conditions of the Grant Agreement	X
Classified R-UE/ EU-R	EU RESTRICTED under the Commission Decision No2015/ 444.	
Classified C-UE/ EU-C	EU CONFIDENTIAL under the Commission Decision No2015/ 444	
Classified S-UE/ EU-S	EU SECRET under the Commission Decision No2015/ 444	

* R: Document, report (excluding the periodic and final reports)
DEM: Demonstrator, pilot, prototype, plan designs
DEC: Websites, patents filing, press & media actions, videos, etc.
DATA: Data sets, microdata, etc.
DMP: Data management plan
ETHICS: Deliverables related to ethics issues
SECURITY: Deliverables related to security issues
OTHER: Software, technical diagram, algorithms, models, etc.

Editor

Ioan Constantin (ORO)

Contributors (ordered according to beneficiary numbers)

Nikos Fotos, Sofianna Menesidou, Panagiotis Banavos, Thanassis Giannetsos (UBITECH)
Iasonas Sakellariou, Stelios Kazazis, Symeon Tsintzos (QUBITECH)
Fabian Schwarz, Meni Orenbach (NVIDIA)
Yalan Wang, Liqun Chen (SURREY)
Alexandru Coleș, Ioan Constantin (ORO)
Vlad Chiriac, Tiberius-George Sorescu, Tudor-Gabriel Ghetau, Ciprian-Romeo Comsa (TUIASI)
Riccardo Orizio, Michael McElligott, Stelios Basayiannis (COLLINS)
Vangelis Kosmatos, Panagiotis Pantazopoulos (ICCS)
Konstantinos Latanis (SUITE5)
Christos Dalamagkas, Ioannis Boukas, Evangelos Syrmos (K3Y)
Aristi Galani, Sokratis Barmpounakis (WINGS)
Pablo Martinez, Antonio Skarmeta (UMU)
Alexandros Fakis, Kostas Maliatsos (FERON)
Gergely Kovacs, Andras Edelmayer (COMMSIGNIA)
Anuj Pathania, Andy Pimentel (UvA)
Jamie Pont, Budi Arief, Theo Dimitrakos (UKENT)

Disclaimer

Executive Summary

Deliverable D6.1 provides the first milestone toward the realization of the CASTOR integrated framework and its evaluation in the context of the project's four use cases. It captures the consolidated blueprint for the integration across all technical artifacts within the CASTOR architecture and reports the development activities pertaining to the creation of early validation scenarios (in the form of Proof-of-Concept narratives), the specification of the CASTOR testbeds, and the development activities for the realization of the use case scenarios.

Specifically, this deliverable documents the transition from theoretical trust models to a functional, integration-ready architectural blueprint. Furthermore, detailed implementation planning utilizing standardized interfaces, such as Kafka-based message buses and REST APIs, establishes a clear trajectory from the initial D2.1 reference architecture to the first integrated framework release planned for D6.2.

To facilitate early validation, the establishment of the PoC Playground provides a controlled, simulation-driven environment for testing the project's security artifacts. These playground experiments aim to validate the end-to-end CASTOR operational lifecycle within an emulated domain before deploying the full-fledged CASTOR integrated framework in the context of the use case evaluations.

A significant operational outcome of this document is the demonstrated readiness of the overarching CASTOR validation and experimentation framework. As established, this relies on a multi-tiered infrastructure anchored by the primary use case validation testbed, which delivers a robust, production-grade forwarding plane capable of handling realistic network workloads. This primary facility is critically supported by specialized auxiliary testbeds dedicated to benchmarking hardware-rooted trust extensions, as well as providing advanced Kubernetes orchestration and lifecycle management.

Finally, this deliverable confirms the readiness of the envisioned CASTOR use case workflows and their strategic positioning within the primary testbed infrastructure. By successfully mapping these realistic, high-demand scenarios onto the overarching validation framework, the project establishes a fully prepared environment for complex evaluation. This ensures that CASTOR is poised to seamlessly support diverse application domains characterized by varying network demands and strict trust requirements in the upcoming evaluation phases.

Contents

List of Figures

List of Tables

Versioning and contribution history

Version	Date	Author	Notes
v0.1	20.12.2025	Ioan Constantin (ORO)	Table of Contents created
v0.2	16.1.2026	Ioan Constantin (ORO)	Input on ORO testbed capabilities in Chapter 4
v0.3	22.1.2026	Michael McElligott (COLLINS), Gergely Kovacs, Andras Edelmayer (COMMSIGNIA), Vlad Chiriac (TUIASI), Ioannis Boukas, Christos Dalamagkas (K3Y)	Summary of use case scenarios and technical description of use case applications in the scenarios. (Chapters 5-8).
v0.4	12.2.2026	ALL	First full draft on PoC specification including KPI dimensions (Chapter 3).
v0.5	26.2.2026	ALL	Revised PoC KPI table in each of the five PoCs in Chapter 3.
v0.6	5.3.2026	Sofianna Menesidou, Nikos Fotos (UBITECH)	First draft of the technical integration diagram towards the CASTOR integrated framework (Chapter 2)
v0.7	9.3.2026	Ioan Constantin, Alexandru Coles (ORO)	Revised input in CASTOR validation framework (Chapter 4).
v0.8.1	9.3.2026	Nikos Fotos, Sofianna Menesidou (UBITECH), Pablo Martinez (UMU), Aristi Galani (WINGS), Alexandru Coleș (ORO)	Revised input for Testbed description (Chapter 4)
v0.8.2	12.3.2026	Michael McElligott (COLLINS), Gergely Kovacs, Andras Edelmayer (COMMSIGNIA), Vlad Chiriac (TUIASI), Ioannis Boukas, Christos Dalamagkas (K3Y)	First full draft on technical readiness of use case environments(Chapters 5-8).
v0.9.1	19.3.2026	ALL	Updated specification of CASTOR Validation framework (including all available testbeds). Updated kafka topic specification and interfaces in Chapter 2. Revised input on PoC narratives and PoC KPIs (Chapter 3).
v0.9.2	26.3.2026	Michael McElligott (COLLINS), Gergely Kovacs, Andras Edelmayer (COMMSIGNIA), Vlad Chiriac (TUIASI), Ioannis Boukas, Christos Dalamagkas (K3Y)	Revised CASTOR Use Case Evaluation Plan (Chapters 5-8).
v0.9.3	2.4.2026	ALL	Final review and revision of CASTOR Integrated framework (Chapter 2)
v1.0	8.4.2026	Daphne Galani (UBI)	Final revision & Submission

Chapter 1

Introduction

1.1 Scope and Purpose

The primary scope of Deliverable D6.1 is to transition the CASTOR project from the conceptualization of individual building blocks—as seen in earlier architectural deliverables—to a unified, observable, and verifiable ecosystem. While other technical work packages focus on the isolated development of the CASTOR in-router trust-extensions (WP3), the Trust Assessment Framework (WP4), and the Dynamic Path Enforcement mechanisms (WP5), the scope of this document is explicitly integration-centric. It defines how these disparate layers communicate via a standardized set of APIs and telemetry protocols to achieve a ”below-zero-trust” environment. Furthermore, the document encompasses the technical deployment of CASTOR assets, ensuring that evidence collection remains consistent across the underlying experimentation infrastructure.

D6.1 serves as a comprehensive technical interim report, documenting both the current integration status of functional assets and the development of use case scenarios in preparation for the first integrated release, which will be reported in D6.2. It provides the formal definition of the logical interfaces and architectural blueprints necessary to establish a Trustworthy Compute Continuum that spans from the routing plane to the core orchestration layer. Finally, the document defines the ”PoC Playground”—a controlled simulation environment designed for the initial verification of core functionalities, culminating in the trust-aware Traffic Engineering pipeline of the CASTOR framework.

The document also details the hardware and software readiness of the various experimentation platforms, including the primary testbed for use case evaluation and its complementary auxiliary environments. At the core of the overarching CASTOR validation framework is the full-fledged ORO testbed, which provides a production-grade forwarding plane designed to handle realistic network workloads. To streamline the validation and benchmarking of the CASTOR in-router trust extensions, a first auxiliary testbed provides the necessary hardware-based Root-of-Trust (RoT) capabilities. In parallel, to benchmark the enhanced lifecycle management of both the trust extensions and the overall forwarding plane, a second auxiliary testbed offers advanced Kubernetes orchestration capabilities.

Finally, this deliverable captures the readiness level of the use case workflows and their positioning within the available testbeds. This constitutes a significant milestone, as it showcases the maturity of all evaluation scenarios, enabling the integration of the Use Case environments into the ORO testbed and marking the commencement of the first evaluation phase to be documented in D6.2. By defining the resource allocations, virtualization layers, and monitoring frameworks in each use case environment, the deliverable ensures that the technical foundation is fully prepared for multi-domain, multi-actor validation activities. In summary, the document bridges the gap between theoretical trust models and practical, scalable enforcement mechanisms within a heterogeneous and dynamic compute continuum.

1.2 Relation to other WPs and Deliverables

The CASTOR Integrated Framework presented in this deliverable functions as a technical nexus, consolidating requirements, architectural specifications, and functional assets derived from multiple work packages while providing the necessary validation substrates for subsequent project phases. The positioning of D6.1 within the project lifecycle is strategically aligned to transition from theoretical modeling to empirical verification across distributed testbed infrastructures.

Fundamental to this integration is the relationship with Work Package 2 (WP2), specifically Deliverable D2.1. The use case definitions, security requirements, and Key Performance Indicators (KPIs) established in D2.1 serve as the primary baseline for the ”As-Is” vs. ”To-Be” comparative analysis conducted in the current report. The four high-level phases of the CASTOR framework—Preparedness, Service Registration, Proactive, and Reactive—were originally identified in D2.1 and are here operationalized through specific component interactions and interface definitions.

The architectural blueprint detailed in this document is closely coupled with Work Package 5 (WP5), which focuses on the development of trust-aware Traffic Engineering (TE) policies and optimization engines. D6.1 provides the logical and physical infrastructure—comprising virtual routers (vRouters) with Segment Routing (SR) and Flex-Algo support—required to execute the TE algorithms developed in WP5. The Trust Awareness API defined herein enables the WP5 optimization processes to ingest real-time Actual Trustworthiness Levels (ATLs) to compute optimal paths that satisfy Secure Service Level Agreements (SSLAs).

Furthermore, the framework incorporates cryptographic primitives and device attestation mechanisms developed in Work Package 3 (WP3) and Work Package 4 (WP4). The Trusted Network Device Extensions (TNDE) and the Trust Assessment Framework (TAF), which are central to the evidence collection and trust quantification processes described in this deliverable, represent the physical and logical implementation of the security assets matured in those work packages. The integration of these assets into the CASTOR Trusted Computing Base (TCB) enables the secure message exchange and I/O virtualization capabilities that define the CASTOR approach.

As an interim report on Proof of Concept (PoC) verification, D6.1 establishes the readiness of the multi-site experimentation platform, including the ORO, COLLINS, CMS, TUIASI, and K3Y testbeds. This infrastructure is essential for the full-scale pilot activities and final validation tasks scheduled for later stages of the project.By defining the common playground and the initial set of PoCs, this deliverable ensures that the functional components developed across the consortium can be evaluated in a controlled, interoperable environment prior to their final deployment.

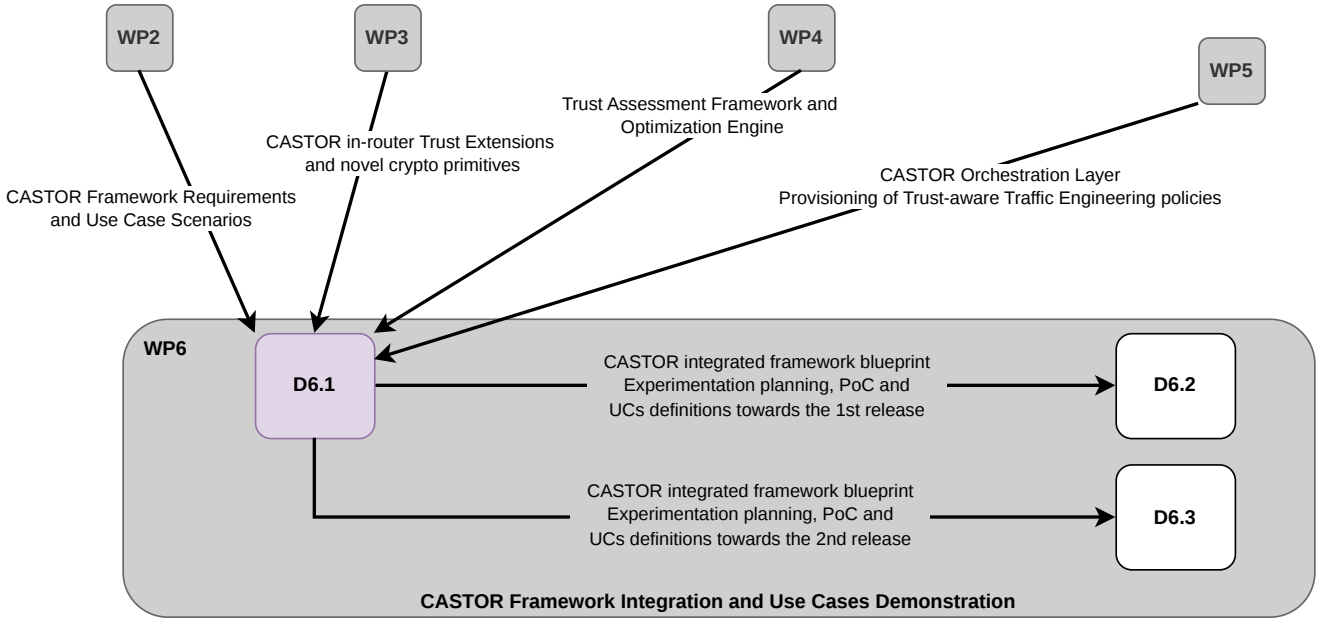

Figure 1.1: Relation of D6.1 with other WPs and Deliverables

1.3 Deliverable Structure

This deliverable is organized into ten chapters, transitioning from the architectural definition of the CASTOR framework to the practical implementation and preliminary verification of its components across a multi-site experimentation platform.

Chapter 1 establishes the document's scope, purpose, and its functional relationship with other project work packages.

Chapter 2 provides the technical blueprint of the CASTOR Integrated Framework. It details the functional and logical architecture, including the CASTOR Trusted Computing Base (TCB) and the specific interfaces—such as REST APIs and KAFKA pub/sub topics—required for inter-component communication. This chapter also outlines the implementation roadmap for upcoming framework releases.

Chapter 3 specifies the "PoC Playground" a common network topology designed for the initial verification of core CASTOR functionalities. It details the evaluation points for trustworthiness evidence collection, trust quantification, the profiling of the optimization engine, and the enforcement mechanisms for Segment Routing (SR) policies within a controlled environment.

Chapter 4 details the overarching CASTOR validation and experimentation framework. It first focuses on the primary ORO testbed, highlighting its 5G Standalone (SA) capabilities, slicing architectures, and monitoring frameworks designed to accommodate realistic network workloads. Furthermore, the chapter presents the two complementary auxiliary testbeds utilized for Proof-of-Concept (PoC) benchmarking: the multi-site TNDE environment (operated by UMU and UBI), which supports heterogeneous hardware-based Trusted Execution Environments (TEEs) and confidential computing technologies, and the WINGS testbed, which delivers advanced Kubernetes orchestration.

Chapters 5 through 8 report the readiness status of the use case application workflows, outlining their planned deployment strategies and the concrete evaluation plans targeted for the subsequent deliverables:

- Chapter 5 (COLLINS): Addresses urban air mobility surveillance and on-airport trusted routing, defining the baseline software readiness for telemetry and command-and-control traffic.

- Chapter 6 (CMS): Focuses on trustworthy V2X communications for first responders, specifying the PKI and Over-the-Air (OTA) update service properties.
- Chapter 7 (TUIASI): Details the functional architecture for accident alert and traffic monitoring workflows, including the supporting dashboard infrastructure.
- Chapter 8 (K3Y): Describes the baseline environment for UAV swarm management and mission coordination within a virtual routing topology.

Chapter 9 summarizes the primary outcomes of this report and establishes the strategic roadmap for the two subsequent rounds of validation and evaluation activities planned for upcoming deliverables of WP6.

Chapter 2

The CASTOR Integrated Framework

2.1 CASTOR Architectural Blueprint and Interfaces

CASTOR's trust-aware approach to Traffic Engineering (TE) requires a comprehensive framework spanning the compute continuum. This framework addresses both the routing and control planes while also supporting the lifecycle management of network elements and ensuring service assurance guarantees. The overall architecture of the CASTOR environment was previously defined in D2.1 [4], where four phases—though not strictly sequential—were identified: (a) Preparedness, (b) Service Registration, (c) Proactive, and (d) Reactive (see Figure 2.1). Next section summarises the CASTOR Architecture describing the CASTOR phases and components, shaping the path towards trust-aware TE policy enforcement.

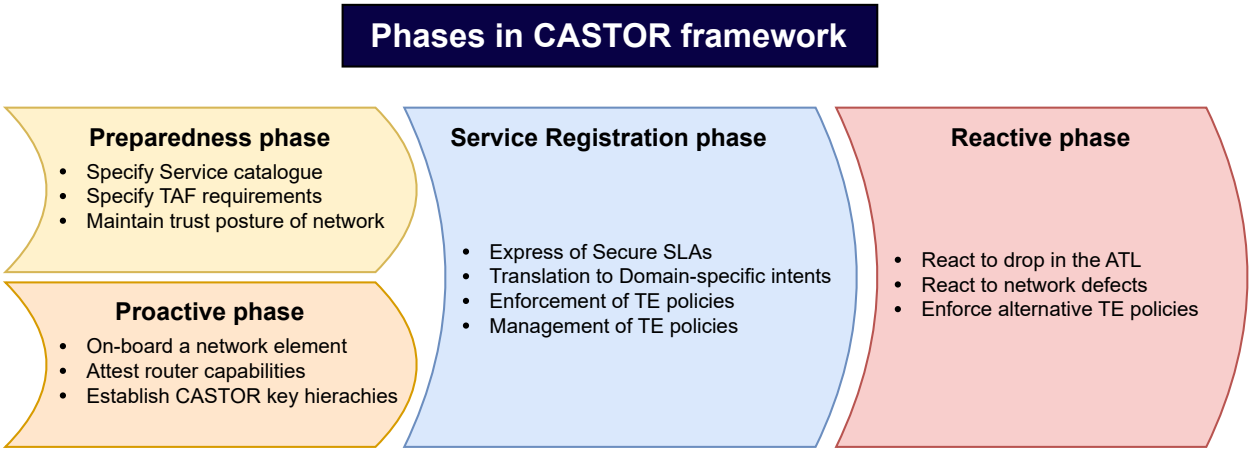

Figure 2.1: High-level phases in CASTOR framework

The **Preparedness phase** includes the configurations required for an OSS environment to support new service requests while ensuring the required assurance levels. It spans the compute continuum and assumes that the infrastructure layer (i.e., the network topology) is already deployed. At the orchestration layer, domain operators define the service catalogue. In CASTOR, this includes the specification of path profiles addressing different network and trust requirements. The preparedness phase also introduces the mechanisms that enable CASTOR to identify optimal paths and enforce trust-aware routing policies in the infrastructure layer. In addition, probing mechanisms are provisioned in network elements to allow the continuous evaluation of (S)SLA compliance.

Once the domain environment is prepared, the Service Orchestrator can process incoming requests. After the service-level objectives are agreed upon, the **Service Registration phase** describes how CASTOR realizes service placement when a provider expresses network and trust intents. This phase translates high-level service requirements into domain-specific constraints and enforces the corresponding

traffic engineering policies in the infrastructure layer, enabling end-to-end connectivity through paths that satisfy the agreed (S)SLA objectives.

The **Proactive phase** addresses the dynamic onboarding of new routers and the adaptation of existing paths. During secure onboarding, the new router and the Service Orchestrator exchange guarantees regarding the integrity and correct operation of critical router functionalities. Similar guarantees are exchanged in the data plane between network elements to discover secure links across the topology. These actions allow the orchestrator to validate and enrol the new node into the topology and incorporate it into candidate paths. The proactive phase also addresses situations where the current infrastructure cannot meet specific security guarantees, ensuring that the necessary security mechanisms are enforced to enable the realization of particular (S)SLA requirements.

Finally, the **Reactive phase** focuses on the framework's response after service deployment. During the service lifecycle, network or trust-related changes may cause a traffic engineering policy to violate the agreed (S)SLA. While routing protocols provide inherent self-healing mechanisms, the CASTOR reactive phase refers to the framework's ability to adapt and enforce updated policies when required. Runtime changes are classified as network-driven (e.g., congestion, link failures, topology changes) or trust-driven (e.g., reduced trustworthiness of a network element). In both cases, CASTOR determines appropriate mitigation strategies, either by applying new security policies to existing paths or by establishing alternative traffic engineering paths.

For completeness, we provide a high-level overview of the architectural figures presented in previous deliverables. Figure 2.2 illustrates the overall CASTOR architecture, highlighting the four phases introduced earlier, while Figure 2.3 presents the updated and zoomed-in architecture of the CASTOR Orchestration Layer.

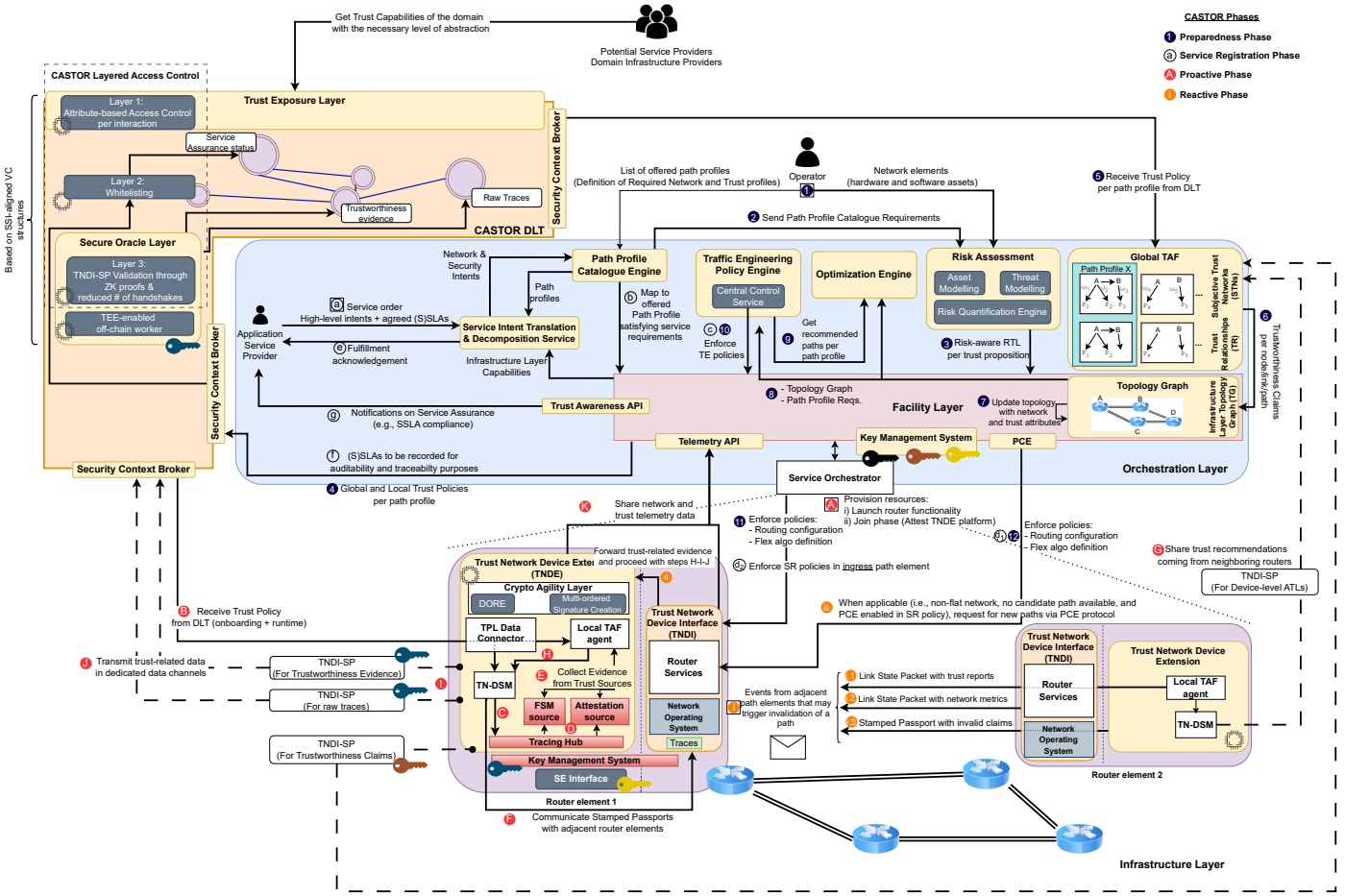

Figure 2.2: CASTOR high-level architecture [4]

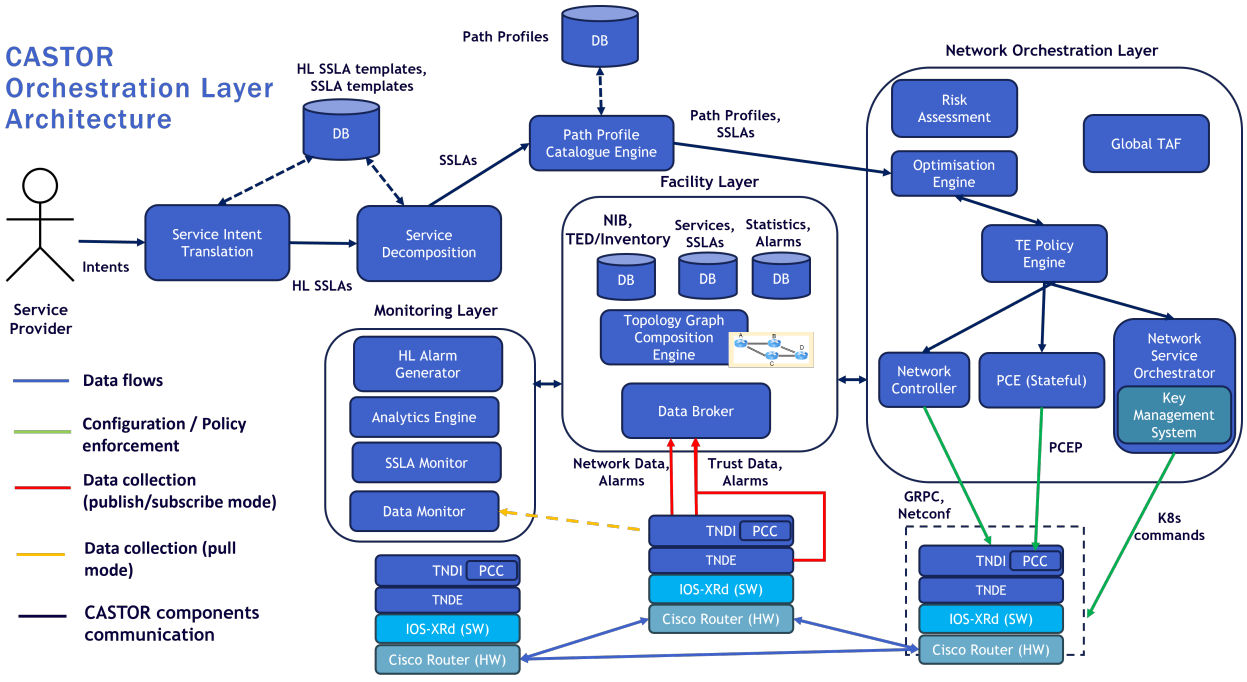

Figure 2.3: High-level architecture of CASTOR management and orchestration framework [5]

2.1.1 CASTOR Functional Component

CASTOR functional components are summarised in Table 2.1 as identified in D2.1 [4] and with minor updates depicting the current status and planning. Components with red are not considered for implementation, components with orange are planned for the 2nd release and components with blue are newly added.

Name	Main purpose
Orchestration Layer (and above)	
Service Intent Translation & Decomposition Service	Given an incoming service request, it translates the received high-level set of Intents to Service Level Agreements (SLAs) covering network and security/trust requirements (SSLAs).
Path Profile Catalogue Engine	Maps service requests and SSLAs to one of the offered path profiles.
Facility Layer	Manage/Serve (CASTOR- and application-related) requests from/to the different orchestration layers.
Service Orchestrator	Manage resources (e.g., spawn new VM for a vRouter function).
Topology Graph	Maintains a view of the trust- and network-profiles associated with the underlying network topology.
Trust Awareness API	Exposes interfaces to provide meaningful information with respect to the Service Assurance to the corresponding Service Provider.
Path Computation Element	Implements the Path Computation Element Protocol (PCEP) and enforces the SR policies realized by the CASTOR framework.
Risk Assessment Engine	Calculates the risk graph of the observable network segments taking into consideration the threats that can affect the router elements (e.g., threats related to HW/OS/SW stack); and contributes to the derivation of the risk-aware RTL logic to be enforced in the Global and Local TAF agents based on the offered path profile catalogue.
Global TAF	Assesses the trust profile of all the observable network segments based on the available evidence (i.e., Local TAF reports and additional trustworthiness evidence from each router).
Optimization Engine	Given the offered path profile catalogue and a snapshot of the (network and trust) characterization of the network segments, it recommends a set of optimal (primary and backup) paths that can satisfy the (network and trust) objectives associated with each path profile.

Traffic Engineering Policy Engine	Maps the set of recommended paths for each path profile into enforceable: i) Flex Algo Definition (FAD) to be advertised across the network segment, and ii) SR policy to be instantiated into the ingress node of the requested workload.
Secure Oracle	Part of the CASTOR DLT; checks the veracity of the data before being stored on the DLT.
Security Context Broker	Part of the CASTOR DLT; provides interfaces for authorized entities to interact with (i.e., process/consume) the on-chain data.
Trust Exposure Layer	Part of the CASTOR DLT; exposes interfaces for external stakeholders to access trust capabilities that are provided by a domain, while maintaining the necessary level of abstraction.
Network Service Orchestrator (NSO)	Manages node onboarding to ensure the availability of the required compute and network capabilities, monitors the compliance with the SLA/SSLA demands, and in case of deviations outside predefined thresholds triggers the responsible Orchestration entities.
Network Controller	The Network Controller serves as the base source of enforcement for all network configurations across routers. It covers all types of configuration, such as IGP and BGP settings, affinity bits for path computation based on trust evaluation, and SR-policy provisioning for head-end routers.
Trust Awareness API (SSLA Monitor)	Trust Awareness API enables relevant information transfer in order to ensure consistency in the abstracted capabilities of the domain or path that are exposed to Service Providers.
Topology Graph Composition Engine	The Topology Graph Composition Engine is responsible to retain and share among components this dynamic enriched topology graph. It combines data from from the Traffic Engineering Database (TED), the Management Information Base (MIB), the Path Profiles/SSLA Database and the Statistics/Alarms Database and enriches them with trust related information.
SSLA Monitor	The SSLA Monitor continuously verify that the SSLA-related statistics collected from the network elements are in line with the agreed SSLAs.
Data Monitor	The Data Monitor supersedes the previously defined Telemetry API. Whereas the original Telemetry API was envisioned as a generic exposure point for network metrics, the Data Monitor assumes a broader and more active role within the Monitoring Layer. It is responsible for collecting telemetry data and event-based information from the Data Broker, as well as metrics via pull-based mechanisms from network elements, compute nodes, and deployed services. It normalises raw data into specific schemas and formats, and relays the processed information to both the SSLA Monitor and the NSO. In addition, the it receives trust-related telemetry from the TNDE Local TAF agents via Prometheus exporters. Through these interactions, the Data Monitor serves as the central telemetry aggregation point within the Orchestration Layer, bridging infrastructure-level observability with the service assurance and trust assessment pipelines.
Infrastructure Layer (Routing plane)	
TN-DSM (TNDE)	(Part of the TNDE) Enables a CASTOR orchestrator to attest the correct state of the TNDE and securely onboard one or multiple TNDIs of a network device to the CASTOR network domain. The Trust Network Device Security Monitor (TN-DSM) implements the TNDI-SP to expose control and data channels towards CASTOR's upper layer components for configuration and trace/evidence/ATL sharing. The TN-DSM manages and reports the secure collection of trustworthiness evidence and local ATLs for each TNDI via the Tracing Hub, supported Trust Sources (i.e., Attestation and FSM sources), and local TAF.
TPL Data Connector (TAF)	(Part of the TAF) Retrieves the Trust Policy (i.e., Trust Model, RTL Logic, Target Trust propositions) of the operator for a TNDI and enables it in the TAF. The Trust Policy Language (TPL) Data Connector process any CASTOR Trust Policy and provision the necessary sessions so as to enable the continuous and dynamic trust assessment of the requested trust propositions.

Tracing Hub (TNDE)	(Part of the TNDE) The Tracing Hub is configured by the TN-DSM based on the trust policy to monitor runtime configuration and behavior traces of a TNDI for the evidence generation by the attestation and FSM sources. The Tracing Hub can operate multiple different Trace Units to collect TNDI traces. The Tracing Layer constitutes an overarching term that incorporates the tracing capabilities in CASTOR. This includes the Tracing Hub as well as any Trace Unit that may be connected to it in order to provide meaningful trace information to the CASTOR framework.
Attestation Source (TNDE)	(Part of the TNDE) Serves attestation requests and provides fresh attestation claims based on the traces collected of a TNDI (e.g., about a router's OS).
FSM Source (TNDE)	(Part of the TNDE) Receives runtime behavior traces of a TNDI via the Tracing Hub. The agent leverages pre-learned FSMs to detect anomalies from a TNDI's "normal" (benign) behavior and generates respective evidence for the TAFs.
Local TAF Agent (TNDE)	(Part of the TNDE) Performs the dynamic trust assessment within the network device based on the evidence received from the Attestation and FSM trust sources. It reports the ATLs to the relying parties (Global TAF, Orchestrator, DLT, and neighbouring routers) via the TN-DSM, e.g., through TNDI-SP data channels.
Trace Unit	The Trace Units are the TNDE-external tracing mechanisms operated by the Tracing Hub to collect configurational and/or behavioural traces from the TNDIs. CASTOR considers the exploration of different types of Trace Units, some located inside and other outside the TNDIs. The Trace Units are part of CASTOR's device-side TCB in the sense that the trustworthiness of the raw traces of that particular unit depends on the unit's security.

Table 2.1: CASTOR Artifacts - Naming convention

2.1.2 End-to-End flow of CASTOR Integrated Framework (release 1)

This section details the end-to-end operational flows of the CASTOR Integrated Framework by mapping the interfaces defined in Table 2.3 to three principal operational scenarios: (a) the Preparedness Phase and typical periodical behaviour, (b) the On-boarding of a new network element, and (c) the Service Request flow. These flows collectively illustrate how the Orchestration Layer coordinates trust-aware path establishment across the compute continuum, from initial domain configuration through service provisioning and continuous assurance. For each flow, the involved CASTOR interfaces are explicitly referenced using their identifiers from Table 2.3 and Figure 2.4, providing a traceable mapping between the high-level architectural operations described in D5.1 [5] and the concrete integration points realised by now.

Note: *The interface definitions listed in Table 2.3 are currently designated as Work In Progress (WIP). Both the exchanged content and the underlying technology for each interface are subject to refinement as the implementation of the CASTOR integrated framework progresses. More details will be provided in D6.2.*

2.1.2.1 Preparedness Phase and Typical Periodical Behaviour

The Preparedness Phase establishes the operational baseline of the CASTOR domain and encompasses all configurations required to enable trust-aware service provisioning. However, in this section we do not only provide the one-time necessary configurations but, we also document interactions that operate periodically to maintain an accurate and up-to-date view of the domain's network and trust posture throughout the system's lifecycle.

Trust Engineering Configuration. The process begins with the Risk Assessment component pushing the Trust Policy Metadata on-chain via the CASTOR DLT (`INT_RA_1`: Risk Assessment → CASTOR DLT; content: Trust Model Template, RTL; technology: HTTP). This interface carries the Trust Model Template and the Required Trust Levels (RTLs), which encode the domain operator's risk posture and serve as

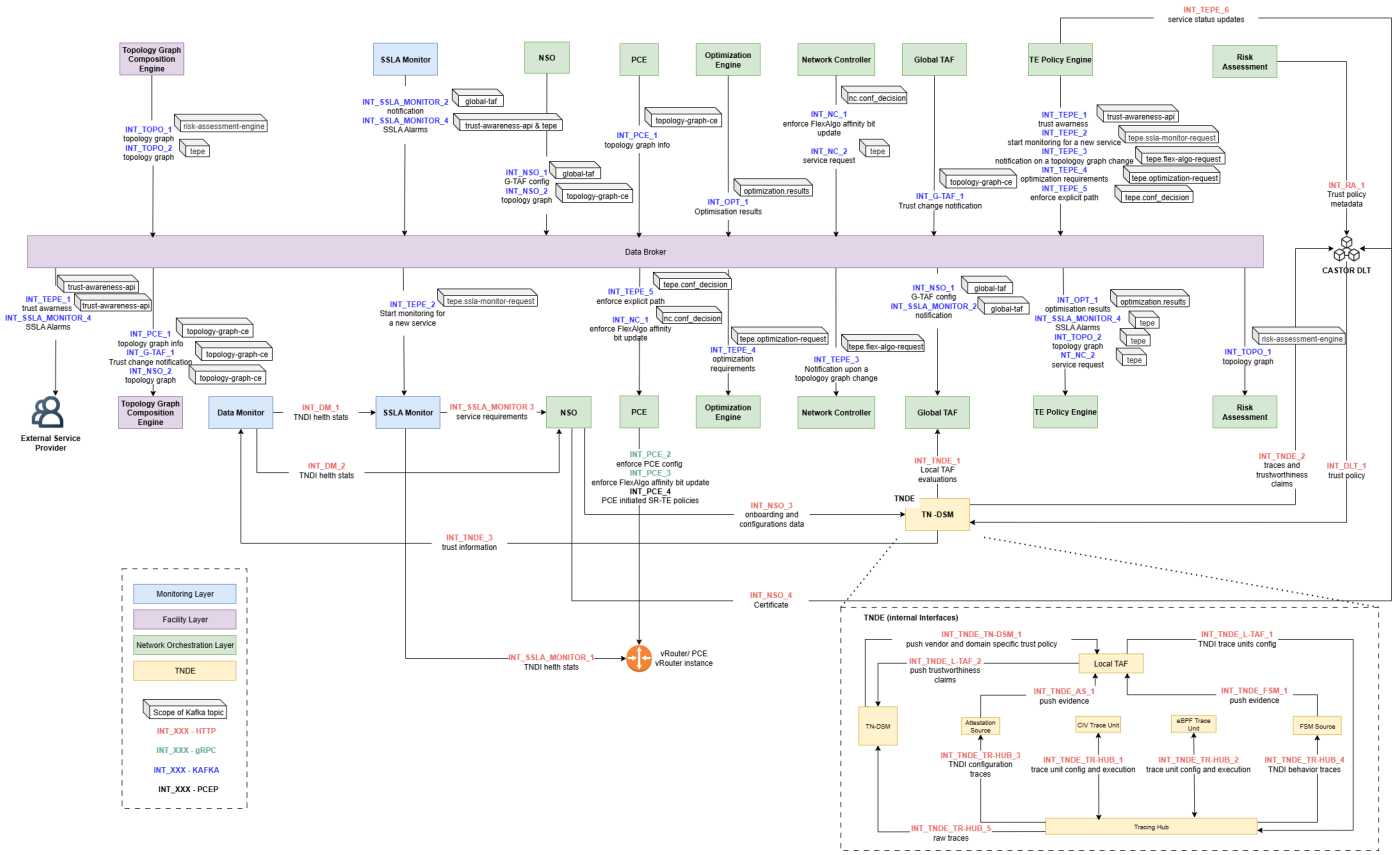

Figure 2.4: CASTOR Interfaces

the reference criteria against which the Actual Trust Levels (ATLs) of individual nodes and paths will be evaluated. As detailed in D5.1 (Section 5.3.1, Flow 1), both the Administrator and the Risk Assessment component can store and dynamically update Trust Policies on the CASTOR DLT, ensuring that the trust engineering configuration reflects the latest risk assessment outcomes.

Global TAF Initialization. Once the trust policies are established, the NSO initiates the federation process by specifying the KAFKA topics on which the Global TAF should begin receiving trust reports (`INT_NSO_1`: NSO → Global TAF; content: G-TAF configuration (kafka topics); technology: KAFKA). This step configures the Global TAF to subscribe to the appropriate event streams, enabling it to receive and process trustworthiness claims from the Local TAF agents deployed across the network elements.

Topology and Trust Evidence Collection (Periodical). Throughout the lifecycle of the domain, the following interfaces operate continuously or on a periodic schedule to maintain fresh situational awareness:

- The PCE provides the latest information on the vRouter TNDI topology to the Topology Graph Composition Engine (`INT_PCE_1`: PCE → Topology Graph Composition Engine; content: Topology Graph; technology: KAFKA), ensuring that the enriched topology graph reflects the current state of the network fabric including all SR-TE and Flex-Algo related information.
- Each TNDE (via the TN-DSM) sends Local TAF reports to the Global TAF (`INT_TNDE_1`: TNDE (TN-DSM) → Global TAF; content: TAF reports; technology: HTTP), conveying the node-level trust assessments that capture both static properties (e.g., configuration integrity) and dynamic properties (e.g., runtime behaviour anomalies) of each router.
- In parallel, the TNDE (Local TAF) exposes Local Trust information to the Data Monitor via Prometheus agents (`INT_TNDE_3`: TNDE (Local TAF) → Data Monitor; content: trust information; technology: HTTP), enabling the Monitoring & Analytics Layer to incorporate trust-related telemetry alongside conventional network performance metrics.

- The Global TAF publishes notifications on changes of trust to the Topology Graph Composition Engine (INT_G-TAF_1: Global TAF → Topology Graph Composition Engine; content: notification; technology: KAFKA). These notifications are triggered whenever the Global TAF detects a significant change in the ATL of any node, link, or path, and they drive updates to the trust-related attributes in the enriched Topology Graph.

Monitoring Pipeline (Periodical). The Data Monitor collects pod-based telemetry data (e.g., Prometheus logs) and relays it to the SSLA Monitor (INT_DM_1: Data Monitor → SSLA Monitor; content: TNDI health statistics; technology: HTTP), which uses this information to continuously verify SLA/SSLA compliance for active services. Concurrently, the Data Monitor pushes the same telemetry data to the NSO (INT_DM_2: Data Monitor → NSO; content: TNDI health statistics; technology: HTTP), enabling the Network Service Orchestrator to detect infrastructure-level anomalies—such as pod/container crashes, failed VNF on-boarding attempts, or resource utilisation deviations—and to trigger appropriate adaptive actions at the worker node level.

Infrastructure Updates. The NSO provides Topology Graph Information regarding the infrastructure topology to the Topology Graph Composition Engine (INT_NSO_2: NSO → Topology Graph Composition Engine; content: Topology Graph; technology: KAFKA). This includes updates related to newly provisioned or decommissioned vRouter instances, changes in compute capacity, and modifications to the virtual network fabric managed by Kubernetes.

Topology Graph Dissemination (Periodical or On-Demand). The Topology Graph Composition Engine periodically shares Topology Graph snapshots with interested consumers. In particular, the enriched Topology Graph is shared with the Risk Assessment—and any other authorised component—via INT_TOPO_1 (Topology Graph Composition Engine → Risk Assessment; content: Topology Graph; technology: KAFKA), enabling the Risk Assessment to recalculate risk indices and potentially update the RTLs and Trust Policies accordingly. The Topology Graph is also shared with the TE Policy Engine via INT_TOPO_2 (Topology Graph Composition Engine → TE Policy Engine; content: Topology Graph; technology: KAFKA), providing the TE Policy Engine with the comprehensive network and trust state information required for generating and revising traffic engineering policies.

2.1.2.2 Flow 1: On-boarding of a New Network Element

The On-boarding flow describes the sequence of interactions required to securely enrol a new vRouter into the CASTOR domain. This process encompasses the verification of the TNDE platform, the secure instantiation of all trust extensions, the provisioning of cryptographic material, and the retrieval of trust policies that govern the router's runtime behaviour. Once on-boarding completes successfully, the vRouter's Local TAF begins producing trustworthiness claims that feed into the periodical trust evidence collection pipeline.

Step 1: TNDE and TNDIs On-boarding Request. The on-boarding process is initiated by the Network Function Manager (NSO), which sends an on-boarding request to the TN-DSM of the target vRouter (INT_NSO_3: NSO → TNDE (TN-DSM); content: onboarding and configuration, pub keys, configs for TNDI-SP Channels; technology: HTTP). This request triggers the secure on-boarding the infrastructure elements and the CASTOR trust extensions. The process involves several coordinated actions:

- **TNDE Platform Verification.** The NSO verifies the TNDE platform key against a valid Root of Trust (RoT). This step ensures that the underlying hardware and TEE environment of the vRouter provide the minimum security guarantees required by the CASTOR framework, as specified in D3.1 [6].
- **Secure Instantiation of TNDIs and CASTOR Extensions.** Upon successful platform verification, the TN-DSM instantiates the vRouter and securely launches all TNDIs and CASTOR extensions—

including the Tracing Hub, the Attestation Source, and the FSM Source. During this process, measurements are extracted from all TNDIs and CASTOR extensions (e.g., measurements of the Tracing Hub) in order to execute the *Join Phase*, which creates the necessary cryptographic keys and establishes the correct crypto-policies for the device.

Step 2: vRouter Certificate Publication to DLT. Then, the NSO publishes the on-boarded vRouter's TNDI public key certificate to the CASTOR DLT via the Security Context Broker (INT_NSO_4: NSO (Network Function Manager) → CASTOR DLT; content: Certificate; technology: HTTP). This dissemination enables all CASTOR components to authenticate subsequent communications originating from this vRouter, and establishes the vRouter's identity within the trust infrastructure.

Step 3: vRouter Registration with Global TAF. In parallel, the NSO registers the newly on-boarded vRouter with the Global TAF (INT_NSO_1: NSO → Global TAF; content: G-TAF configuration (kafka topics); technology: KAFKA). This step ensures that the Global TAF subscribes to the relevant KAFKA topics for receiving trust reports from this specific vRouter, thereby incorporating the new network element into the domain-wide trust assessment pipeline.

Step 4: Trust Policy Acquisition and Local TAF Initialization. The TN-DSM retrieves the applicable trust policy from the CASTOR DLT (INT_DLT_1: CASTOR DLT → TN-DSM; content: trust policy; technology: HTTP) and initiates the Local TAF execution. This triggers a cascade of TNDE-internal interactions that configure the complete trust assessment pipeline within the vRouter:

- **4a.** The TN-DSM pushes the vendor- and domain-specific trust policy to the Local TAF (INT_TNDE_TN-DSM_1 TN-DSM → Local TAF; content: trust policy data, e.g., trust model, periodicity; technology: HTTP). The Local TAF decodes the policy and initialises the corresponding trust model.
- **4b.** The Local TAF configures the TNDI trace units and periodically triggers the Tracing Hub (INT_TNDE_L-TAF_1: Local TAF → Tracing Hub; content: trace units configurations; technology: HTTP).
- **4c.** The Tracing Hub configures the CIV trace unit and retrieves the configuration-related traces (INT_TNDE_TR-HUB_1: Tracing Hub → CIV trace unit; content: trace unit configuration and execution; technology: HTTP).
- **4d.** The Tracing Hub configures the eBPF trace unit and retrieves the behaviour-related traces (INT_TNDE_TR-HUB_2: Tracing Hub → eBPF trace unit; content: trace unit configuration and execution; technology: HTTP).
- **4e.** The Tracing Hub provides the configuration-related traces to the Attestation Source (INT_TNDE_TR-HUB_3 Tracing Hub → Attestation Source; content: Attestation source trace configurations; technology: HTTP).
- **4f.** The Tracing Hub provides the behaviour-related traces to the FSM Source (INT_TNDE_TR-HUB_4: Tracing Hub → FSM Source; content: FSM source trace configurations; technology: HTTP).
- **4g.** The Attestation Source pushes the resulting evidence to the Local TAF (INT_TNDE_TR-AS_1: Tracing Hub → Local TAF; content: evidence; technology: HTTP).
- **4h.** The FSM Source pushes the resulting evidence to the Local TAF (INT_TNDE_FSM_1: Tracing Hub → Local TAF; content: evidence; technology: HTTP).
- **4i.** The Tracing Hub provides raw traces to the TN-DSM (INT_TNDE_TR-HUB_5: Tracing Hub → TN-DSM; content: raw traces; technology: HTTP) for anchoring and auditability purposes.

- **4j.** The Local TAF generates the initial trustworthiness claims based on the collected evidence and pushes them to the TN-DSM (INT_TNDE_L-TAF_2: Local TAF → TN-DSM; content: trustworthiness claims; technology: HTTP).

Step 5: Local TAF Evaluations. Once the trust assessment pipeline is fully initialised, the TNDE (TN-DSM) begins sending Local TAF evaluations to the Global TAF (INT_TNDE_1: TNDE (TN-DSM) → Global TAF; content: TAF reports; technology: HTTP). This step marks the transition from the on-boarding flow to the periodical trust evidence collection. From this point onward, the vRouter continuously produces trustworthiness claims that feed into the Global TAF's domain-wide trust assessment.

Step 6: Traces and Trustworthiness Claims to DLT. The TNDE (Local TAF) pushes the TN-DSM output—comprising traces and trustworthiness claims—to the CASTOR DLT (INT_TNDE_2: TNDE (Local TAF) → CASTOR DLT; content: Traces and trustworthiness claims; technology: HTTP). This on-chain anchoring ensures the auditability and immutability of the trust evidence produced during on-boarding, providing a verifiable record of the vRouter's initial trust posture.

2.1.2.3 Flow 2: Service Request

The Service Request flow describes the sequence of interactions triggered when a new service must be provisioned across the CASTOR domain. This flow demonstrates how the Orchestration Layer translates high-level service requirements into trust-aware TE policies that are enforced on the underlying routing infrastructure.

Step 1: Service Request Reception. The flow is initiated when a new service request—including the associated SSLA and path profile requirements—reaches the TE Policy Engine through the Network Controller (INT_NC_2: Network Controller → TE Policy Engine; content: service id, set of trust and network requirements, service endpoints; technology: KAFKA). As described in D5.1 (Engineering Story-XI), this step corresponds to Phase 2 of the service registration process, where the SSLA requirements have already been mapped to a domain-specific Path Profile by the Service Intent Translation & Decomposition Service.

Step 2: Topology Graph Acquisition. Upon receiving the service request, the TE Policy Engine retrieves a fresh snapshot of the enriched Topology Graph from the Topology Graph Composition Engine (INT_TOPO_2: Topology Graph Composition Engine → TE Policy Engine; content: Topology Graph; technology: KAFKA). This snapshot provides the TE Policy Engine with the current network topology, trust attributes (ATLs per node and link), traffic engineering metrics, and the status of existing services and SSLAs—all of which are necessary for informed policy generation.

Step 3: Optimization Request. The TE Policy Engine formulates the optimization requirements and forwards them to the Optimization Engine (INT_TEPE_4: TE Policy Engine → Optimization Engine; content: Topology snapshot, Ingress/egress nodes, path profile; technology: KAFKA). As described in D4.1 [3], the Optimization Engine solves the multi-objective problem of co-enforcing network performance and trust requirements, producing one or more near-optimal recommended paths.

Step 4: Optimization Results. The Optimization Engine returns the near-optimal recommended paths to the TE Policy Engine (INT_OPT_1: Optimization Engine → TE Policy Engine; content: Near-optimal recommended paths satisfying the service request; technology: KAFKA). These results encode the explicit forwarding path(s) that achieve the best trade-off between network metrics and trust assurance levels, as dictated by the agreed SSLA.

Step 5: Start Monitoring. In parallel with the enforcement process, the TE Policy Engine instructs the SSLA Monitor to begin the monitoring process for the new service (INT_TEPE_2: TE Policy Engine → SSLA Monitor; content: service id, set of trust and network requirements, service endpoints, network policy; technology: KAFKA). The SSLA Monitor then initiates three subordinate actions:

- **5a.** The SSLA Monitor begins collecting vRouter network telemetry data—such as performance, jitter, and bandwidth metrics—from the vRouter instances involved in the service path (INT_SSLA_MONITOR_1: SSLA Monitor → vRouter instances; content: TNDI network statistics; technology: HTTP). This ensures that network-level compliance is continuously tracked.
- **5b.** The SSLA Monitor registers or unregisters composite trust evaluations with the Global TAF (INT_SSLA_MONITOR_2: SSLA Monitor → Global TAF; content: notification; technology: KAFKA), subscribing to notification events that will allow it to receive trust-related updates relevant to the monitored service.
- **5c.** The SSLA Monitor communicates the new service requirements for topology resources to the NSO (INT_SSLA_MONITOR_3: SSLA Monitor → NSO; content: service id, set of trust and network requirements, service endpoints; technology: HTTP), enabling the NSO to ensure that sufficient compute and network resources remain allocated to support the service.

Step 6: Policy Enforcement. The TE Policy Engine translates the optimised paths into enforceable configurations and coordinates their deployment through two complementary mechanisms, as described in D5.1 [5] (Section 4.3):

- **6a. Explicit Path Enforcement (PCE-driven).** For the primary path derived from the Optimization Engine, the TE Policy Engine sends the enforceable policy to the PCE (INT_TEPE_5: TE Policy Engine → PCE; content: Policy Name, Explicit Label Stack (Array), IP Endpoints; technology: KAFKA). The PCE then enforces the PCE configuration on the Cisco IOS XR PCE vRouter instance (INT_PCE_2: PCE → PCE vRouter instance; content: Updated PCE SR TE configuration; technology: gRPC). Finally, the PCE vRouter instance installs the PCE-initiated SR-TE policy on the Ingress vRouter via the Path Computation Element Communication Protocol (INT_PCE_4: PCE vRouter instance → Ingress vRouter; content: PCE initiated SR-policy; technology: PCEP over TCP). This mechanism ensures that traffic is steered along the precise path calculated by the Optimization Engine, satisfying both network and trust constraints.
- **6b. Dynamic Path / Restoration Strategy (Flex-Algo).** As a complementary fallback mechanism, the TE Policy Engine notifies the Network Controller of the revised topology graph snapshot for updating the Flex-Algo configuration (INT_TEPE_3: TE Policy Engine → Network Controller; content: revised topology graph snapshot for updating flex algo configuration; technology: KAFKA). The Network Controller then enforces the Flex-Algo affinity bit updates on the PCE (INT_NC_1: Network Controller → PCE; content: Router id, Interface name, Color; technology: KAFKA). Subsequently, the PCE propagates the updated Flex-Algo configuration to all participating vRouters (INT_PCE_3: PCE → Any vRouter; content: FlexAlgo configuration; technology: gRPC). This mechanism provides a rapid restoration strategy: if the primary explicit path is disrupted, traffic automatically falls back to a Flex-Algo-computed path that still respects the trust-enhanced affinity constraints (e.g., excluding links labelled as LOW_TRUST). As detailed in D5.1 (Section 7.4), this approach bridges the gap between an SSLA violation event and the availability of fresh recommendations from the Optimization Engine.

Step 7: Trust Awareness Notification. Throughout the service lifecycle, the TE Policy Engine exposes service status updates to external authorised entities—including the Service Provider—via the Trust Awareness API (INT_TEPE_1: TE Policy Engine → External Service Provider; content: trust-related information; technology: KAFKA). In case of an SSLA violation, the SSLA Monitor generates SSLA Alarms that are forwarded to the TE Policy Engine (INT_SSLA_MONITOR_4: SSLA Monitor → TE Policy Engine; content: SSLA Alarms; technology: KAFKA), which can then trigger the reactive phase (re-optimization and re-enforcement). In parallel, relevant trust and compliance information is exposed through the Trust Awareness API, enabling the Service Provider to maintain visibility into the trust posture of the domain

and the status of the established services. The event-driven communication is facilitated via the KAFKA distributed event-streaming platform, ensuring scalable and low-latency dissemination of status updates.

Step 8: DLT Anchoring. Finally, the TE Policy Engine stores information on the service status updates SSLA (e.g., violations and/or the deployed SSLAs) to the CASTOR DLT (`INT_TEPE_6`: TE Policy Engine → CASTOR DLT; content: Updates on SSLA violation, Deployed SSLAs; technology: HTTP). This on-chain anchoring ensures auditability and immutability of all service-level decisions and compliance events, supporting both real-time auditability and post-incident forensic analysis. As described in D5.1 (Section 5.3.3, Flow 3), the SSLA content is committed through the Computation Worker enclave, which validates the data integrity prior to on-chain storage.

2.1.3 CASTOR Interfaces

A breakdown of the types of interfaces used for communication between CASTOR components is provided in Table 2.2. These interfaces, which facilitate interaction among various CASTOR components, are categorized as follows:

- **REST API**: This interface facilitates data exchange between a sender and a receiver. Within the CASTOR framework, it is primarily used to support: (a) the interactions between the in-router TNDE components and (b) the TNDI-SP channels - the interactions between the TNDE and the Orchestrator (Global TAF) or the TNDE.
- **KAFKA Pub/Sub**: The KAFKA interface supports scalable, low-latency exchange of large data volumes via a publish–subscribe model, enabling real-time stream processing by multiple consumers. In CASTOR, it is primarily used within the orchestration layer.
- **SSH/NETCONF/GNMI/CLI/gRPC/PCEP**: Other type of interfaces to support several interactions between the Orchestration Layer and the vRouters.

Table 2.2 summarizes the CASTOR components and their interfaces. The symbol ↑ indicates communication from the row component to the column component, while ↓ denotes the reverse direction. The presence of both symbols signifies bidirectional communication.

CASTOR Components	PCE	Risk Assessment	Global TAF	Optimization Engine	TE Policy Engine	CASTOR DLT	NSO	Network Controller	Topology Graph Composition Eng	SSLA Monitor	Data Monitor	TN-DSM	Tracing Hub	Trace Units	Attestation Source	FSM Source	Local TAF
PCE					↓			↓	↑								
Risk Assessment						↑			↓								
Global TAF							↓		↑	↓		↓					
Optimization Engine					⇅												
TE Policy Engine	↑			⇅		↑		⇅	↓	⇅							
CASTOR DLT		↓			↓		↓					⇅					
NSO			↑			↑			↑	↓	↓	↑					
Network Controller	↑				⇅												
Topology Graph Composition Eng	↓	↑	↓		↑			↓									
SSLA Monitor			↑		⇅			↑			↓						
Data Monitor								↑		↑							↓
TN-DSM			↑			⇅		↓									⇅
Tracing Hub														↑	↑	↑	↓
Trace Units													↓				
Attestation Source													↓				↑
FSM Source													↓				↑
Local TAF											↑	⇅	↑		↓	↓	

Table 2.2: CASTOR Communication Interfaces

Table 2.3 lists all CASTOR Interfaces along with a description, the exchanged content and the used technology (e.g., HTTP or KAFKA). Table 2.3 includes in the end also the internal TNDE interfaces.

ID	Source	Target	Description	Content (WIP)	Technology (WIP)
INT_NC_1	Network Controller	PCE	Enforce flex algo affinity bit update	Router id; Interface name; Color	KAFKA
INT_NC_2	Network Controller	TE Policy Engine	Service request including SSLA and path profile required	service id; set of trust and network requirements; service endpoints	KAFKA
INT_TEPE_1	TE Policy Engine	External Service Provider	Trust Awareness API	Trust-related information and SSLA violations (provided via the INT_SSLA_MONITOR_4)	KAFKA
INT_TEPE_2	TE Policy Engine	SSLA Monitor	Start monitoring process for a new service	service id; set of trust and network requirements; service endpoints; network policy	KAFKA
INT_TEPE_3	TE Policy Engine	Network Controller	Notification of a change in the topology graph	reviced topology graph snapshot for updating flex algo configuration	KAFKA
INT_TEPE_4	TE Policy Engine	Optimization Engine	Optimization requirements	Topology snapshot; Ingress/egress nodes; path profile	KAFKA
INT_TEPE_5	TE Policy Engine	PCE	Enforce a new explicit path	Policy Name; Explicit Label Stack (Array); IP Endpoints	KAFKA
INT_TEPE_6	TE Policy Engine	CASTOR DLT	Service status updates (e.g., SSLA violation; SSLA deployment)	Notifications on service status updates	HTTP
INT_OPT_1	Optimization Engine	TE Policy Engine	Optimization results	Near-optimal recommended paths satisfying the service request	KAFKA
INT_PCE_1	PCE	Topology Graph Composition Engine	Topology Graph Information (TNDI topology)	Topology Graph	KAFKA
INT_PCE_2	PCE	PCE vRouter instance	Enforce PCE configuration to the Cisco IOS XR PCE vRouter	Updated PCE SR TE configuration	gRPC
INT_PCE_3	PCE	Any vRouter	Enforce flex algo affinity bit update	FlexAlgo configuration	gRPC
INT_PCE_4	PCE vRouter instance	Ingress vRouter	PCE-initiated SR-TE policies via PCEP	PCE initiated SR-policy	PCEP (over TCP)
INT_G-TAF_1	Global TAF	Topology Graph Composition Engine	Notifications on changes of trust	notification	KAFKA
INT_DM_1	Data Monitor	SSLA Monitor	Collect pod-based telemetry data (e.g., prometheus logs)	TNDI health statistics	HTTP
INT_DM_2	Data Monitor	NSO	Push the collected pod-based telemetry data (e.g., prometheus logs)	TNDI health statistics	HTTP
INT_SSLA-MONITOR_1	SSLA Monitor	vRouter instances	Collect vRouter network telemetry data (e.g., performance, jitter, bandwidth metrics)	TNDI network statistics	HTTP
INT_SSLA-MONITOR_2	SSLA Monitor	Global TAF	Register/Un-register composite trust evaluations	notification	KAFKA
INT_SSLA-MONITOR_3	SSLA Monitor	NSO	Service requirements for topology resources	service id; set of trust and network requirements; service endpoints	HTTP
INT_SSLA-MONITOR_4	SSLA Monitor	TE Policy Engine	In case of service degradations (e.g., an SSLA violation) the SSLA Monitor generates SSLA Alarms in order to notify the TE Policy Engine.	SSLA Alarms	KAFKA
INT_RA_1	Risk Assessment	CASTOR DLT	Push Trust Policy Metadata On-chain	Trust Model Template; RTL	HTTP
INT_TOPO_1	Topology Graph Composition Engine	Risk Assessment	Share Topology Graph Notifications	Topology Graph	KAFKA
INT_TOPO_2	Topology Graph Composition Engine	TE Policy Engine	Share Topology Graph snapshot	Topology Graph	KAFKA
INT_TNDE_1	TNDE (TN-DSM)	Global TAF	Send Local TAF evaluations	TAF reports	HTTP
INT_TNDE_2	TNDE (TN-DSM)	CASTOR DLT	TN-DSM output	Traces and trustworthiness claims	HTTP
INT_TNDE_3	TNDE (TN-DSM)	Data Monitor	Collect Local Trust information via Prometheus agents	trust information	HTTP
INT_NSO_1	NSO	Global TAF	Start the federation process (Specify the kafka.topics that the G-TAF should start sending reports)	G-TAF configuration (kafka topics)	KAFKA

INT_NSO_2	NSO	Topology Graph Composition Engine	Topology Graph Information (infrastructure topology)	Topology Graph	KAFKA
INT_NSO_3	NSO (Network Function Manager)	TNDE (TN-DSM)	Onboard request (Multiple requests may be needed) 1) Trigger new onboarding 2) Establish control/data TNDI-SP channels	onboarding and configuration, pub keys, configs for TNDI-SP Channels	HTTP
INT_NSO_4	NSO (Network Function Manager)	CASTOR DLT	Disseminate on-boarded vRouter (certificate of the TNDI public key)	Certificate	HTTP
INT_DLT_1	CASTOR DLT	TNDE (TN-DSM)	Get Trust Policy Metadata	trust policy	HTTP
INT_TNDE_TN-DSM_1	TN-DSM	Local TAF	push vendor and domain specific trust policy	trust policy data (e.g., trust policy model, periodicity, etc.)	HTTP
INT_TNDE_L-TAF_1	Local TAF	Tracing Hub	TNDI trace units configurations	trace units configurations	HTTP
INT_TNDE_L-TAF_2	Local TAF	TN-DSM	Push trustworthiness claims	trustworthiness claims	HTTP
INT_TNDE_TR-HUB_1	Tracing Hub	CIV trace unit	trace unit configuration and execution	trace unit configuration and execution	HTTP
INT_TNDE_TR-HUB_2	Tracing Hub	eBFP trace unit	trace unit configuration and execution	trace unit configuration and execution	HTTP
INT_TNDE_TR-HUB_3	Tracing Hub	Attestation Source	TNDI trace configuration	Attestation source trace configurations	HTTP
INT_TNDE_TR-HUB_4	Tracing Hub	FSM Source	TNDI trace configuration	FSM source trace configurations	HTTP
INT_TNDE_TR-HUB_5	Tracing Hub	TN-DSM	raw traces	raw traces	HTTP
INT_TNDE_TR-AS_1	Tracing Hub	Local TAF	traced evidence	evidence	HTTP
INT_TNDE_FSM_1	Tracing Hub	Local TAF	traced evidence	evidence	HTTP

Table 2.3: List of CASTOR Interfaces

2.2 KAFKA Topics

This section details the interaction topics defined for the CASTOR components via the Kafka distributed event-streaming platform. These Kafka topics serve as the communication buses provided by the Data Broker at the CASTOR Orchestration Layer, interconnecting all orchestration-level artifacts to enable seamless interaction. This integrated communication framework facilitates the end-to-end provisioning and enforcement of trust-aware Traffic Engineering (TE) policies. As illustrated in Figure 2.4, these interfaces represent the core connectivity of the system, while Table 2.4 provides a comprehensive list of all defined Kafka topics alongside the relevant interface identifiers (defined in Table 2.3).

To ensure a structured and scalable messaging architecture, the CASTOR framework utilizes a standardized naming convention for its Kafka topics. The ”<component_name>” notation designates a primary topic acting as a dedicated endpoint for a specific component. This creates a ”listener” pattern, allowing any CASTOR artifact to publish information to that component through a single, well-defined entry point. Conversely, the ”<sender_component>.<purpose>” notation is used for targeted kafka buses, typically reserved for streaming specific data types or telemetry from a source component to specialized consumers.

As highlighted in Table 2.3, this initial set of Kafka topics constitutes a work in progress and will be further consolidated, finalized, and documented in D6.2. Nevertheless, the specification provided here demonstrates the maturity of the technical definitions and the ongoing integration activities across the CASTOR technical artifacts.

Table 2.4: CASTOR Integrated Framework Kafka Topics

Kafka Topic	Publishers	Subscribers	Relevant interfaces
tepe	Network Controller; SSLA Monitor; Topology Graph Composition Engine	TE Policy Engine	INT_NC_2; INT_SSLA-MONITOR_4; INT_TOPO_2
tepe.ssla-monitor-request	TE Policy Engine	SSLA Monitor	INT_TEPE_2
tepe.flex-algo-request	TE Policy Engine	Network Controller	INT_TEPE_3
tepe.optimization-request	TE Policy Engine	Optimization Engine	INT_TEPE_4
optimization.results	Optimization Engine	TE Policy Engine	INT_OPT_1
tepe.conf-decision	TE Policy Engine	PCE	INT_TEPE_5
nc.conf-decision	Network Controller	PCE	INT_NC_1
topology-graph-ce	Global TAF; PCE; NSO	Topology Graph Composition Engine	INT_PCE_1; INT_G-TAF_1; INT_NSO_2
global-taf	SSLA Monitor; NSO	Global TAF	INT_SSLA-MONITOR_2; INT_NSO_1
risk-assessment-engine	Topology Graph Composition Engine	Risk Assessment Engine	INT_TOPO_1
trust-awareness-api	TE Policy Engine; SSLA Monitor	External Service Provider	INT_TEPE_1; INT_SSLA-MONITOR_4

Chapter 3

Proof-of-Concept Playground

This chapter introduces the CASTOR Proofs of Concept (PoCs) and the experimentation framework used to initially validate the project's technical approach. Evaluating CASTOR's functionalities through these early PoCs will provide the critical insights needed to fine-tune each technical component before deploying the integrated framework into realistic testbeds for full-scale use case evaluation. The following experimentation plan comprises of complementary PoC scenarios, aiming to demonstrate and evaluate specific functionalities of the CASTOR integrated framework.

The five Proofs of Concept (PoCs) designed for the CASTOR project are purposefully structured to capture the complete lifecycle of trust-aware network traffic engineering. While each PoC can be executed and evaluated independently, they are fundamentally complementary by design. They span a diverse set of operations, from granular, device-level operational monitoring to topology-wide trust evaluations that unlock trust-aware routing decisions for both seamless deployment of routing and restoration policies. This diversity ensures that all foundational capabilities (i.e., secure data collection, trust assessment, policy enforcement, continuous telemetry, and incident reaction) of the CASTOR framework are validated before deploying the CASTOR integrated framework in the context of the use case scenarios.

To demonstrate the operational lifecycle of the CASTOR integrated framework (as explained in detail in Chapter 2, the PoCs are organized as follows:

- **PoC 1: Trust Network Device Extension for supporting the continuous Trust Assessment:** Section 3.2 focuses on the foundational data layer of the CASTOR in-device trust enablers which allow the secure monitoring, collection and reporting of trustworthiness evidence and claims towards the orchestration layer. Ths incorporates the validation of how critical router behaviours are monitored, how in-device Trust Sources process these traces, and how novel cryptography secures the sharing of assurance claims.
- **PoC 2: Trust Engineering Process:** Section 3.3 elevates this device-level data by demonstrating how federated Trust Assessment Framework (TAF) agents aggregate claims to produce actionable trust scores, enabling the Global TAF to formulate topology-wide trust characterizations.
- **PoC 3: Trust-Aware Traffic Engineering Provisioning:** Section 3.4 bridges assessment and action, showcasing how the Path Computation Element translates multi-objective optimizations into the actual enforcement of combined trust and network policies within the forwarding plane.
- **PoC 4: Telemetry and SSLA Monitoring:** Section 3.5 introduces continuous observation, enriching the trust pipeline with network-based and host-based telemetry to actively monitor Secure Service Level Agreements (SSLAs) and swiftly identify Service Level Objective (SLO) violations.
- **PoC 5: TE Restoration Strategies:** Section 3.6 closes the operational loop by demonstrating the reactive capabilities of the framework, detailing how the system responds to network-driven or trust-driven events to mitigate SSLA violations and restore optimal service.

By seamlessly gluing these five PoCs together, we transition from individual component validation to the realization of the fully integrated CASTOR framework. The lifecycle is highly interdependent: TNDI operational assurances (PoC 1) generate the trust characterization of the forwarding plane (PoC 2) that dictates routing provisioning (PoC 3), which is then continuously monitored at the CASTOR Orchestration Layer (PoC 4) to trigger automated restorations when necessary (PoC 5). Collectively, they form the core operational functionalities of the CASTOR architecture, providing a robust, unified foundation that will be systematically evaluated against CASTOR's real-world use case scenarios.

The specification for each PoC below incorporates an initial evaluation strategy based on specific PoC-related KPIs. These PoC KPIs highlight the key dimensions that will be validated during the two releases of the CASTOR integrated framework. While several PoC KPIs map directly to the technical KPIs of the individual artifacts defined in D2.1 [4], the integrated nature of the PoCs offers a unique opportunity to validate multi-artifact functionalities. Consequently, the evaluation dimensions extend beyond the strict metrics of D2.1, focusing on CASTOR's performance under specific scenarios across the forwarding plane's operational lifecycle. For example, although the KPIs for PoC 5 (see Section 3.6) overlap with those of other PoCs, they target a distinct operational situation: restoring a provisioned service following a network- or trust-driven event in the topology.

3.1 Common playground network topology

Before delving into the specification of the PoC narratives, it is essential to frame the validation testbed that will act as a common platform that will allow us to yield comparable results under a given set of conditions. To this end, CASTOR introduces the PoC playground, a common simulation-driven environment where CASTOR trust enablers are first integrated and evaluated before their deployment on the use case testbeds. The PoC playground provides a controlled and programmable infrastructure that can be rapidly reconfigured to emulate different network topologies, service workloads, and operational risk conditions. This enables systematic experimentation over programmable infrastructures and supports the progressive validation of CASTOR mechanisms for trusted path establishment under varying trust requirements. Within this environment, the end-to-end operational loop of the CASTOR framework can be validated within a single emulated domain[1].

As highlighted below, this initial PoC playground topology is designed to validate all core functionalities of the CASTOR framework. For example, the emulated hierarchical vRouter topology lacks direct adjacency between ingress and egress routers. This requires the ingress vRouter to consult the CASTOR Path Computation Element (PCE) for the network policies needed to reach the intended destination (PCE capabilities are detailed in D5.1 [5]). Furthermore, the playground is highly reproducible, enabling diverse PoCs with distinct deployment requirements to instantiate it while maintaining consistent forwarding plane conditions. For instance, PoC 1 (Section 3.2) requires vRouters (TNDIs) to be deployed on infrastructure supporting hardware-based Root-of-Trust (RoT). In the second experimental release, this will allow the TNDE platform—which houses the CASTOR Trusted Computing Base—to bind to a hardware RoT (e.g., Intel SGX) to deliver elevated operational assurances.

The common playground topology is a lab built with ten Cisco IOS XRd control-plane nodes, running Segment Routing (SR) with MPLS (SR-MPLS). The playground demonstrates hierarchical IS-IS, PCE-driven SR-TE, and BGP service Provider Edges (PEs). Recommended specifications for running the control-plane topology in a VM are: Ubuntu 24.04 with docker and docker-compose, minimum 8 vCPUs, 32 GB RAM, and an official Cisco XRd control-plane image file. It has to be noted that docker and

[1] For the validation of the PoC narratives in the first release, the PoC playground is envisioned to be a routing topology within a single administrative domain. Moving towards the second release, CASTOR will investigate the expansion of the PoC playground to emulate a multi-domain environment. This will enable the validation of the cross-domain capabilities of CASTOR before deploying them in the use case testbeds as part of the second version of the CASTOR integrated framework.

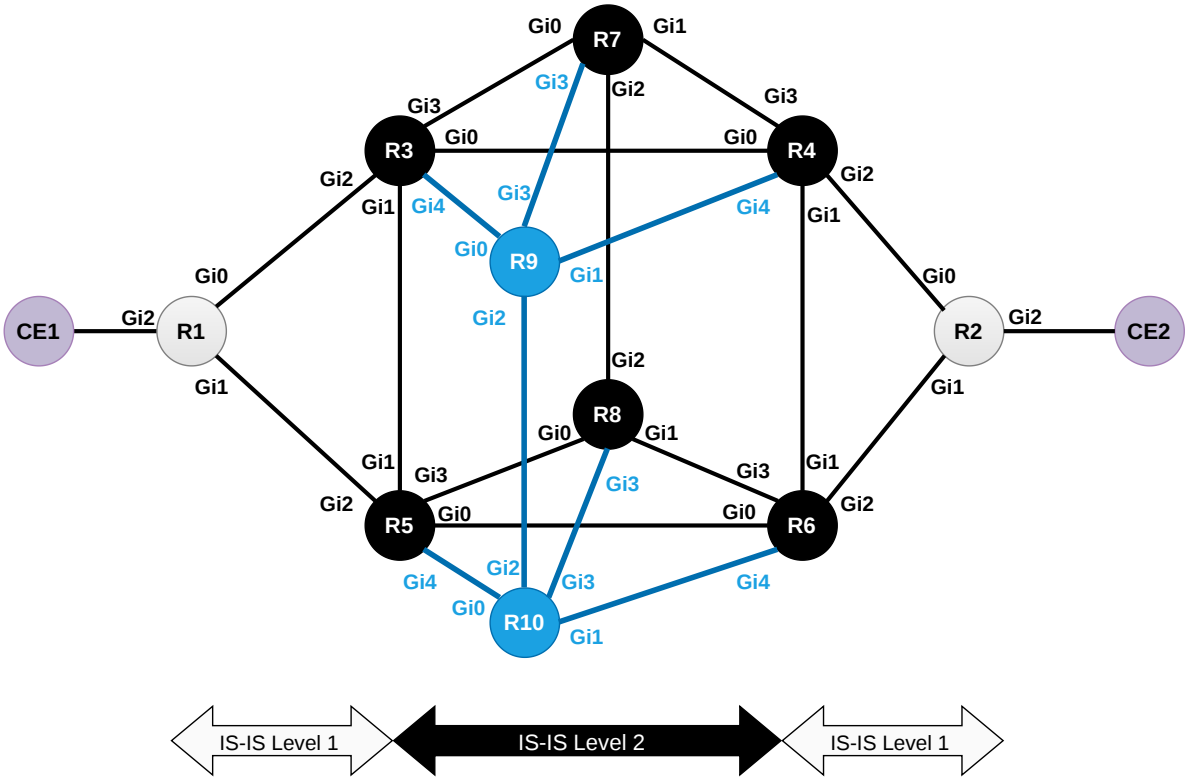

Figure 3.1: Control-plane vRouter hierarchical topology

docker-compose apply only to the PoCs, where the Network Service Orchestrator (e.g., K8s resource orchestration) is not present. As depicted in Figure 3.1, the network is divided into two IS-IS Level 1 domains, interconnected by a Level 2 backbone, providing end-to-end reachability and SR-MPLS label forwarding.

R1 and R2 act as Provider Edge (PE) routers, connecting Customer Edge (CE) devices CE1 and CE2 to the MPLS core. R7 operates as a centralized Path Computation Element (PCE); it receives full topology information from R3 and R4 BGP link-state (BGP-LS) address family. Any topology events, including link affinity color changes, will be propagated from IS-IS to BGP link-state, eventually reaching R7. If any paths need updating, R7 will send those updates to the clients (R1 and R2). R8 and R9 are BGP Route Reflectors (RRs) for IPv4 VPN, IPv6 VPN, and EVPN address families; they will distribute service routes between R1 and R2.

Each XRd device in the topology has four flex-algos preconfigured: 128-131. Each one uses a different metric: IGP cost, TE metric, bandwidth and delay. Each router will advertise a node SID for every flex-algo, including the base-topology (algo ID 0). The node SID configuration is present in the “router isis” config, under the Loopback0 IPv4 address family. Since IOS-XR Segment Routing Global Block (SRGB) starts at 16000, a configured index of 101 will translate to MPLS label 16101 (see Listing 3.1).

Listing 3.1: Node SIDs configured in router R1 for the flex-algo and the base topologies

```
1
2  RP/0/RP0/CPU0:xrd-1#show isis segment-routing label table | in Loopback
3  16101 100.100.100.101/32 SPF Loopback0
4  16201 100.100.100.101/32 FA 128 Loopback0
5  16301 100.100.100.101/32 FA 129 Loopback0
6  16401 100.100.100.101/32 FA 130 Loopback0
7  16501 100.100.100.101/32 FA 131 Loopback0
8
9  RP/0/RP0/CPU0:xrd-1#show run router isis 1 interface loopback 0
10 router isis 1
```

```
 interface Loopback0
  passive
  address-family ipv4 unicast
   prefix-sid index 101
   prefix-sid algorithm 128 index 201
   prefix-sid algorithm 129 index 301
   prefix-sid algorithm 130 index 401
   prefix-sid algorithm 131 index 501
  !
 !
!
```

There is no IGP route for PE2 in PE1's Routing Information Base (RIB), effectively isolating the service layer from the base topology. BGP next-hop resolution is instead performed via SR-TE policies, triggered by the BGP Color Extended Community. Since PE1 and PE2 do not have full visibility to each other, they will ask the PCE for a path comprised of a label stack to use in the data plane. When a path is received from the PCE, it goes through internal validation checks, and if it passes, it is programmed in the Forwarding Information Base (FIB/CEF) steering the traffic into the Segment Routing tunnel. Listing 3.2 showcases a sample CLI output from PE1 (R1):

Listing 3.2: BGP next hop configuration in router R1

```

RP/0/RP0/CPU0:xrd-1#show route 100.100.100.102

Routing entry for 0.0.0.0/1
  Known via "static", distance 1, metric 0 (connected)
  Installed Feb 17 13:55:23.313 for 5d19h
  Routing Descriptor Blocks
    directly connected, via Null0
      Route metric is 0
  No advertising protos.


RP/0/RP0/CPU0:xrd-1#show bgp vrf 100 labels
BGP VRF 100, state: Active
BGP Route Distinguisher: 1.1.1.1:100
VRF ID: 0x60000001
BGP router identifier 100.100.100.101, local AS number 100
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000001 RD version: 54
BGP main routing table version 62
BGP NSR Initial initsync version 17 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network Next Hop Rcvd Label Local Label
Route Distinguisher: 1.1.1.1:100 (default for vrf 100)
Route Distinguisher Version: 54
*> 1.10.1.1/32 0.0.0.0 nolabel 24009
*>i1.10.1.2/32 100.100.100.102 C:100
                                  24009 nolabel
```

```
*> 10.1.1.0/24 0.0.0.0 nolabel 24009
*>i10.3.1.0/24 100.100.100.102 C:100
                                  24009 nolabel

Processed 4 prefixes, 4 paths


RP/0/RP0/CPU0:xrd-1#show cef vrf 100 1.10.1.2
1.10.1.2/32, version 12, internal 0x5000001 0x30 (ptr 0x875b4600) [1], 0x0 (0x0), 0x208
    (0x89496768)
 Updated Feb 17 13:58:37.639
 Prefix Len 32, traffic index 0, precedence n/a, priority 3
  gateway array (0x8868ec98) reference count 2, flags 0x2038, source rib (7), 0 backups
               [1 type 1 flags 0x48441 (0x894d1640) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Feb 17 13:58:37.639
 LDI Update time Feb 17 13:58:37.640
   via local-label 24026, 5 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x8765b630 0x0]
    recursion-via-label
    next hop VRF - 'default', table - 0xe0000000
    next hop via 24026/0/21
     next hop srte_c_100_ep_100.100.100.102 labels imposed {ImplNull 24009}

    Load distribution: 0 (refcount 1)

    Hash OK Interface Address
    0 Y recursive 24026/0
```

SR-TE policy checks can be done with ping/traceroute between PE Virtual Routing and Forwarding (VRF) loopbacks. The SR-TE label stack is shown in Listing 3.3.

Listing 3.3: Reachability validation from R2(PE2) to R1(PE1)

```

RP/0/RP0/CPU0:xrd-2#traceroute vrf 200 1.20.1.1
Type escape sequence to abort.
Tracing the route to 1.20.1.1

 1 100.102.106.106 [MPLS: Labels 16204/16203/16205/16201/24015 Exp 0] 3 msec 2 msec 2 msec
 2 100.104.106.104 [MPLS: Labels 16203/16205/16201/24015 Exp 0] 2 msec 2 msec 2 msec
 3 100.103.104.103 [MPLS: Labels 16205/16201/24015 Exp 0] 2 msec 2 msec 2 msec
 4 100.101.103.101 [MPLS: Labels 16205/16201/24015 Exp 0] 2 msec 2 msec 2 msec
 5 100.101.105.105 [MPLS: Labels 16201/24015 Exp 0] 2 msec 2 msec 2 msec
 6 100.101.105.101 4 msec * 6 msec

RP/0/RP0/CPU0:xrd-2#show segment-routing traffic-eng policy color 200
SR-TE policy database
---------------------

Color: 200, End-point: 100.100.100.101
  Name: srte_c_200_ep_100.100.100.101
  Status:
    Admin: up Operational: up for 5d20h (since Feb 17 13:57:30.361)
  Candidate-paths:
```

```
    Preference: 300 (configuration) (active)
      Name: PC200
      Requested BSID: dynamic
      Constraints:
        Protection Type: protected-preferred
        Maximum SID Depth: 10
      Explicit: segment-list MYSEGLIST2 (valid)
        Weight: 300, Metric Type: TE (IANA PCEP/IGP: 2/2)
          SID[0]: 16206 [Prefix-SID, 100.100.100.106]
          SID[1]: 16204
          SID[2]: 16203
          SID[3]: 16205
          SID[4]: 16201
    Preference: 200 (BGP ODN) (inactive) (shutdown)
      Name: bgp_c_200_ep_100.100.100.101_discr_200
      Requested BSID: dynamic
      Constraints:
        Prefix-SID Algorithm: 128
        Protection Type: protected-preferred
        Maximum SID Depth: 10
      Dynamic (inactive)
        Metric Type: NONE (IANA PCEP: 0), Path Accumulated Metric: 0
    Preference: 100 (configuration) (inactive)
      Name: PC200
      Requested BSID: dynamic
      PCC info:
        Symbolic name: cfg_PC200_discr_100
        PLSP-ID: 2
      Constraints:
        Prefix-SID Algorithm: 128
        Protection Type: protected-preferred
        Maximum SID Depth: 10
      Dynamic (pce 100.100.100.107) (inactive)
        Metric Type: TE (IANA PCEP/IGP: 2/2), Path Accumulated Metric: 3003
          SID[0]: 16206
          SID[1]: 16205
          SID[2]: 16201
    Preference: 100 (BGP ODN) (inactive)
      Name: bgp_c_200_ep_100.100.100.101_discr_100
      Requested BSID: dynamic
      PCC info:
        Symbolic name: bgp_c_200_ep_100.100.100.101_discr_100
        PLSP-ID: 6
      Constraints:
        Prefix-SID Algorithm: 128
        Protection Type: protected-preferred
        Maximum SID Depth: 10
      Dynamic (pce 100.100.100.107) (inactive)
        Metric Type: TE (IANA PCEP/IGP: 2/2), Path Accumulated Metric: 3003
          SID[0]: 16206
          SID[1]: 16205
          SID[2]: 16201
  Attributes:
    Binding SID: 24022
```

```
    Forward Class: Not Configured
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
    Max Install Standby Candidate Paths: 0
    Path Type: SRMPLSv4
```

Connectivity checks can also be run by attaching to the CEs (Alpine containers). These CEs correspond to the UC infrastructure (see Chapter 4). To test across other flex-algos, VRFs can be changed on R1/R2, while keeping the previous IP subnets (see Listing 3.4).

Listing 3.4: Reachability validation from the application entrypoints (via the ”source” CE)

```

$ docker attach source
/ # traceroute 10.3.1.3
traceroute to 10.3.1.3 (10.3.1.3), 30 hops max, 46 byte packets
 1 xrd-1.100_v2_10x_srmpls_source-xrd-1 (10.1.1.3) 1.796 ms 0.347 ms 0.379 ms
 2 100.101.103.103 (100.101.103.103) 4.083 ms 2.764 ms 1.939 ms
 3 100.103.104.104 (100.103.104.104) 3.679 ms 3.271 ms 1.863 ms
 4 100.102.104.102 (100.102.104.102) 2.765 ms 2.439 ms 1.736 ms
 5 10.3.1.3 (10.3.1.3) 1.850 ms 2.236 ms 1.911 ms
```

SSH, NETCONF, and gNMI/gRPC access is allowed over the XRd management interfaces. If there is an IP range overlap between the management LAN and the test environment, IP addresses need to be changed in the XRd initial configs, and the docker-compose file (see Listing 3.5).

Listing 3.5: Management Plane connectivity in all vRouters in the topology (examples on R1)

```

$ ssh cisco@172.30.0.101
(cisco@172.30.0.101) Password:
Last login: Mon Feb 23 12:02:53 2026 from 172.30.0.1
RP/0/RP0/CPU0:xrd-1#exit

$ ssh cisco@172.30.0.101 -p 830 -s netconf
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
</capabilities>
</hello>]]>]]>
#143
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
 <get-config>
  <source>
   <running/>
  </source>
 </get-config>
</rpc>
##

$ bash -c "$(curl -sL https://get-gnmic.openconfig.net)"

```

```
27 $ gnmic -a 172.30.0.101:57400 -u <USERNAME> -p <PASSWORD> --skip-verify subscribe --path
     "openconfig-interfaces:interfaces" --mode once
```

3.2 PoC 1: Trust Network Device Extension for supporting the continuous Trust Assessment

The goal of this PoC is to validate the evidence collection and monitoring process as part of trusted path establishment paradigm. This constitutes a foundational layer for the entire CASTOR framework as in this PoC we provide the trust extensions that enable all other operations: from Trust Assessment Framework to the overall specification of trust-aware TE policies. Table 3.1 outlines the PoC KPIs, which constitute key dimensions for validating and evaluating these CASTOR functionalities in a unified operational narrative. As observed in these KPIs, the evaluation sequence mirrors the operational lifecycle of the TNDE, which was introduced during the onboarding description in Chapter 2.

The PoC lifecycle initiates with the onboarding of both the CASTOR TNDE within an infrastructure element and the corresponding TNDIs on top of it. Through this phase in the PoC, we are able to validate the secure launch of the CASTOR TCB and the provisioning of the necessary TNDI keys, which are critical for securely sharing router-specific trustworthiness evidence and facilitating interactions with the rest of the CASTOR framework (e.g., securely logging TNDI-specific traces to the CASTOR DLT). Following the successful onboarding of all elements, the validation shifts to the configuration of the TNDE artifacts. The newly admitted TNDE interacts with the CASTOR DLT to fetch the designated Trust Policy. Using this policy, the Local TAF agent systematically configures the available Trust Sources and provisions the underlying Trace Units required for continuous evidence collection. Up to this point, the PoC allows the validation of the end-to-end CASTOR protocol for enabling a new TNDI instance (e.g., a new vRouter) to actively participate in the topology, both in terms of providing forwarding capabilities within the trusted topology and participating in the overall trust engineering process.

In addition. a core dimension of this PoC involves evaluating the overhead associated with the operation of the available Trust Sources. For the Attestation Source, the validation focuses on the secure provisioning and management of the necessary attestation keys. In parallel, for the FSM Source, the evaluation targets the training and development of the FSM device model, specifically measuring its coverage and accuracy with respect to the critical monitored network functions. Within this context, the underlying Trace Units remain essential for observing the runtime behaviour and configuration integrity of the TNDIs. Building upon the developments in D3.2 [7], this PoC demonstrates how traces collected via eBPF and kernel extensions are essential for the operation of the Trust Sources. By processing these traces, the Trust Sources can validate critical security properties, such as configuration and behavioural runtime integrity. This validation, in turn, provides the foundational claims required by the CASTOR TAF (see PoC 2 in Section 3.3) to formulate evidence-based ATL trust opinions. The evaluation in this PoC KPI relates to the performance overhead of these Trace Units and the overall monitoring process across different TNDI modalities and isolation levels, such as containerized versus VM-based TNDI deployments.

Apart from the overall Local TAF operational workflow, this PoC evaluates the runtime performance and overhead of the novel attestation schemes detailed in D3.2 [7]. The first aspect examines how the initial flat-hierarchy attestation of all TNDE applications (by the TN-DSM) can be optimized through a layered attestation scheme (see D3.2[7] for details on the proposed scheme). By adopting a hierarchical approach, each application layer only observes its immediate child (e.g., the bootloader attests the attestation agent, which in turn attests the Local TAF agent). The second aspect validates the composite attestation mechanisms designed to evaluate higher-level claims beyond a single node, verifying the assurance level of deployed segments or entire paths of TNDIs.

Finally, this PoC offers a unique opportunity to validate the trust-related data sharing mechanisms integral

to the CASTOR framework, which operate both at the routing and at the orchestration layer. The former can be addressed through the novel Delegatable Order-Revealing Encryption (DORE) scheme presented in D3.2 [7], which ensures the secure, zero-knowledge sharing of (e.g., path-centric) trust metrics directly across the forwarding plane, maintaining confidentiality and integrity even within complex cross-domain scenarios. The latter is evaluated through the Trust Exposure Layer of the CASTOR DLT, which handles the secure logging, sharing, and exposure of these trust claims at the centralized orchestration level.

Table 3.1: PoC KPIs for TNDE operation in the context of the CASTOR Trust Engineering process

PoC KPI name	PoC KPI definition	Unit
Infrastructure and routing topology initialization	This includes the execution of the overarching Secure Onboarding protocol for the TNDE platform and, ultimately, the TNDI vRouter instances running on top of the infrastructure element. Related to SR.R.6.	sec
TNDE Provisioning Time	Time to enforce the Trust Policy on the TNDE, after the successful onboarding of a new TNDI. This includes the initialization of the appropriate Trust Model in the Local TAF agent, the configuration of the Trace Units and the Trust Sources to collect and report trustworthiness evidence, and the production of the first trustworthiness claim at the Local TAF agent. Related to TNDE.R.2.	sec
Trust Source provisioning and execution overhead	Overhead imposed when securely provisioning the available Trust Sources. In the context of the Attestation Source, this relates to the execution of the JOIN phase for the establishment of the appropriate key restriction usage policies. For the FSM source, however, this refers to the overall Training process for constructing the corresponding behavioural device model that captures critical network operations (see D3.2 [7]). Related to SR.12 and SR.13 respectively.	%
Trace Unit performance overhead	Overhead imposed by the various CASTOR Trace Units on the TNDI operational profile. As highlighted in D3.2 [7], Trace Units leveraging eBPF tracing and kernel extensions will provide critical information on the available CASTOR Trust Sources (i.e., FSM and Attestation Source). Evaluation will be focused on different TNDI instantiations (e.g., multiple TNDIs deployed as containerized applications in a single infrastructure element vs. one TNDI deployed per VM environment). Related to SR.11.	%
Layered Attestation performance overhead in different Hw-based RoT guarantees	Captures the performance gain that the layered attestation scheme initially presented in D3.2 [7] will offer compared to the full-blown attestation of layers in a ”flat hierarchy”. This validation is envisioned to be performed under different levels of isolation assumptions and different RoT flavours (e.g., running TNDE within an Intel SGX, TDX, or simple hypervisor environment).	%
Composite Attestation performance	Runtime performance of the core operations (e.g., key generation, signing, and verification) highlighted in the novel composite attestation schemes presented in D3.2 [7]. This focuses on the transition from node-centric to path-centric attestation primitives. Related to SR.12.	sec

PoC KPI name	PoC KPI definition	Unit
Inter-domain Trust Exposure performance	Performance of trust exposure capabilities, either through the Trust Exposure Layer of the CASTOR DLT or through the forwarding plane and the novel Delegatable Order Revealing Encryption (DORE) scheme (presented in D3.2 [7]). Related to DLT.R.1 and SR.12.	sec

3.3 PoC 2: Trust Engineering process

PoC 2 validates the end-to-end trust engineering pipeline within the CASTOR framework, demonstrating how the TAF evaluates the trust posture of the forwarding plane. This PoC executes a topology-wide assessment by operationalizing three core, equally important pillars: establishing the RTL knowledge base and the Trust Policy that guides trust evaluations, quantifying the evidence-based ATL trust opinions, and resolving the final Trust Decision under uncertainty. By capturing this complete pipeline, the PoC ensures that the resulting trust characterizations are semantically valid and ready to guide the TE provisioning demonstrated in subsequent PoCs.

Trust Engineering process - PoC System Overview
The pipeline initiates with the establishment of the RTL through comprehensive risk analysis. Through the CASTOR Risk Assessment Engine, the PoC common topology is modeled to identify prominent risks and define the security controls required to bring the risk posture into an acceptable state. Executing this phase allows us to validate the proposed topology-aware risk methodology, specifically testing its ability to capture complex dependencies and emergent vulnerabilities across the network architecture. Furthermore, as positioned in D4.2 [8], this evaluation serves as a critical stepping stone toward the probabilistic modelling of cascading attacks, enabling the observation of how dynamically refining transition probabilities via Monte Carlo sampling impacts the overall risk posture. Concurrently, this analytical phase embeds the RTL methodology to generate a formal Trust Policy with strict baseline constraints. By defining the minimum required belief and maximum allowable disbelief for target trust propositions, this step ultimately dictates the exact volume and type of runtime evidence necessary to continuously maintain trust properties across the domain

Guided by the constraints of this Trust Policy, the second pillars of the PoC focuses on trustworthiness quantification to derive the ATL. Building on the preliminary evaluations in D4.2 [8], the Global TAF continuously aggregates runtime evidence streams and trust opinions from the CASTOR in-router trust extensions (i.e., the TNDE). Because these multi-agent sources naturally exhibit varying levels of credibility, the Global TAF employs Subjective Logic operators to combine all agents' opinions accurately. This process accumulates the dynamic, runtime trustworthiness evidence and claims into a finalized ATL opinion for each target trust proposition. Consequently, this PoC enables us to validate the overall federated modality of the CASTOR TAF, both in terms of performance and systemic behaviour. While the isolated evaluations in D4.2 [8] focused on atomic trust propositions (i.e., device-level trust characterizations based on a single type of evidence), this PoC assesses the feasibility of elevating to composite trust propositions. For instance, this involves combining atomic propositions regarding secure boot and configuration integrity into a single, integrity-centric composite proposition. Going beyond device-level trust characterization, this composition unlocks the evaluation of even more complex trust propositions that characterize links, network segments, or entirely established paths. This advancement provides greater flexibility for the path profiles we envision evaluating within the CASTOR use cases.

In the final pillar, the PoC demonstrates the Trust Decision process, effectively bridging the static baselines with the dynamic runtime observations. The Global TAF reaches a definitive conclusion by evaluating the real-time ATL tuples against the established RTL constraints. The introduction of multiple

constraints - specifically, enforcing a minimum required level of belief while maintaining strict maximum allowable caps on both disbelief and uncertainty - significantly restricts the acceptable ATL trust opinion space. This PoC is vital for evaluating how these modelled ATL and RTL values ultimately impact the final trust decision outcome, which inherently carries significant side effects for the Traffic Engineering (TE) decision-making engine. For instance, overly sensitive RTL thresholds may yield non-converging trust evaluations, causing the TE Policy Engine to rapidly fluctuate between ”trustworthy” and ”untrustworthy” routing decisions. Furthermore, this validation provides a significant opportunity to explore the research potential of transitioning toward more complex, multi-state trust propositions. Moving beyond simple binary outcomes (e.g., determining whether a proposition is simply true or false), we can leverage multinomial ATL trust opinions to formulate sophisticated multi-state decisions. Rather than merely evaluating a boolean state such as ”Router 1 is not in the highest level of assurance,” this multi-state approach could allow the TAF to dynamically determine exactly which tiered Assurance class a specific network segment currently operates within, thereby feeding highly nuanced, actionable intelligence into the TE Policy Engine.

Dimensions of validation within the the Trust Engineering PoC

Table 3.2 summarizes the key dimensions that are planned to be validated in this PoC. By deploying the CASTOR Trust Assessment Framework within the common playground topology, the primary objective is to validate the TAF behaviour in relation to the rest of the CASTOR integrated framework. This integrated environment allows us to gather meaningful validation insights regarding the specific trustworthiness evidence required to effectively drive TAF operations, heavily relying on the device-level assurances collected in PoC 1. In parallel, this PoC setup also demonstrates how the resulting domain-wide trust characterization directly influences the broader TE operational lifecycle, serving as the foundational input for the trust-aware routing decisions executed in the remaining PoCs.

Overall, this integrated approach highlights how the TAF provides the essential trust attributes required by the Optimization Engine and the broader TE policy provisioning mechanisms. As indicated in D4.1 [3], when addressing complex multi-objective optimization problems that cannot be modelled as a single, enforceable dynamic Flex-Algo routing policy[2], the TAF outputs are utilized in two distinct ways. First, the granular ATL scores can serve as continuous optimization metrics, used to maximize the overall conditional probability that an end-to-end path will satisfy a specific trust proposition. Second, the definitive Trust Decision outcomes can be applied as hard constraints, instructing the Optimization Engine to prune specific network segments that have been explicitly labelled as untrusted for a given context. Further details on how these trust attributes drive the optimization and provisioning outcomes are discussed in PoC 3 (see Section 3.4).

Table 3.2: PoC KPIs for Trust Engineering process

PoC KPI name	PoC KPI definition	Unit
RTL Methodology Correctness	Behaviour of RTL methodology based on the asset chartography that relates to the PoC topology. Related with RA.R.2	Y/N
TAF Federation Correctness	Behaviour of ATL evolution in both benign and malicious scenarios in the network. Evaluation of Trust Decision outcome against RTL constraints defined as part of the Risk Analysis. Related with TAF.R.2	Y/N

[2] As discussed in PoC 5 (see Section 3.6), when a path profile’s requirements can be satisfied through Flex-Algo definitions, the Global TAF output can be used to label links (e.g., by assigning link affinities or colors). This allows the dynamic, Flex-Algo-enabled routing policy to automatically include or exclude network segments that fail to guarantee the necessary trust requirements.

PoC KPI name	PoC KPI definition	Unit
Overhead of Local TAF agents isolation	Local TAF agent (as well as the entire in-router CASTOR TNDE artifact) shall be equipped with robust mechanisms to withstand against possible attacks (or unexpected failures) in the TNDI or the underlying host environments. Evaluate the overhead that security measures (e.g., isolation of Local TAF agent as an enclavized application) will have on the overall Trust Engineering process. Related with TAF.R.3	%
TAF Federation performance	Evaluate the overhead of the novel CASTOR TAF Federation modality against typical standalone TAF evaluations where the Global TAF is consuming trustworthiness evidence directly from the TNDIs in the forwarding plane. Related with TAF.R.3	%
Trust Exposure Layer (TEL) capabilities & Performance	Evaluate Trust Exposure Capabilities. For TEL capabilities that involve the novel DORE crypto scheme (see [7], performance evaluation on the comparison of available ATL values is envisioned. Related to SR.15	ms

3.4 PoC 3: Trust-aware TE provisioning

PoC 3 demonstrates the core decision-making phase of the CASTOR framework, where the trust and network properties (collected and evaluated in PoCs 1 and 2 respectively) are actively utilized to provision trust-aware routing services. As an endmost goal, it demonstrates the operational integration of a Path Computation Element (PCE) within a multi-node network topology, leveraging IS-IS as the Interior Gateway Protocol (IGP) for dynamic route dissemination and PCEP (Path Computation Element Communication Protocol) as the signaling mechanism for enforcing Segment Routing Traffic Engineering (SR-TE) policies.

This PoC narrative begins with the CASTOR Optimization Engine, which - when triggered - is able to provide near-optimal recommended paths that can satisfy specific network- and trust-related requirements. These requirements directly correspond to the tiered levels of service defined within the CASTOR Path Profile Catalogue (further details on this provisioning flow are available in D5.1 [5]). As detailed in D4.2 [8], the first prototype of the CASTOR Optimization Engine focuses on solving bi-objective optimization problems. This represents a significant evolution from typical Internet Service Provider (ISP) traffic engineering, which traditionally relies on a single routing metric bounded by a set of TE constraints (e.g., in the context of Flex Algo, the provisioned TE policy may select to exclude specific areas of the topology). Instead, the CASTOR Optimization Engine simultaneously evaluates optimal paths across both network performance and trust dimensions. Validating this bi-objective approach constitutes the primary evaluation step for the first integrated version, establishing a baseline before expanding to more complex multi-objective optimization scenarios in the second and final release.

Following the path computation phase, the output from the Optimization Engine must be translated into actionable routing directives. As highlighted in Chapter 2, this responsibility falls to the TE Policy Engine, which determines how the optimization results are encoded into an enforceable network policy. The resulting Segment Routing (SR) policies can exhibit varying levels of complexity and support different reaction approaches. For instance, the TE Policy Engine may choose to encode a comprehensive Segment Routing (SR) policy comprising multiple candidate paths (i.e., one active path while reserving several backup paths to support a robust restoration strategy). In its simplest form, however, the resulting network policy may consist of a single, explicitly defined path that is directly enforced within the forwarding plane via the PCE.

The evaluated topology consists of multiple nodes in hierarchical topology. One of the deployed nodes

in the PoC topology acts as the PCE itself, forming a unified control and data plane environment. All nodes participate in IS-IS routing, enabling dynamic discovery and advertisement of network prefixes and topology state. In addition to dynamically learned routes, select routers within the topology are provisioned with static MPLS label entries, providing a deterministic forwarding baseline that coexists with the dynamic IGP control plane.

From the PCE, SR-TE policies are instantiated and distributed to Path Computation Clients (PCCs) using PCEP. These policies incorporate explicit segment lists with elevated metric weights, enabling the PCE to override default shortest-path forwarding decisions and impose traffic-engineered Label Switched Paths (LSPs) across the topology without requiring any modification to the underlying IGP configuration. This separation between control plane intent and IGP state is a key attribute of the demonstrated architecture, as it illustrates the flexibility and non-disruptive nature of PCE-driven traffic engineering.

To validate correct policy enforcement, traceroute commands are executed from a host located at one end of the topology to a host at the opposite end, both before and after SR-TE policy application. The resulting hop sequences are compared to confirm that traffic follows the explicitly defined MPLS forwarding paths dictated by the PCE, demonstrating a measurable and verifiable path deviation attributable solely to the applied policy. In conjunction with traceroute analysis, the routing tables corresponding to the relevant color and/or VRF used during path selection are inspected, providing further evidence that the correct forwarding context is active and validating end-to-end coherence between control plane intent and data plane behavior.

To further characterize the responsiveness of the PCE-driven policy enforcement mechanism, a latency measurement procedure is incorporated into the validation methodology. Network captures are performed simultaneously at two distinct points in the topology: one at the PCE's interface, to record the precise moment at which the PCEP signaling for a new SR-TE policy is transmitted, and one at the router connected to the host designated as the policy application point, to detect the moment at which traffic begins traversing the newly enforced segment routing path. The timestamp of the outgoing PCEP message serves as the start marker, while the first traceroute response or forwarded packet reflecting the updated MPLS forwarding behaviour serves as the end marker. The difference between these two timestamps yields a quantifiable measure of the end-to-end change propagation latency — that is, the time elapsed between the PCE issuing a new path intent and that intent being effectively realized in the data plane. Note that both traffic captures must be synchronized with an error of less than 1 ms to obtain conclusive evidences. This metric provides valuable insight into the operational efficiency of the PCEP-based control loop and the practical responsiveness of the SR-TE policy lifecycle within the demonstrated architecture.

Table 3.3 shows the KPIs that would be included in this section.

Table 3.3: PoC KPIs for Trust-aware TE provisioning

PoC KPI name	PoC KPI definition	Unit
Comparison of optimisers	Optimality Gap (vs. Exhaustive Oracle) on small examples. Related with OPT.R.1	%
Routing selection	Selected path compliance with ATL. Related with OPT.R.2	Y/N
Alternative Paths per Request	Number of alternative routes proposed by the component. Related with OPT.R.3 (1)	%
Average Service Time	Average Optimization Engine Invocation Arrival Time. Related with OPT.R.3 (2)	$<=$
Re-optimization Success Rate	High ratio to support re-optimisation and enable checkpointing to allow search rollbacks. Realted with OPT.R.4	%
Problem Scaling Factor	Being able to solve the problem of trust path optimization problem in practical time frame for real-world representative size instances. Related with OPT.R.5	$<=$

PoC KPI name	PoC KPI definition	Unit
PCE path deploy	Time to enforce via PCEP a new path to a PCC. Related with TE.R.1	s
TEPE & PCE computation latency overhead	Overhead imposed by CASTOR framework to the PCE computation latency. Related with TE.R.2	%

3.5 PoC 4: Telemetry and SSLA Monitoring

This PoC demonstrates the integrated and core mechanisms of the Orchestration Layer and, through the scenario “Trust-Aware UAV Data Delivery Across Mobile Edge Attachments” presented in D2.1 [4], focuses on the operation of the Network Service Orchestrator (NSO) and its S(S)LA detection functionality, highlighting the capabilities enabled by leveraging telemetry and trust-aware network assessment in conjunction with application behaviour in the field-edge domain. The overall application scenario of this PoC along with the interaction with a CASTOR-enabled forwarding plane is envisioned to be validated within the WINGS testbed presented in Section 4.3.

The NSO functionalities can be classified into two main categories, with a primary focus on management and monitoring operations, respectively. The former encompasses the management of network node onboarding, to ensure the availability of the required compute and network capabilities, and of the corresponding virtualized routing functions, in order to enforce the requested SLA/SSLA/path profiles. The latter comprises the monitoring operation to verify compliance with SLA/SSLA and to trigger the appropriate adaptive actions when needed.

In the framework of the management operations, the NSO selects, when needed, a suitable node based on placement, available resources, possible node affinity rules, trust level and topology constraints, monitors the instantiation of virtualized routing functions and manages the respective lifecycle operation, irrespective of the underlying implementation technology. The management of virtualized routing functions encompasses onboarding performance and correctness, rescaling efficiency, control-plane and data-plane stability and telemetry collection reliability and timeliness. The lifecycle of vRouters instances typically progresses through several operational states, including onboarding, running, operational readiness, degraded, scaling, reconfiguration, recovery, and termination/decommissioned. While the running state indicates that the router process is active, operational readiness is achieved only after full configuration deployment with the active intervention of the Network Controller. The onboarding procedure refers to the complete process from orchestration request to full operational availability and telemetry activation and is considered successful when the vRouter is successfully scheduled with the proper networking interfaces enabled/attached, the vRouter management plane is reachable (e.g.NETCONF/gNMI), the telemetry pipeline is operational (e.g. Prometheus scrape success, telemetry exporter produces router metrics, metric freshness within threshold) [D2.1, D5.1], the vRouter is visible in topology graph and is eligible for path computation and consequently may participate in Traffic Engineering policies. The failure in any of these main functionalities results in onboarding rejection. During vRouter’s operation, resource sufficiency is monitored via proper mechanisms and evaluated in order to reveal probable operational impact of limited resources, e.g. impact on routing stability in traditional routing protocols such as BGP, OSPF, IS/IS. In this case, rescaling actions to ensure avoidance of control plane disruption, preservation of forwarding state and maintenance of session continuity, may involve – depending on specific technical characteristics of deployment - increasing or decreasing CPU limits, adjusting memory allocation, modifying hugepages, adjusting I/O queue depth, or instantiating additional vRouters. The definition of parameters (KPIs) that enables the systematic assessment of the required operational conditions, is critical to finally ensure that the vRouter deployment and operation is successful. Indicative examples of such basic parameters, independent of the implementation technology, are included in table 3.3. The node onboarding and rescaling KPIs may be leveraged to establish a common reference, depending on how the application is implemented by the application/ service provider. Methods and approaches for

rescaling KPIs evaluation, such as the CPU utilization at VM level on the host, will be further investigated in D6.2.

Table 3.4: PoC KPIs for Node Onboarding

PoC KPI name	PoC KPI definition	Unit
Resource allocation success rate	Percentage of requested compute, memory, storage, and network resources that are successfully allocated to a virtual network function instance.	%
Trust verification success rate	Percentage of trust verification processes that are successfully completed during node onboarding and validation.	%
Network attachment correctness	Percentage of onboarding cases in which a node is correctly connected to the intended network interfaces, segments, and domains.	%
Configuration transaction success rate	Percentage of configuration transactions initiated by the orchestration system that are successfully completed without errors or inconsistencies.	%
End-to-end onboarding time	Total time required to complete the onboarding process of a network node, from the initial request until the node becomes fully operational and ready for service delivery.	sec

In the context of dynamic, multi-domain, and trust-aware environments, the effectiveness of orchestration logic is inherently framework for assessing the successful operation of the required application instantiations, dependent on the continuous availability of accurate and low-latency telemetry data. Particularly, in the case of the S(S)LA monitor operation that is used to assess adherence to SLA/SSLA/path profiles, telemetry is of critical importance not only for trust-aware network operation, but also for the services and applications that are built upon it. The effectiveness of NSO monitoring capabilities is contingent upon accurate acquisition of telemetry data with minimal retrieval latency. High data accuracy ensures that monitoring decisions are based on trustworthy inputs, minimizing false positives or undetected violations, while low-latency retrieval is critical for near real-time detection of anomalies and policy deviations, thereby achieving the desired operational state of both network functions and dependent applications. Therefore, efficient telemetry pipelines leveraging corresponding mechanisms - streaming collectors, exporters, data brokers — are necessary to ensure that the Orchestration Layer operates with a consistent and accurate view of the network state.

The telemetry data (see D5.1 [5]) includes information concerning network operation from network entities as virtual routers, gateways, and service function chain (SFC) elements (e.g. latency, jitter, packet loss, throughput, interface counters), infrastructure resources (e.g. CPU, memory and storage utilization of physical routers, and vRouter resource and compute metrics), end-to-end service provision (e.g. end-to-end latency, throughput stability), trust (e.g. security posture indicators), and alarms (e.g. failed VNF onboarding). All telemetry data feeds the SSLA monitoring function of NSO, by evaluating them against defined SLOs of SLA/SSLA. When SLA/SSLA compliance indicators exhibit deviations outside the predefined evaluation window, remediation or fallback actions from the responsible Orchestrator entities are triggered. These actions are required to restore the demanded network operation and performance, but also the operational context under which the services operate.

This PoC centers on how CASTOR's orchestration layer can interact with application level services and far-edge compute nodes, through the use of telemetry insights combined with trust-informed network assessment. Specifically, the S(S)LA monitor operation of NSO guarantees that S(S)LA-compliant paths are available to be used by IoT Gateway in the field-edge domain, for sending alert messages along with the accompanying evidence from the IoT devices and UAVs. In this case, WINGS Orchestration Layer (WSMO) orchestrates the wi.BREATHE application workload in WINGS cloud domain for the implementation of a trusted fire detection and alert dissemination mechanism. The fire detection results are sent

to the responsible unit of the Civil Protection Service together with assessment of trust and network conditions, based on the telemetry data, which provide evidence that the information transfer was carried out through a path that meets the SLA/SSLA requirements, therefore, the application results are fully reliable and reflect the real conditions, and can be immediately propagated to the operational units for natural disaster response (fire brigade, aerial units, healthcare units). If the S(S)LA monitoring function detects a violation of the SLOs, and the actions of the respective orchestration layer entities fail to restore the required conditions, the WSMO orchestrates the wi.BREATHE application workload in the UAV/Edge, where the UAV/Edge must evaluate the trustworthiness of the fire detection itself.

In this context, KPIs regarding monitoring/telemetry operation are included in Table 3.4.

Table 3.5: PoC KPIs for Monitoring/Telemetry

PoC KPI name	PoC KPI definition	Unit
Telemetry availability	The percentage of telemetry data collection attempts that are successfully completed, resulting in valid monitoring data.	%
Metric freshness	The maximum time difference between the moment a metric is generated at the monitored system and the moment it becomes available for consumption by monitoring or orchestration components.	sec
Scrape interval	The time interval at which the monitoring system collects telemetry data from a target endpoint (e.g., exporter or network node).	sec
Network monitoring overhead	The proportion of monitoring-related network traffic relative to the total network traffic.	%
Alert detection accuracy	The percentage of correctly identified and triggered alerts corresponding to actual network or service anomalies, failures, or SSLA violations.	%
Monitoring recovery time	The time required for the monitoring system to restore normal operation after a failure or disruption.	sec
S(S)LA violation detection latency	The time elapsed between the moment a service deviates from its defined SLA/SSLA objectives, as reflected in telemetry data, and the moment the SLA Monitor identifies the respective violation.	sec

3.6 PoC 5: TE Restoration strategies

The main objectives of CASTOR framework are not only to establish a connection with agreed SSLA requirements, but also to continuously monitoring the provided network QoS and trust level in order to assure that the agreed SSLA is fulfilled throughout the connection lifetime. In this direction a set of restoration strategies (i.e., defined as the mechanisms used to re-establish network connectivity after an occurred failure [13]) will be introduced to ensure fast and reliable responses to underlying changes; those include any network (e.g., link failure) or trust (e.g., update of trust opinion) change that may affect the provided SSLAs. Regarding the events of interest that will be evaluated in the context of CASTOR experimentation, a number of indicative 'update cases' are listed below:

- **Node/link failure**: Node (router) or link/interface issues that cause an unexpected node or link operation termination. All services that have their traffic passing through such nodes or links will be affected (i.e., no operational path through that node or link).

- **Changes in trust level**: Changes in the calculated trust level of a node or link, potentially updated through a change in a trust opinion. In that case the TAF calculates trust levels below the agreed SSLA for a link or a node. All the services with greater trust requirements are affected (no operational path exhibits the SSLA-prescribed trust level through the node or link).
- **Network QoS degradation**: Due to high utilisation of a node or a link the provided network performance is below the agreed SSLA (e.g. lower provided bandwidth due to high link utilisation or high latency due to congestion).

Table 3.6 depicts an indicative set of performance indicators that will help us evaluate the performance of the CASTOR system for the considered PoC. As already mentioned at the beginning of this chapter, even if some of the KPIs reflect similar notions of interest (e.g., latency) appearing in other PoCs, they retain a specific name in each PoC for ease of reference.

Table 3.6: PoC KPIs for the evaluation of CASTOR TE Restoration strategies

PoC KPI name	PoC KPI definition	Unit
Restoration latency	The latency between the time that an SSLA degradation event is identified by the CASTOR system until the time that the new configuration is applied on the routers.	ms
Service interruption time	The latency between the time the service is starting affected by the degradation event (SSLA violation time) until the time the service performance is restored to the agreed SSLA.	ms
SSLA degradation identification latency	The latency between the time the SSLA degradation actually happens until the time the CASTOR system identifies this degradation.	ms
Optimisation latency (for generation of a set of paths)	The latency required by the Optimisation Engine in order to generate the set of new paths.	ms
Optimisation latency (for selecting the appropriate path)	The latency required by the TE Policy Engine in order to select the final path from the set of paths generated by Optimisation Engine.	ms
Policy generation latency	The latency required by the TE Policy Engine in order to generate the appropriate set of policies.	ms
Configuration enforcement latency	The latency required by PCE to enforce the decided configuration on the routers.	ms
Monitoring overhead	The traffic overhead imposed by the CASTOR monitoring process.	%
Optimisation success rate	The percentage of cases that the solution provided by the Optimisation Engine fulfills (in practice) the SSLA requirements.	%
SSLA degradation identification accuracy	The percentage of SSLA degradation cases that were successfully identified by the CASTOR system (including also false positive).	%

In response to the above events, CASTOR plans to establish and evaluate a set of restoration strategies, given an initial network configuration (paragraph 4.2 in D5.1 [5]). The main technical concept in each case is briefly explained below:

- **Connections are established using SR flex-algorithms with protection paths**: in case of detected events, the protection paths will be activated, and the corresponding service traffic will be diverted toward the protection path.

- **Connections are established using SR flex-algorithms without protection paths**: in case of detected events, a new flex-algorithm path will be calculated by PCE and enforced to routers.
- **Connections are initially established using SR flex-algorithms (baseline), while explicit paths are used in order to fulfil the SSLAs and as responses to events**: Initially connections are established using SR flex-algorithms. In addition, explicit paths are used in order to ensure the required SSLAs. In case of events, the explicit paths become unavailable, therefore the system falls back to the initial flex-algo paths. Then, the Optimization Engine is responsible to generate a set of new explicit paths, which are sent to TE Policy Engine. The TE Policy Engine selects the most appropriate explicit path from the set and generates the appropriate policies. The selected explicit path is subsequently enforced by the PCE.

The plan is to further experiment with cases where the performance of one service may be affected by other services traffic passing through the same link or node. In such cases, the Optimisation Engine is expected to offer a combinatorial solution identifying the set of paths (utilised by each different service) in order to alleviate the problem. Our performance evaluation of the restoration strategies mentioned above, will rely on the collection, processing and evaluation of a set of PoC KPIs (i.e., some of them capture similar concepts with the D2.1 KPIs) illustrated in Table 3.6.

Chapter 4

The CASTOR Validation and Experimentation Platform

Realizing a comprehensive validation and evaluation framework is critical to accommodate not only the realistic use case evaluation scenarios but also all the critical added value and aspects of the CASTOR integrated framework. Being at the center of the CASTOR integrated framework evaluation in the context of the use cases, the full-fledged ORO testbed is equipped with a production-grade forwarding plane consisting of data-plane virtual routers (vRouters). This setup ensures it can accurately accommodate realistic network workload conditions, serving as the primary environment for assessing the framework's overall performance and scalability in realistic applications presented in the four Use Cases of the CASTOR project.

Alongside this primary infrastructure, two complementary testbeds are utilized to streamline the validation and benchmarking of the specific behaviours described in the various PoCs. First, the TNDE testbed provides diverse hardware-based Root-of-Trust (RoT) capabilities required to evaluate CASTOR's in-router trust extensions and the secure lifecycle management of the deployed TNDIs. Second, the WINGS testbed delivers advanced Kubernetes orchestration capabilities, which are essential for managing the lifecycle and automated deployment of the forwarding plane across the computing continuum. This testbed diversity showcases the richness of the overall CASTOR validation and evaluation framework to accommodate all needs towards evaluating the technical characteristics of the integrated framework and also the overhead that it introduces in the context of the use case application scenarios.

4.1 CASTOR Testbed for UC validation

The "Orange 5G Lab" ORO Testbed functions as a vital hub for discovering and utilizing the capabilities of 5G technology. This ecosystem is designed to unify a diverse group of stakeholders, including industry partners, innovators, and businesses, to test, develop, and showcase new use cases empowered by 5G networks. The labs serve as an innovative and experimental platform where businesses can validate and refine their products and services using advanced 5G networks. This setup aims to reduce the resources and time usually needed to progress from concept to market deployment. The Orange 5G Lab ecosystem is deployed in two cities, in Romania – Bucharest, and Iași and it is operated with Academia partners from the two cities.

4.1.1 Testbed Architecture and Equipment

ORO Testbed sites in Bucharest and Iași, are the main places that host the infrastructure used in ORO's 5G-enabled research activities. The Bucharest site is located inside the CAMPUS Research Center

of UNSTPB and currently implements a full 5G Standalone (SA) infrastructure, comprising 5G RAN, Core, Edge Computing and advanced Software Defined Networking (SDN) elements in the associated datacenter. The Iasi 5G Lab is located inside TUIASI's Faculty of Electronics and Telecommunications and hosts 5G SA RAN components. Figure 4.1 depicts the overall architecture of ORO's 5G Labs and the outdoor extensions of the testbeds, which are described in the following sections.

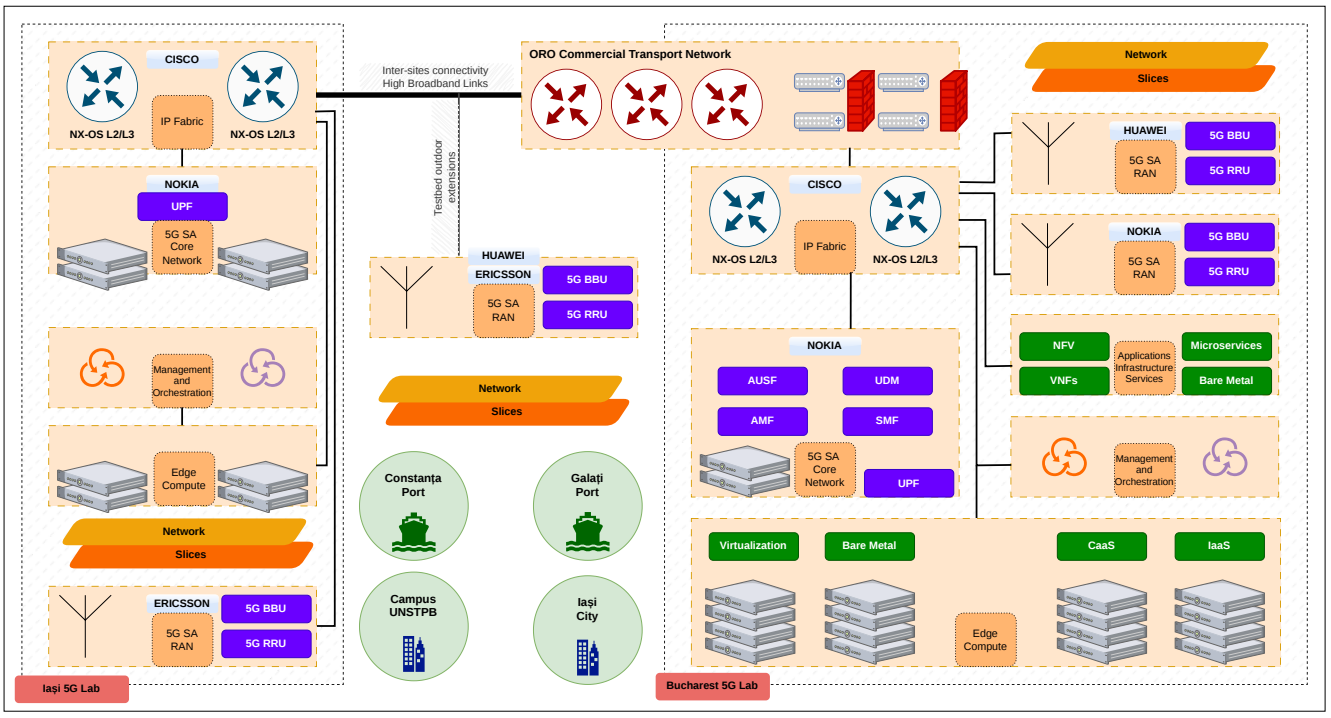

Figure 4.1: ORO Testbed Architecture

4.1.1.1 5G Core Capabilities

ORO testbed is based on virtualized network infrastructure, for both 5G options, the Non-Standalone (NSA) and Standalone (SA) variants.

For the NSA variant, the testbed is connected to the commercial LTE Core Network. The NSA Core architecture is based on the virtual Evolved Packet Core (vEPC) solution from Ericsson, deployed in an OpenStack infrastructure – Orange IaaS. The existing core architecture can support broadband connectivity featuring configurable Downlink (DL)/Uplink (UL) throughput values up to 1Gbps for DL and 100Mbps for UL. For this existing core architecture, the 5G services are manually configured, by using dedicated QoS, traffic prioritization, private Access Point Names (APNs) and network provisioned capabilities in terms of throughput, as the main service that can be served is enhanced Mobile Broadband (eMBB).

The SA Core architecture uses the 5G SA Compact Mobility Unit (CMU) Core Network solution from Nokia, which is running on a virtualized infrastructure in the Bucharest 5G Lab datacenter. Three main components are integrated, the Cloud Mobile Gateway (CMG), the Cloud Mobility Manager (CMM), and the Authentication and Policy Control (APC). These components implement the 5G SA 3GPP defined Network Functions (NFs), represented in Figure 4.2, as follows:

- The Authentication and Mobility Management Function (AMF), managed by the CMM, handles UE tracking and authentication inside the core;
- The Session Management Function (SMF), managed by the CMG, handles Protocol Data Unit (PDU) sessions, selection of the UPF for data service based on the QoS needs and session authentication via the UDM;

- The User Plane Function (UPF) handles the data sessions payload, payload prioritization, multi-access edge computing integration and communications to the access domain; sits inside the Bucharest 5G Lab CMG and will be locally deployed in Iasi in a Local Break-Out (LBO) approach;
- The Authentication Server Function (AUSF), managed by the APC, handles the users' authorization, authentication, and accounting (AAA) tasks;
- The Unified Data Management (UDM) unit acts as a subscriber management frontend for 5G SA users;
- The Unified Data Repository (UDR) works as a subscriber profile repository.

Moreover, the testbed facility supports different slices based on 3GPP NSSAIs parameters, providing eMBB with throughput up to 1.5Gbps for DL and 150Mbps for UL and URLLC with a one-way delay of $<$4ms.

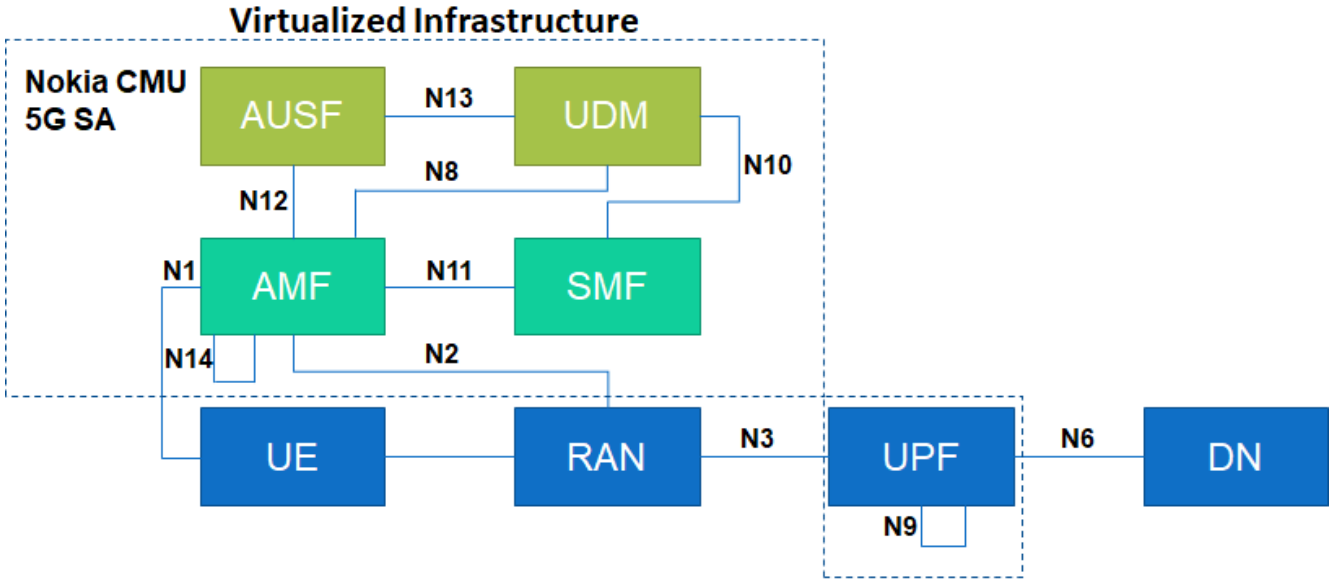

Figure 4.2: 5G SA Implementation

4.1.1.2 Radio Access Networks

For the implementation of the 5G NSA architecture in the testbed, ORO is using the N78 band (3.5 GHz) and the 3GPP Option 3x. As showcased in Figure 4.3, by using the Fourth Generation of mobile communications (4G) as the anchor point for the control plane traffic, a good service continuity can be met while supporting rapid network construction in the initial stage of 5G deployment.

The ORO Testbed 5G SA RAN architecture follows the 3GPP Option 2; in this approach, as showcased in Figure 4.4, the RAN consists of only gNBs that are connected to the 5G Core network. The gNBs connect to AMF over the NG-C interface for control plane signaling and to UPF over the NG-U interface for user plane data transfer. gNBs provide the NR (New Radio) air interface access for the User Equipments (UEs). The deployed 5G SA RAN components also include support for network slicing, leveraging on priority management with differentiated 5G QoS flows, prioritized UE TCP sessions and 5G QoS flows to Data Radio Bearer (DRB) mapping – in this approach the network maps different IP flows with the same QoS requirements into the same QoS flow.

In the Bucharest site's workspace area the main 5G RAN setup is provided by Huawei (SRAN18) and is based on general-purpose hardware (that is the exact representation of what ORO is using on commercial

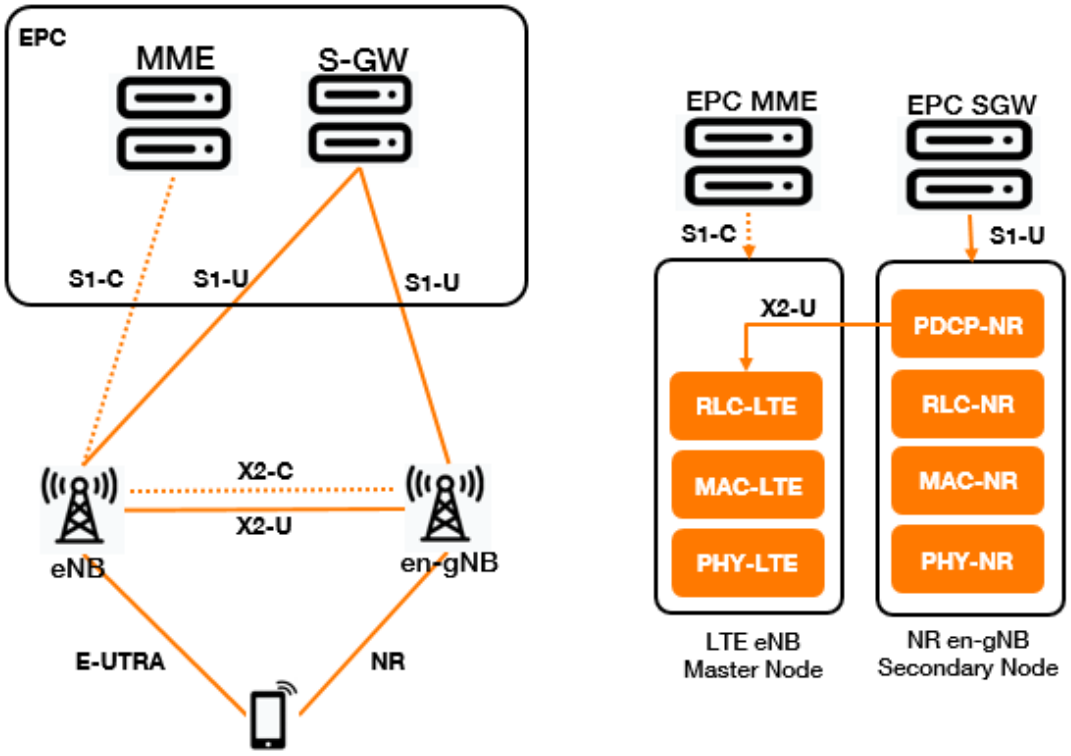

Figure 4.3: 5G NSA 3GPP Option 3x

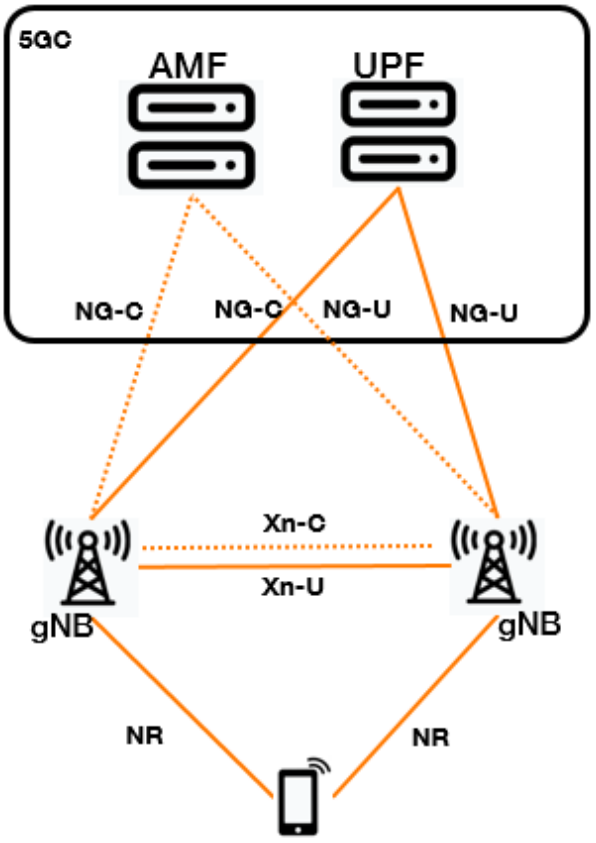

Figure 4.4: 5G SA 3GPP Option 2

radio sites). The setup consists of a Baseband Unit (BBU), several Radio Units (RUs) (for different technologies and frequencies), attenuators and passive antennas with Multiple Input, Multiple Output (MIMO) capabilities. The secondary 5G RAN setup is provided by Nokia and is based on commercial indoor hardware. The setup consists of a BBU, a radio distributer and two active antennas; due to poor performance and stability during ORO's RAN testing and validation activities this solution is not being used currently. Furthermore, it's important to note that for very high demanding testing cases, the Bucharest site is also equipped with an anechoic cage.

In the Iasi site, the 5G RAN is provided by Ericsson and based on a commercial indoor active Distributed Testbed Nodes (DOTs) solution. The setup consists of a BBU, an Indoor Radio Unit (IRU) and two active antennas (DOTs), one covering the 5G Lab workspace area and another one covering the associated IoT teaching laboratory of the university. The 5G spectrum reserved for the ORO Testbed environment is up to 100MHz in the N78 band, different radio Carrier Aggregation (CA) solutions for improving DL and UL performance being possible based on 3GPP specifications.

Finally, as showcased in Figure 4.1, ORO Testbed is also extended in the outdoor space within two of the biggest ports in Romania (Constanta and Galati) and in several outdoor areas of Iași, where the principal experimentation activities in TUIasi's Use-Case are going to be validated.

Radio components for CASTOR deployment

Besides the commercial infrastructure placed in the two sites and already described in the above paragraphs, ORO has also fitted the two laboratories with open-source components that could emulate RAN functions. In this approach, the testbeds are hosting several Software Defined Radio (SDR) units for radio research purposes (NI USRP B210/X300/X310/N310) that were already tested and validated using open-source 5G SA RAN software components from OpenAirInterface or srsRAN.

O-RAN components

Complementing the research-oriented radio components described above, the Bucharest Site is also currently fitted with one O-RAN compliant RU coming from AW2S that was validated to work with OpenAirInterface virtualized BBU software and it's also compatible with other O-RAN solutions coming from providers like Acceleran, BubbleRAN or Amarisoft.

4.1.1.3 Transport Networks and Security

ORO Testbed is built on top of an IP/MPLS transport network, consisting of access, aggregation and Core Network layers, interconnected through high-speed links – 100Gbps broadband connectivity between the network elements, QoS control and fast services delivery. The ORO Testbed focus is mainly the integration and deployment of the 5G SA target architecture based on IaaS/CaaS virtualization solutions. ORO Testbed's associated transport network architecture is split into three main components:

- 5G transport network in the Bucharest site, deployed as a cluster of advanced network equipment, mainly for the integration of the virtualized infrastructure, 5G gNBs and 5G Core Network;
- 5G transport network in the Iasi site, deployed as an extension of the Bucharest IP network infrastructure (LEAF) for the local UPF integration, gNB aggregation and direct connection to local edge computing servers;
- E2E transport network service configuration agents used for manual configuration of the network, dedicated traffic VPNs, QoS definition and service traffic KPIs assurance purposes. This transport network was designed to fulfil the 5G transport requirements, based on the architecture illustrated in Figure 4.5, with two main key objectives:
  - To ensure the high transport capacity and QoS requirements associated with the specific network service slices transported by the network;

– To provide dynamic, resilient and redundant transport paths.

The ORO Testbed transport network solution is using the IPFABRIC concept (a vendor-neutral insights), using Border Gateway Protocol (BGP) Ethernet Virtual Private Network (EVPN) for control plane and Virtual Extensible LAN (VXLAN) for data plane encapsulation. The solution offers a lot of benefits compared to the legacy layer-oriented datacenters using Core, Distribution and Access layers respectively.

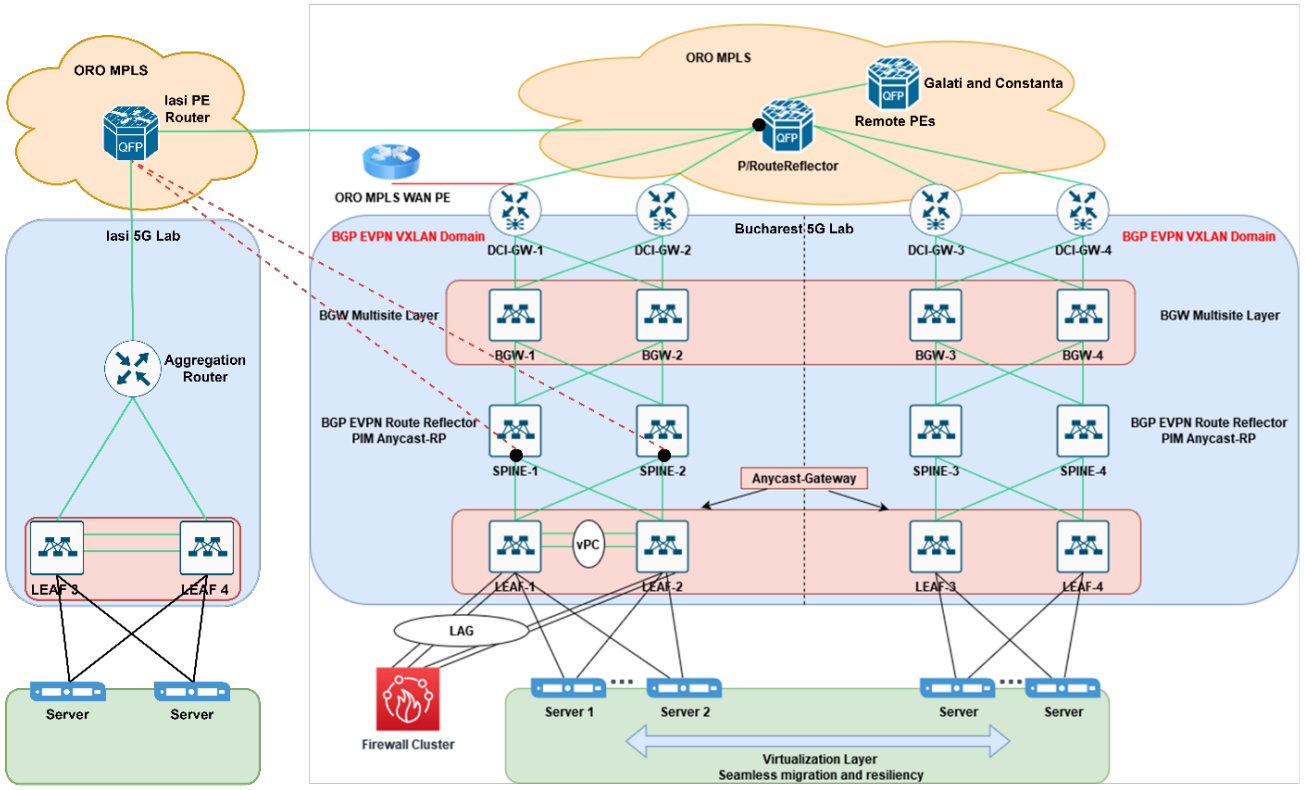

Figure 4.5: ORO Testbed transport network design

Furthermore, ORO Testbed is fitted with a security solution designed to alleviate the issues and concerns that may appear when allowing 3rd party experimenters to onboard and operate their applications within the testbed infrastructure. The firewall deployment showcased in Figure 4.5 is meant to streamline the access procedure to the testbed for the CASTOR project partners and is assuring the proper isolation between different network applications and services.

4.1.1.4 Edge Computing Capabilities

Connected to the 5G Core Network and RAN infrastructure, through the IPFABRIC transport network, is the ORO Testbed edge computing facility. This facility is shared between the two labs from Bucharest and Iasi and consists of 18 compute servers, totalizing in 544 logical CPU cores, 8 TB of RAM, 42 TB of storage and 6 GPUs (64 GB VRAM), as showcased in Table 4.1 and Table 4.2, below:

No	Server	Model	CPU				RAM	Network		Storage		GPU	
			Qty	Model	Freq	vCores	Qty	Interfaces	Capacity	Capacity	Type	Qty	Model
1	5glab-node01	HPe DL360 G10	2	Intel Xeon 4114	2.2 Ghz	20	96 GB	10	10 Gb	2.02 TB	Hybrid – SSD & HDD	-	-

No	Server	Model	CPU				RAM	Network		Storage		GPU	
			Qty	Model	Freq	vCores	Qty	Interfaces	Capacity	Capacity	Type	Qty	Model
2	5glab-node02	HPe DL360 G10	2	Intel Xeon 4114	2.2 Ghz	20	96 GB	10	10 Gb	2.02 TB	Hybrid – SSD & HDD	-	-
3	5glab-node03	HPe DL360 G10	2	Intel Xeon 4116	2.1 Ghz	24	192 GB	10	10 Gb	2.2 TB	SSD	-	-
4	5glab-node04	HPe DL360 G10	2	Intel Xeon 4114	2.2 Ghz	20	96 GB	10	10 Gb	3.6 TB	Hybrid – SSD & HDD	-	-
5	5glab-node05	HPe DL360 G10	2	Intel Xeon 4116	2.1 Ghz	24	192 GB	10	10 Gb	3.84 TB	SSD	-	-
6	5glab-node06	HPe DL360 G10	2	Intel Xeon 4116	2.1 Ghz	24	192 GB	10	10 Gb	1.92 TB	SSD	-	-
7	5glab-node07	HPe DL360 G10	2	Intel Xeon 4114	2.1 Ghz	20	96 GB	10	10 Gb	3.6 TB	HDD	-	-
8	5glab-node08	HPe DL360 G10	2	Intel Xeon 4116	2.1 Ghz	24	192 GB	10	10 Gb	1.92 TB	SSD	-	-

Table 4.1: ORO Testbed EDGE Servers (Fixture A)

No	Server	Model	CPU				RAM	Network		Storage		GPU	
			Qty	Model	Freq	vCores	Qty	Interfaces	Capacity	Capacity	Type	Qty	Model
9	5glab-node09	HPe DL360 G10	2	Intel Xeon 4114	2.2 Ghz	20	96 GB	10	10 Gb	4.02 TB	Hybrid – SSD & HDD	-	-
10	5glab-node10	HPe DL360 G10	2	Intel Xeon 4116	2.2 Ghz	24	192 GB	10	10 Gb	2.45 TB	Hybrid – SSD & HDD	-	
11	5glab-node11	HPe DL360 G10	2	Intel Xeon 4116	2.2 Ghz	24	192 GB	10	10 Gb	2.45 TB	Hybrid – SSD & HDD	-	
12	5glab-node12	HPe DL360 G10	2	Intel Xeon 4116	2.2 Ghz	24	192 GB	10	10 Gb	3.84 TB	SSD	-	
13	5glab-storage1	HPe DL360 G10	2	Intel Xeon 4208	2.1 Ghz	16	64 GB	6	25 Gb	32 TB	HDD	-	
14	5glab-node14	HPe DL360 G11	2	Intel Xeon 6430	2.1 Ghz	64	2048 GB	13	10 Gb	15.36 TB	SSD	-	
15	5glab-node15	HPe DL360 G10+	2	Intel Xeon 6348	2.6 Ghz	56	1024 GB	2	25 Gb	7.68 TB	SSD	3	nVIDIA T4

No	Server	Model	CPU				RAM	Network		Storage		GPU	
			Qty	Model	Freq	vCores	Qty	Interfaces	Capacity	Capacity	Type	Qty	Model
16	5glab-node16	HPe DL360 G10+	2	Intel Xeon 6348	2.6 Ghz	56	960 GB	2	25 Gb	7.68 TB	SSD	3	nVIDIA T4
17	5glab-node17	HPe DL360 G10	2	Intel Xeon 4114	2.2 Ghz	20	640 GB	2	10 Gb	1.2 TB	HDD	-	
18	5glab-node18	HPe DL360 G11	2	Intel Xeon 6430	2.1 Ghz	64	2048 GB	2	25 Gb	15.36 TB	SSD	-	

Table 4.2: ORO Testbed EDGE Servers (Fixture B)

Associated with the ORO Testbed deployments, there are two datacenters, showcased in Figure 4.6 and Figure 4.7, that are hosting the networking and computing equipment allowing for the implementation of all the functions and capabilities mentioned in the above and following subsections.

The Bucharest data center is hosted in the same building as the laboratory workspace area and consists in 6 racks (4 of them fitted with UPS units), high-performance climate units and high-throughput links (100 Gbps) to ORO's commercial transport network.

The Iași data center is hosted in an auxiliary building, in the TUIași venue, and in proximity to the principal 5G Lab experimentation spaces.

The two data centers are interconnected in a high-availability, resilient network deployment consisting of high-speed (25 GbE) links.

4.1.1.5 Computing Capabilities of the "CASTOR" testbed tenant

ORO will make available a dedicated tenant for advanced experimentation, in the CASTOR framework, and to support the onboarding, testing and validation of the project's use-case.

The dedicated compute and storage capabilities will be physically located in the București and Iași facilities. The CASTOR tenant is a logical partition hosted within the Orange 5G Labs infrastucture.

The provided tenant will include telemetry and performance monitoring capabilities, allowing for the granular collection of data during validation. The underlying infrastructure, as described in 4.1.1.4 Edge Computing Capabilities will be used to ensure access to a baseline resources allocation of more than 100 vCPUs based on Intel x86 Gen 12/Gen 13 cores, 256GB RAM, 10Gb Network Interfaces, tiered hybrid storage (hot – warm – cold) and nVidia CUDA processing.

4.1.2 ORO Testbed Platforms and Capabilities

4.1.2.1 Slicing Capabilities

The ORO Testbed implements two different network slices based on 3GPP Network Slice Selection Assistance Information (NSSAI) parameters, providing eMBB with throughput up to 1.5Gbps for DL and 150Mbps for UL and URLLC with a one-way delay of $<$4ms, the overall slicing architecture being shown in Figure 4.8. At the RAN level the slicing performance is ensured by leveraging gNB-specific 5G Quality Indicators (5GQIs) functionality for proper slice implementation (Uplink Preallocation, SR Period Configuration). The two slices have different Data Network Names (DNNs), "*orange5glab*" for the eMBB slice and "*orangeurllc*" for the URLLC slice.

The network slicing is implemented based on 3GPP-defined parameters; the Single-NSSAI (S-NSSAI) identifies network slices in NG-RAN and 5GC and is composed of a Slice Service Type (SST) and Slice

Figure 4.6: Bucharest Data Center

Figure 4.7: Iași Data Center

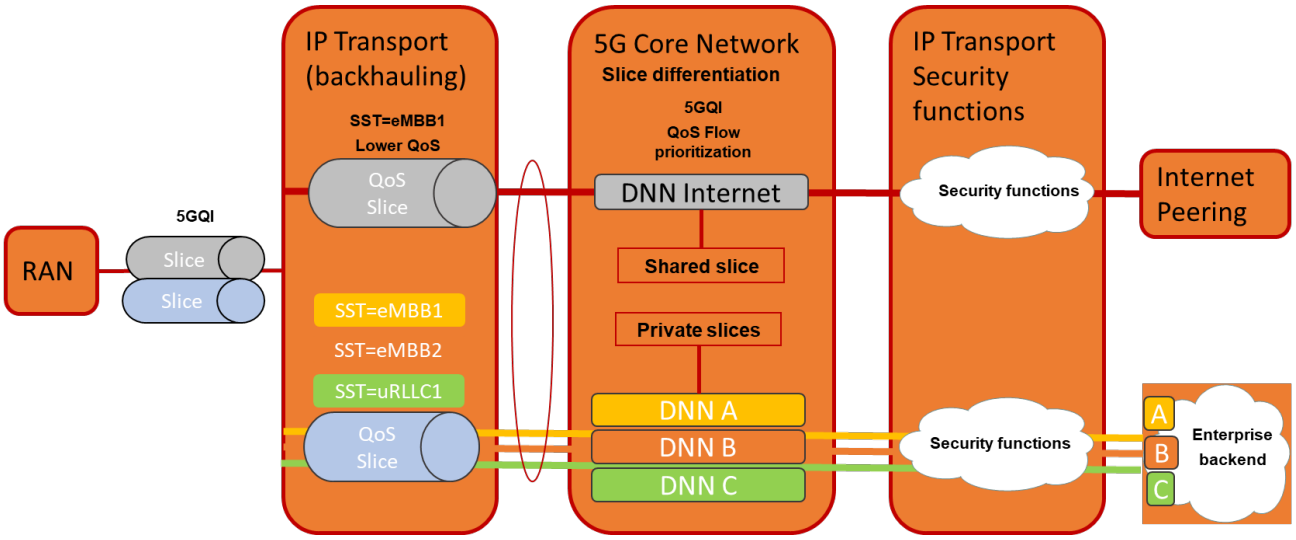

Figure 4.8: ORO Testbed slicing architecture

Differentiator (SD), SST identifying a type of network slice with specific capabilities and characteristics, while Slice Differentiator (SD) pointing out different network slices belonging to the same type. ORO Testbed-based examples are provided in Table 4.3 and Figure 4.9.

Network Slice Information		
Slice Name	**Slice / Service Type**	**Slice Differentiator**
NSSAI-1	1	abcdef
NSSAI-2	2	abcdea

Table 4.3: ORO Testbed slicing parameters

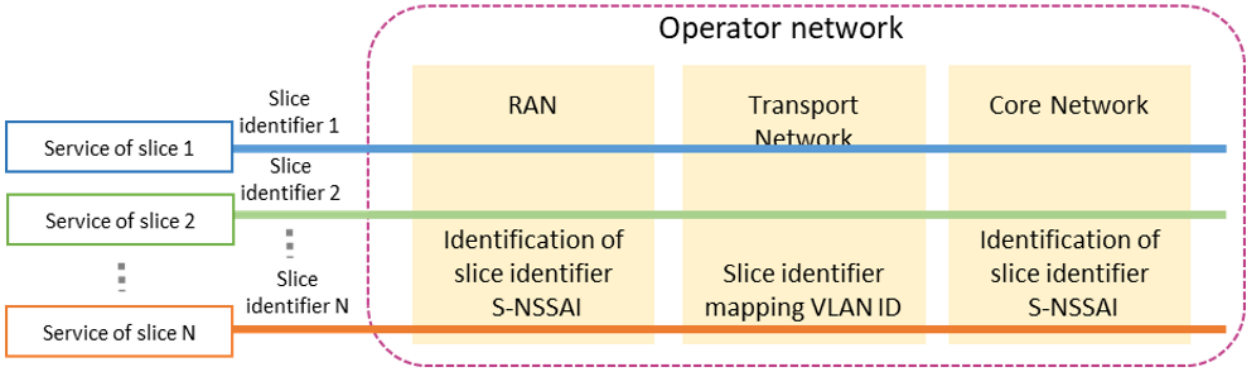

Figure 4.9: 3GPP 5G slicing model

Furthermore, the 5G Labs slicing architecture allows for the creation of user profiles featuring access to specific or multiple slices, as presented in Table 4.4:

Type	Slice	No. of slices in UE	Slice availability	Location access
Shared Slices	eMBBs	Mono slice	UE connects only to the mono slice	Everywhere or Restricted to enterprise
	eMBBs or uRLLC	Dual slice / Mono slice UE	UE can connect to either only one of the slices	
	eMBBs or uRLLC	Dual slice	UE can connect to both slices in the same time	
Dedicated Slices	eMBBs or uRLLC	Mono slice	UE can connect to either only one of the slices	Restricted to enterprise
	eMBBs or uRLLC	Dual slice	UE can connect to both slices at the same time	

Capabilities	VNF capabilities(up to)			VMs/CNFs Total Number	Total Virtualized Infrastructure resources			
Bucharest & Iasi	16	32 GB	40 GB	$>$ 200	1600/ 3200	16400	8000	6 / 12

Table 4.5: Overview of VNF capabilities and infrastructure resources.

The ORO Testbed edge computing ecosystem supports multiple types of applications, ranging from bare metal solutions to virtualized or containerized variants as showcased in Figure 4.10. The resulting modularized services, flexible and adaptable, can leverage on the configurable networking capabilities of the testbed, on the fast deployment cycles, and dynamic services launched in the network. In this approach, the applications functions may be instantiated, terminated, configured, scaled in/out and updated more rapidly compared to traditional bare-metal implementations, to support the on-demand communication services instantiation, also including configuration of specific network slices to support the applications requirements.

Development and deployment tools implemented in ORO testbed

- **OpenStack** is the cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, providing standard IaaS functionality, orchestration, service management and user applications deployment, Cloud Infrastructure for Virtual Machines, and Containers;
- **Docker** is the open platform for developing, shipping, and running applications, by separating the applications from the infrastructure so the software can be quickly delivered; an application is running in a loosely isolated environment called a container, for a fast-consistent app deployment;
- **Kubernetes** is an open-source system for automating deployment, scaling and management of containerized applications; Kubernetes is the cloud native applications orchestrator;
- **OSM** is the open source ETSI NFV Management and Orchestration (MANO) stack for E2E network service orchestration for telco services; OSM key aspects are the Information Model aligned with ETSI NFV SOL006 [10], the unified Northbound interfaces (NBIs) based in SOL005 [9], Network Services extension across multiple-domain and Life Cycle Management of Network Slices;
- **CI/CD** or Continuous Integration and Continuous Delivery is the framework for creating a build and running automated software application tests and to extend the integrations to deploy all code changes to a testing and production environment;
- **Prometheus** is an open-source monitoring and alerting system that collects metrics as series data; the data is stored on a single server node, the data collection is pulled/pushed and multiple modes of graph are supported; the tool works well for machine centric monitoring or monitoring of highly dynamic service-oriented architectures;
- **Grafana** is the open-source multi-platform analytics and interactive visualization web application, allowing dynamic dashboards creation for centralized data analysis, visualization and alerting.

4.1.2.3 Monitoring Capabilities

A key component of ORO Testbed is the network and services monitoring capabilities, through the collection collection of a variety of metrics, benefitting by different open software tools used for monitoring - Prometheus and Grafana - and data presenting through dashboards. In general, the monitoring stream is focused on three main areas:

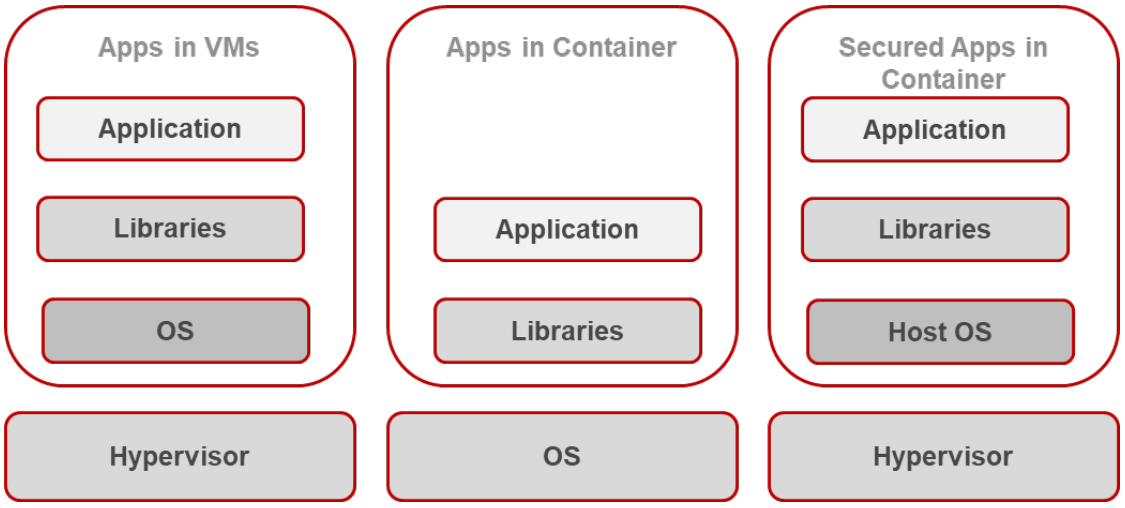

Figure 4.10: ORO Testbed supported Virtualization Layers

- **Infrastructure monitoring**: collecting the data from the infrastructure components, e.g., UE, RAN, CN, Transport, virtualized infrastructure, MEC;
- **Network and service performance monitoring**: evaluating network resources, services status, users;
- **Devices monitoring**: collecting data related to the traffic volumes specific to certain network slices, as well as information related to the radio link quality.

In this approach, ORO Testbed performs the monitoring process, for resources and elasticity profiling, testing and services validation, gathering general information about network infrastructure, the 5G services and the associated slicing parameters. ORO Testbed supports end-to-end (E2E) service monitoring, across the service's lifecycle and the associated metrics and KPIs being collected and further used for preventive activities, network control, quality measurements, fault detections and E2E validation. ORO Testbed monitoring framework is highlighted in Figure 4.11 while the monitored parameters are depicted in Table 4.6.

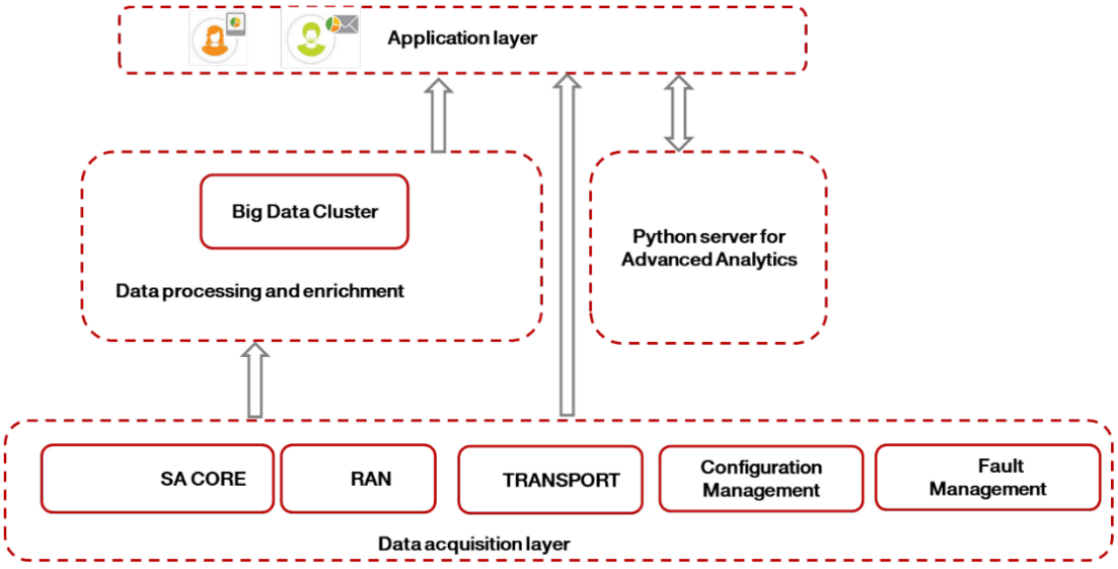

Figure 4.11: ORO Testbed Monitoring Framework

Network Domain Overview			
Radio	**Core**	**Transport**	**Monitoring**
• 5G SA traffic volumes • 5G SA Call Setup • 5G SA Drop Call • Average reported CQI • Average number of Users • Physical resource Blocks Load • 5G SA average user throughput	• 5G SA Core REST service • List of alarms • Root cause / impacts • Tree causes / impact for a specific alarm • Network performance monitor in real-time or periodically	• Troubleshooting • Ports, interfaces, LSPs • Link utilization • Latency optimization • TCEs on errors, discards, utilization • IP traffic usage	• Grafana monitoring tool • Internal tools

Table 4.6: ORO Testbed Available Metrics

4.1.2.4 3rd Party Integration Capabilities

ORO Testbed provides an onboarding platform, developed in the context of Horizon 2020 VITAL-5G project[1] and implemented as a flexible appliance which can be adapted to serve and streamline the specific needs of 5G-enabled applications developers and third party experimenters, being focused on the creation, deployment, management and validation of applications, including service and network monitoring, slice inventories, testing capabilities and orchestration.

By providing an integrated web portal, where the 5G-enabled application developers can select the needed resources from the service catalogues, ORO aims to ease the integration process specific to applications that require novel 5G services. The provided platform also helps with vertical service Life Cycle Management (LCM), blueprint validation, service testing and monitoring, results analysis, AI-based diagnosis, and multi-slice inventory management, its architecture being depicted in Figure 4.12.

Based on this platform, the testbed infrastructure can interact with various 3rd party testbeds through a unified South-Bound Interface (SBI) implemented through common APIs, while the translation between the specific interfaces implemented by the testbed management, monitoring and orchestration platforms are handled through testbed specific plugins to be implemented by 3rd parties.

Such plug-ins, when required, could implement the translation between the messages, protocols and information models adopted in the testbed and the common messages expected at the unified SBI of the ORO Testbed unified platform. This approach enables the seamless interoperability of the various 3rd party testbeds with the testbed centralized platform.

Table 4.7 specifies the main functionalities exposed by the 5G Labs testbed and the corresponding abstract interfaces. In summary the following functions are supported:

- **Network slice management**: for creation, selection and configuration of network slices in the 5G infrastructure deployed in the testbed (Slice_conf interface);
- **Onboarding**: deployment of 5G-enabled applications, VNF packages and network service descriptors for vertical services through the orchestration platform connected to the edge/cloud computing infrastructure of the testbed (Nfvo_cat interface);
- **Life cycle management**: provisioning of network services for the instantiation of vertical services composed by 5G-enabled applications (Nfvo_lcm interface);

[1] https://www.vital5g.eu/

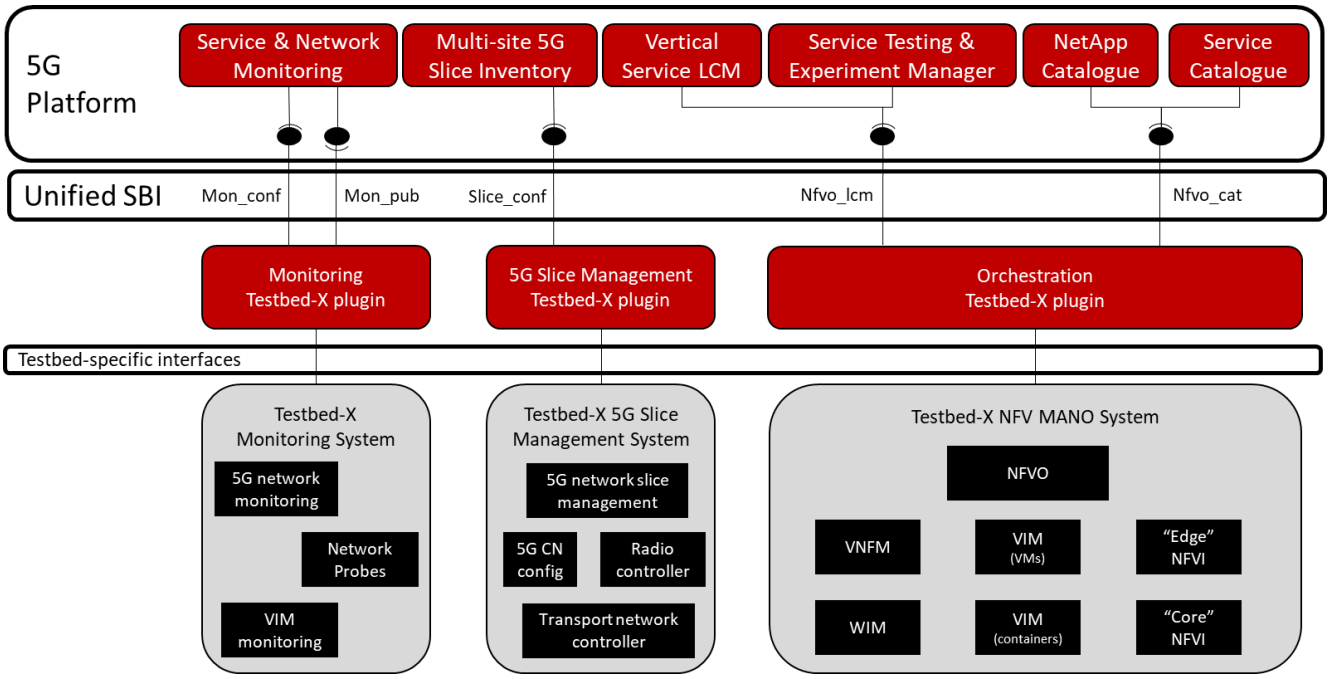

Figure 4.12: Interfacing between ORO Testbed and 3rd Party Testbeds

- **Monitoring**: configuration (Mon_conf) and retrieval through a publish/subscribe mechanism (Mon_pub) of monitoring data, for both service and infrastructure KPIs, including 5G network KPIs and metrics related to the consumption of computing resources.

5G Labs testbed	Interface	Reference standards	Implementation
NFV MANO system	Nfvo_cat: management of VNF packages and Network Service Descriptors	ETSI NFV SOL 006 [10] for NFV descriptors ETSI NFV SOL 005 [10] for REST API on management of VNF packages and NSDs	OSM 10 NBI API for: • VFND Details • NSD Details • PDU Details
	Nfvo_lcm: Management of the life-cycle of NFV Network Services and configuration of their virtual functions	ETSI NFV SOL 005 [10] for REST API on lifecycle management of NFV network services Non-standard extensions for configuration of VNFs, adopting the ETSI OSM mechanisms for VNF configuration.	OSM 10 NBI API for: • NSLCM Details OSM based Day-1 and Day-2 operations for VNF configuration
5G slice management system	Slice_conf	3GPP TS 28.531 [2] for REST API to create Network Slice Instances 3GPP TS 28.541 [1] for the information model on network slices	Implementation in progress
Monitoring System	Mon_conf Mon_pub	-	REST APIs for monitoring configuration (implementation in progress). Publish/subscribe mechanism for publication of monitoring data, based on a Kafka bus, adopting specific information models

Table 4.7: ORO Testbed Onboarding Platform Interfaces

4.1.3 ORO Testbed Future Developments, Upgrades and Extensions

In the context of the CASTOR project targeting the implementation of more secure communication infrastructures, the testbed will implement an overlay trust-path-routing tunnel between Bucharest and Iasi facilities, allowing for the transport of CCAM messages coming from commercial and simulated OBUs and RSUs, as showcased in Figure 4.13. To support this V2X application, ORO's testbeds will be fitted with Commsignia provided 4x RSUs and 2x OBUs which will be 5G-enabled and programable through the associated SDKs. Moreover, novel CISCO NX-OS virtual appliances, allowing for the customization of the trust-path-routing algorithms based on the Key Exploitable Assets, developed in the CASTOR project, will be integrated within the existing transport network infrastructure.

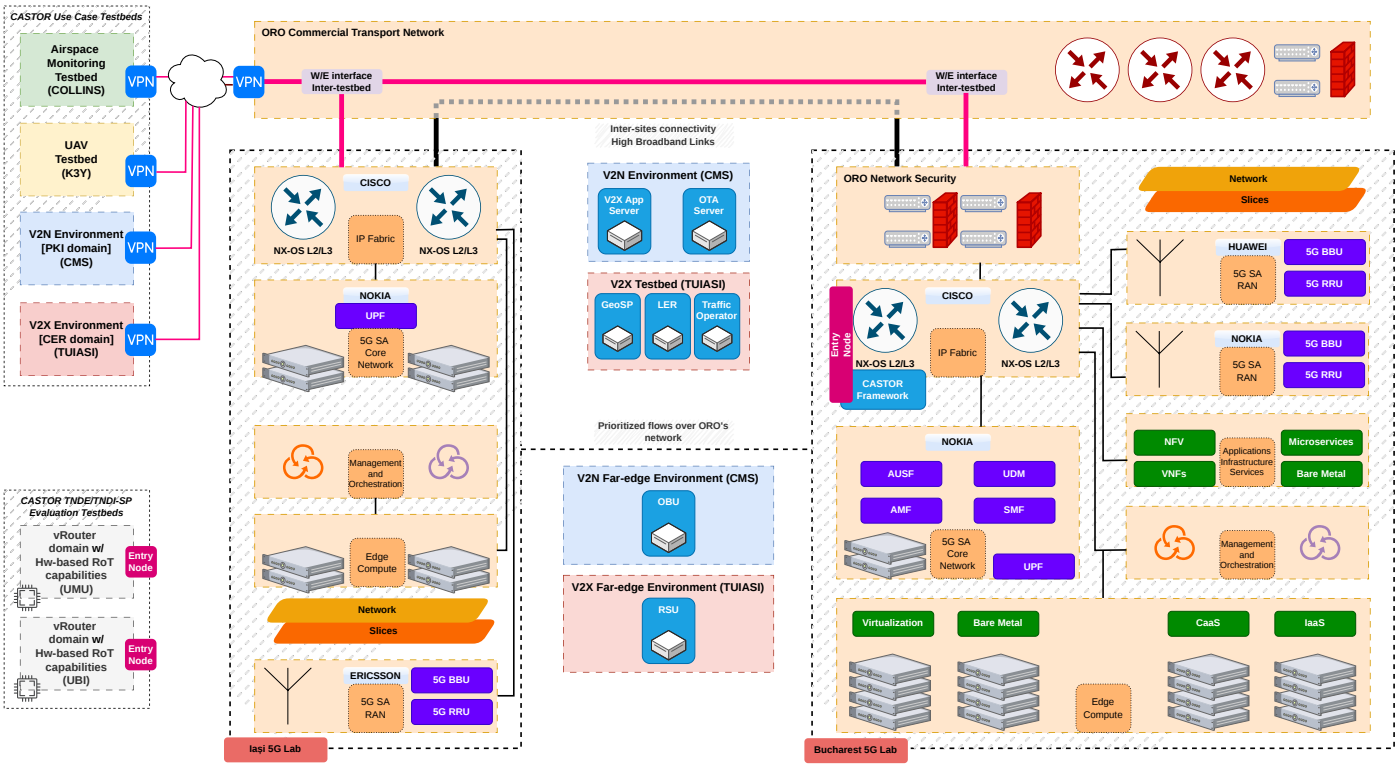

Figure 4.13: ORO Testbed architecture for CASTOR experimentation

4.2 CASTOR Testbed for TNDE/TNDI-SP Development and Evaluation

A core evaluation dimension of the CASTOR integrated framework is the in-router trust extensions of the TNDE ecosystem. As highlighted in D3.2 [7], the TNDE incorporates a Trusted Computing Base (TCB). While agnostic to specific Root-of-Trust (RoT) technologies, the host infrastructure layer must provide a specific set of fundamental security functions (see Chapter 3 in D3.1 [6]). These capabilities anchor the CASTOR TCB to the physical hardware, enabling it to securely implement the necessary TNDE mechanisms required for trusted path routing (e.g., trust establishment and verifiable runtime trustworthiness evidence collection). To address these needs, the CASTOR Testbed is extended with a complementary facility for the end-to-end evaluation of the TNDE mechanisms on top of hardware-based RoT guarantees. This environment consists of two distinct sites operated by consortium partners UMU and UBI (see Figure 4.14). These sites provide the hardware-based RoT capabilities necessary to benchmark TNDE operations across the forwarding plane. This evaluation ranges from the secure provisioning of key hierarchies (managing cryptographic materials backed by the TCB) to the performance assessment of the

novel cryptographic schemes introduced in [7]. Ultimately, this environment establishes the foundation for validating relevant PoCs—primarily PoC 1, detailed in Section 3.2. Furthermore, as depicted in Figure 4.13, it complements the production-ready ORO infrastructure (see Section 4.1) by interconnecting diverse forwarding planes (e.g., different ASes) with varying trust assurances.

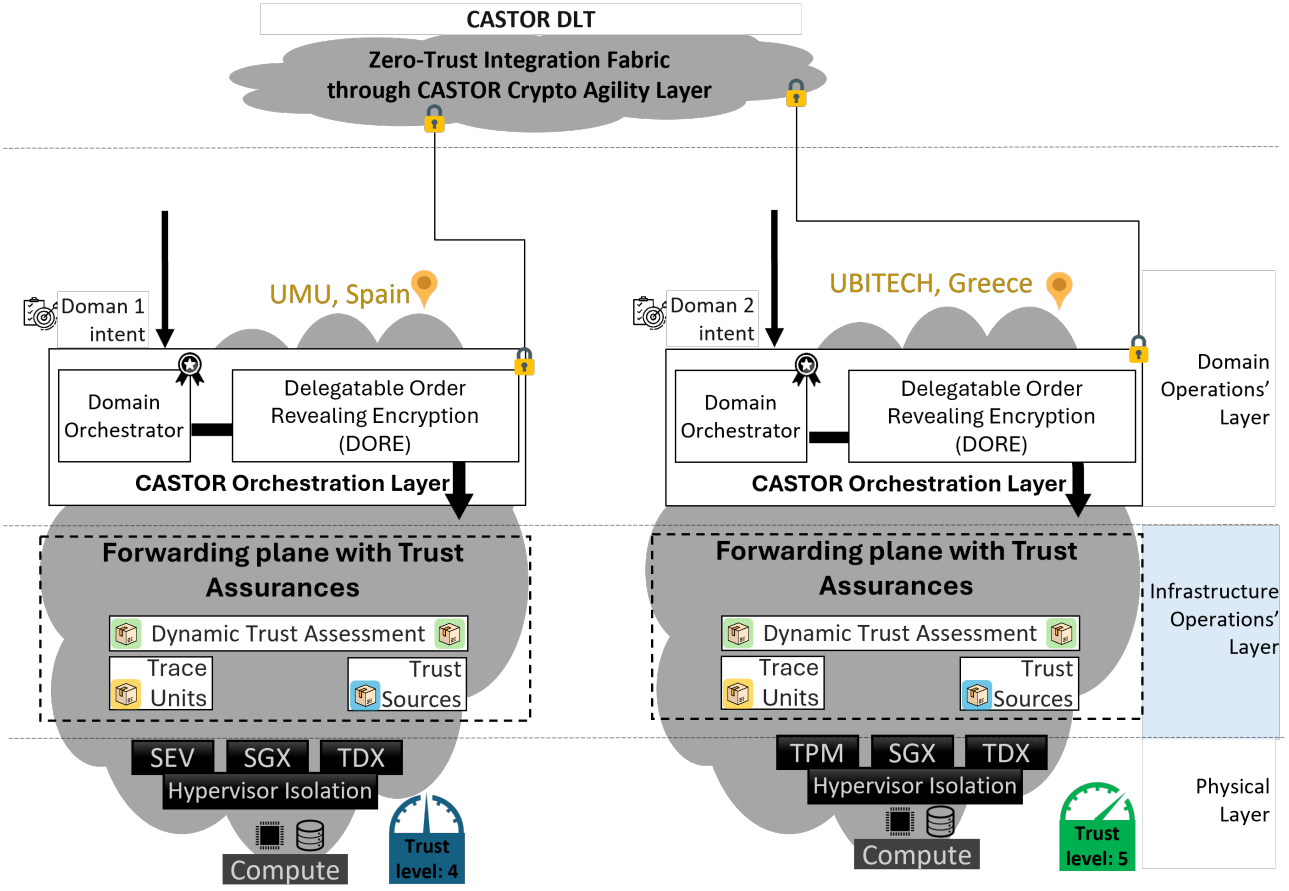

Figure 4.14: CASTOR Testbed Overview for TNDE/TNDI-SP evaluation

The UBITECH testbed leverages a rack server equipped with dual Intel Xeon Platinum 8592+ CPUs (supporting SGX and TDX), 128 cores/256 threads, and 4TB of memory (64×64GB DDR5-5600 RDIMMs). It facilitates the composition of domains with varying trust cardinalities instantiated over a compute fabric. These domains are protected by different Root-of-Trust (RoT) levels, ranging from baseline hypervisor isolation to software/hardware-rooted Trusted Execution Environments (TEEs) and pure hardware-based security elements (TPM 2.0). This setup is ideal for evaluating CASTOR TNDE artifacts in both single- and cross-domain scenarios. Consequently, the onboarding of the TNDE platform, the registration of new TNDIs, and the overall management of their operational lifecycles can be comprehensively evaluated across diverse RoT technologies.

Similarly, CASTOR will use UMU's testbed to deploy, validate, and compare CASTOR's trusted routing, attestation, and confidential-computing components across heterogeneous TEE platforms and virtualized network setups under realistic, securely connected experimental conditions. The UMU's testbed provides a heterogeneous, hardware-based Trusted Execution Environment (TEE) infrastructure designed to support research and experimentation across multiple confidential computing technologies. Our testbed integrates Intel Software Guard Extensions (SGX) and Intel Trust Domain Extensions (TDX), alongside AMD Secure Encrypted Virtualization (SEV), enabling workloads to be executed in isolated, hardware-enforced secure enclaves. This multi-vendor approach ensures broad compatibility and allows cross-platform evaluation of confidential computing primitives under realistic deployment conditions.

Furthermore, the overall TNDE testbed is supported by a robust and flexible underlying infrastructure capable of accommodating a wide range of research and experimentation requirements. Connectivity

options include high-speed fiber optic links and Layer 2 VPN (L2VPN) tunneling, enabling secure and adaptable network configurations that can replicate diverse real-world deployment topologies as needed. Time synchronization services, such as NTP and PTP servers, are available to support distributed workloads and experiments where temporal consistency is relevant. The infrastructure also offers virtualization capabilities, allowing dynamic provisioning and isolation of environments to facilitate reproducible and flexible experimental setups across heterogeneous workloads.

4.3 CASTOR Testbed for Orchestration Development and Evaluation

This section presents the final testbed environment: the WINGS testbed. Due to its enhanced infrastructure management capabilities, the testbed serves a dual purpose within the CASTOR project. First, regarding the PoC 4 narrative (Section 3.5), it will demonstrate how CASTOR's orchestration layer interacts with application-level services and far-edge compute nodes using telemetry insights combined with trust-informed network assessments. Furthermore, it will showcase how orchestration functionalities can be effectively realized through the automated deployment of service components across the computing continuum. Additionally, as noted in this chapter's introduction, the WINGS testbed will complement and streamline the validation of both the PoC narratives and the overall CASTOR integrated framework. Specifically, its enhanced orchestration features equip us to evaluate the CASTOR NSO's lifecycle management of TNDI instances (e.g., virtualized router instances) and the TNDE within each underlying infrastructure element. The remainder of this section details the WINGS testbed's capabilities. We begin with the high-level features utilized by the CASTOR Orchestration Layer to manage the forwarding plane, followed by the specific resources available to support PoC 4's application orchestration requirements.

The WINGS testbed provides end-to-end 5G/B5G functionality alongside extensive cloud and edge computing capabilities. It leverages 3GPP (Rel. 15 and beyond) architecture, transitioning from an initial Public Network Integrated Non-Public Network (PNI-NPN) with a shared Control Plane to a fully isolated Standalone Non-Public Network (SNPN) in its final phase, where all User Plane (UP) and Control Plane (CP) functions reside strictly on-premises. As depicted in Figure 4.15, the site supports various vertical domains and B5G/6G-oriented services, acting as a pre-commercial testing ground for new equipment and features. WINGS provides the comprehensive hardware, software, and configuration ecosystem required to seamlessly deploy application components across diverse use cases. Flexible and scalable infrastructure management is achieved through open-source tools—such as Proxmox, Kubernetes — which support virtual machines, containers, and serverless code execution from the cloud down to extreme edge devices like Raspberry Pis. Furthermore, the infrastructure ensures robust security and flexibility by offering strict access controls, dedicated computing, precise resource scheduling, and isolated networks. *From the perspective of the CASTOR integrated framework, these features enable the NSO component to be evaluated through the comprehensive lifecycle management of TNDIs and the provisioning of resources required to treat the CASTOR-enabled forwarding plane as a standard application workload. For instance, if a TNDI acting as a central hub between multiple Autonomous Systems (ASes) becomes a routing bottleneck, the infrastructure can dynamically reallocate resources based on real-time stress levels or the specific CASTOR TNDE features (e.g., execution of a rather complex and resource demanding FSM model for the FSM source) that must be activated at different stages of its lifecycle.*

Overall, the compute resources within the WINGS testbed comprise two powerful GPUs having two Intel Xeon Platinum 8462Y+ Processors each (60MB Cache 32 cores, 64 threads, 2.8 GHz to 4.1 GHz Turbo, 300W), a Memory of 256GB (8x32GB, DDR5, 4800MHz, RDIMM ECC), storage controllers of 2x Integrated Intel 4 port SATA controllers, storage configuration of Internal PCIe SSD (Dell Ultra-Speed Drive) and C11 Dell Ultra-Speed Drive Boot (M.2 PCIe NVMe SSD). Moreover, WINGS analyses the test results through four (4) powerful workstations, having an Intel Core i9-13900H, vPro Enterprise (24MB Cache,

14 Cores, 20 Threads, 2.6 - 5.4 GHz Turbo, 45W) processor, NVIDIA RTX 2000 Ada/8GB GDDR6 Video Card, Memory of 32 GB (2x16GB, LPDDR5, 6000MT/s).

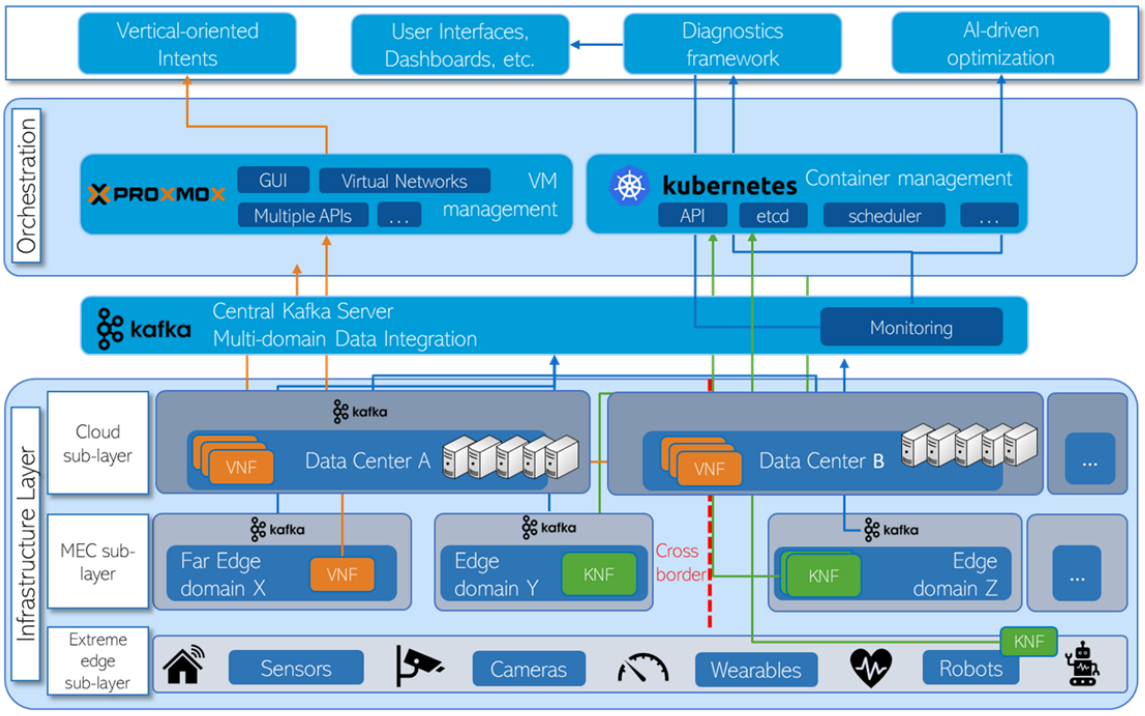

Figure 4.15: WINGS Testbed Orchestration suite

The WINGS testbed has been progressively extended with respect to its existing software, hardware, and network infrastructure to support advanced Cloud, Edge, and IoT functionalities as part of PoC 4 (see Section 3.5). The testbed facilitates the testing of services involving Automated Guided Vehicles (AGVs) and drones across various domains, including smart warehousing, healthcare, public safety, disaster recovery, and network coverage extension. Furthermore, WINGS has demonstrated advanced use cases involving Digital Twins and collaborative robots by leveraging the system's native-AI B5G/6G capabilities. Equipped with the necessary frameworks to build, test, and validate innovative applications, the facility focuses on providing diverse cloud, edge, compute, network, and device infrastructures. This comprehensive setup enables end-to-end (E2E) experimentation tailored to the requirements of the PoC. Specifically, for the purposes of this PoC, the WINGS testbed comprises the following capabilities:

- **5G Radio Access Network (RAN)**: The system leverages an Ericsson DOT-based architecture operating under 3GPP Release 15+ standards, optimized for low-latency and high-reliability communication in private industrial and research settings.
- **Software-Defined Radio (SDR) Infrastructure**: A pool of 10 Universal Software Radio Peripheral (USRP) devices is utilized, supporting custom waveform generation, real-time signal processing, and 5G NR experimentation across diverse frequency bands.
- **Edge/Cloud Compute Infrastructure**: A flexible compute environment supporting hybrid edge-cloud deployments, enabling latency-sensitive processing and orchestration of resources for distributed AI/ML workloads.
- **IoT and Device Layer**: A variety of energy-harvesting IoT devices, including environmental sensors, high-resolution cameras, and wearable technologies, are deployed to capture real-time data for energy-efficient monitoring and control scenarios.

- **Robotic Systems**: A fleet of robotic platforms, based on Robot Operating System (ROS/ROS2), includes 6 autonomous mobile robots and 2 UAVs. These platforms are equipped with advanced sensing modalities, such as LIDAR (Light Detection and Ranging), high-definition (HD) cameras, and odometry sensors, providing rich multi-modal data for navigation, SLAM (Simultaneous Localization and Mapping), and collaborative autonomy experiments.
- **High-Performance Compute (HPC) Resources**: A combination of GPU and FPGA resources is provisioned for handling compute-intensive tasks, specifically targeting AI/ML model training, inference, and large-scale data processing. GPUs are NVIDIA A100/Tesla-class, supporting CUDA, cuDNN, and TensorRT frameworks, while FPGAs offer Xilinx Alveo support for real-time hardware acceleration.
- **Energy-Aware Management Framework**: A suite for energy monitoring and dynamic device management ensures the optimization of power consumption and system performance. This framework enables multi-connectivity support (Wi-Fi 6, 5G NR, etc.) and dynamic switching between data transmission and local storage modes based on real-time energy availability and demand.

External services from other partners can be integrated into the WINGS platform via logical interfaces, following WINGS ISO-certified IT procedures. These procedures ensure secure, compliant, and efficient integration with external services, adhering to best practices for interoperability and data integrity.

- **Logical Interfaces**: WINGS supports external service integration through a wide range of logical interfaces, such as RESTful APIs, Kafka for real-time messaging, and MQTT for IoT communication. These open-standard interfaces enable seamless data exchange and interoperability between external services and WINGS infrastructure. Additionally, our system can integrate microservices, containers, and serverless functions using platforms like Kubernetes and Proxmox, allowing for flexible and scalable service deployment across the cloud and edge.
- **VPN-based Interfaces**: While direct physical connections are typically not used, we facilitate secure integration with external remote clusters via VPNs and other secure network tunneling protocols. This approach ensures that external services can connect to our infrastructure in a controlled, encrypted environment, providing secure access to critical services without the need for physical proximity.

Chapter 5

Highly Available & Secure Airspace Monitoring in Urban Air Mobility (UAM) Environments in the COLLINS Testbed

The Collins testbed provides a controlled network environment used to ground the Collins use case in a realistic system setting and to establish baseline operational behaviour prior to the introduction of CASTOR trust-aware mechanisms. The testbed is currently aligned with the Collins use case defined in Deliverable D2.1 [4] and reproduces the operational interactions between airborne assets, surveillance data sources, and ground-based coordination services within an airport network environment.

The testbed is designed to support the two Collins experimentation scenarios defined in the use case specification. *Scenario 1: On-Airport Trusted Routing Loop for Real-Time Surveillance* as depicted in Figure 5.1, focuses on the exchange of telemetry, command-and-control, and surveillance information within a single administrative network domain corresponding to the airport Zone LAN.

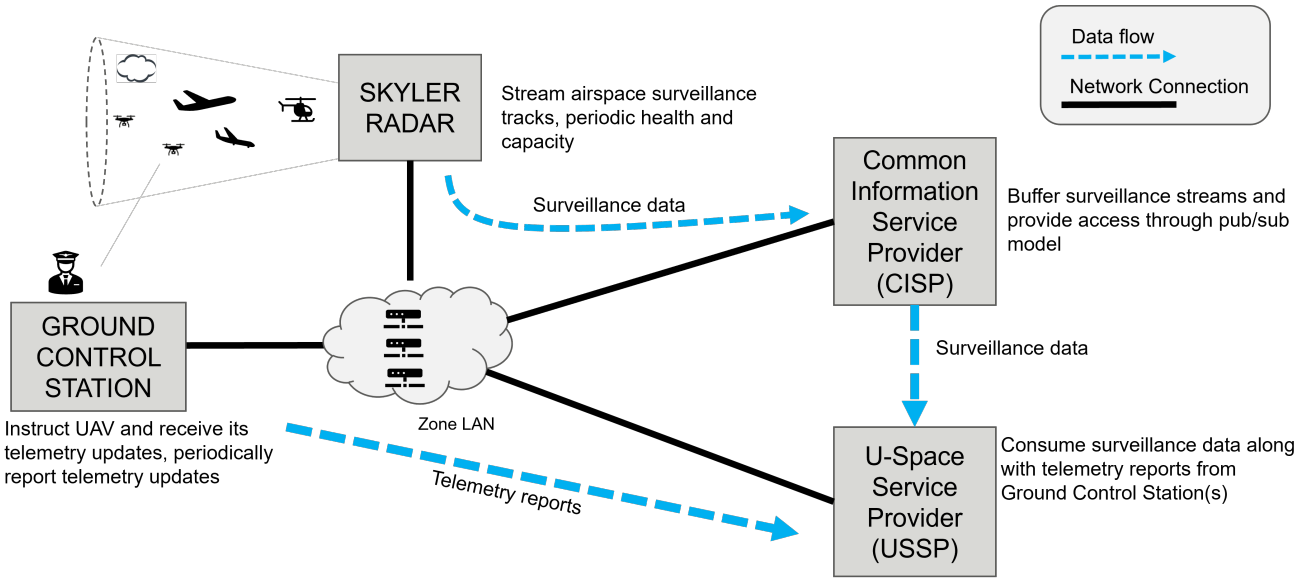

Figure 5.1: High-level deployment view of Scenario 1: On-airport trusted routing loop for real-time surveillance

Scenario 2: Collaborative Airport Operational Control Centres for Agile Decision Making extends this environment to a multi-domain deployment in which services such as the CISP and USSP are hosted in separate network domains representing remote or cloud infrastructure. In this configuration, the Zone LAN becomes one domain within a broader distributed environment, enabling experimentation with CASTOR mechanisms for trust-aware routing, orchestration, and path agility across administrative boundaries.

The Collins testbed infrastructure described in this chapter has been provisioned to support both deployment models. While the initial configuration focuses on the single-domain Zone LAN scenario, the underlying compute and networking infrastructure allows additional virtual routing domains and service placements to be instantiated without changes to the physical platform. This ensures that the testbed can evolve seamlessly from single-domain baseline experimentation to the multi-domain environments required for the later phases of CASTOR validation.

5.1 Objectives, Scope and Planning

The Collins testbed has been designed to support the experimentation scenarios defined for the Collins use case in Deliverable D2.1. Its primary objective is to provide a controlled experimental environment capable of reproducing the network, service, and data exchange interactions required for the evaluation of CASTOR trust-aware routing and orchestration mechanisms.

The testbed infrastructure is provisioned to support both of the experimentation scenarios defined for the Collins use case. The initial deployment configuration focuses on *Scenario 1: On-Airport Trusted Routing Loop for Real-Time Surveillance*, which considers the exchange of telemetry, command-and-control, and surveillance information within a single administrative network domain representing the airport Zone LAN.

In addition, the testbed infrastructure is capable of supporting *Scenario 2: Collaborative Airport Operational Control Centres for Agile Decision Making*, in which services such as the CISP and USSP are deployed in separate network domains representing remote or cloud environments. This enables experimentation with CASTOR mechanisms for trust-aware traffic steering and path agility across multiple administrative network domains.

The key objectives of the Collins testbed are therefore:

- To realise a representative network deployment corresponding to the airport Zone LAN environment described in the use case.
- To provide a controlled experimental environment capable of exercising telemetry, surveillance, and coordination traffic flows representative of operational airport scenarios.
- To support experimentation with both single-domain and multi-domain network deployments required by the Collins use case.
- To provide a stable system configuration onto which CASTOR trust-aware routing, orchestration, and optimisation components can be progressively introduced.

The testbed is designed to support realistic information flows between airborne assets, ground control functions, and surveillance data sources.

The infrastructure supporting the Collins testbed has been provisioned with sufficient compute and networking capacity to support the virtualised routing infrastructure, use-case services, and experiment coordination components required for the experimentation scenarios. The modular architecture of the testbed allows additional virtual routing domains, services, and experimental conditions to be introduced without requiring changes to the underlying physical infrastructure.

As the project progresses, CASTOR domain-local components such as orchestration, trust assessment, optimisation, and monitoring, will be introduced alongside the virtual routing infrastructure. This approach enables the testbed to evolve from a conventional network deployment towards a fully CASTOR-enabled experimental environment while preserving the core topology and service interactions required for consistent experimentation.

5.2 High-level overview of UC applications

At a functional level, the Collins use case involves airborne assets, ground control functions, surveillance data sources, and a set of coordination and information-sharing services operating within an airport operational environment. These components collectively support situational awareness, UAV supervision, and the exchange of operational information across the airport network infrastructure.

Figure 5.2 illustrates the functional architecture of the use-case applications and supporting services. The diagram presents the principal components involved in the generation, exchange, and visualisation of operational data. On the left-hand side, airborne platforms and ground control systems generate telemetry and offer visual realtime views of the deployed UAVs. In the centre of the architecture, a set of coordination and information services - including the USSP, CISP, and surveillance data sources—process and exchange information relevant to airspace awareness and operational coordination. On the right-hand side, a dashboard environment provides situational awareness to operators through a modular visualisation framework.

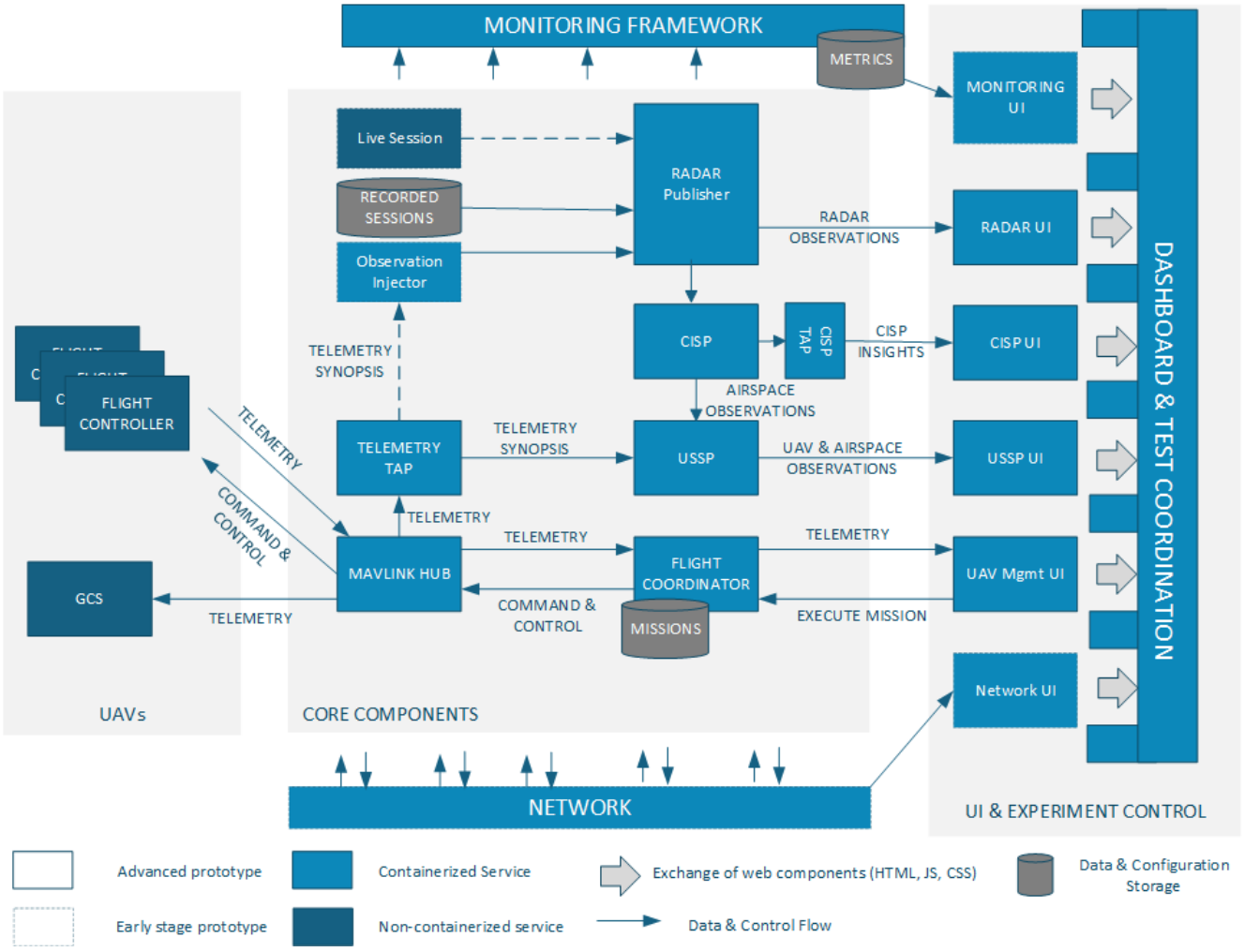

Figure 5.2: Functional architecture of the Collins use case services

Supporting the application components are two cross-cutting layers. A monitoring framework provides visibility into the behaviour of the services and the key metrics we are focused on such as communication latency, jitter and bandwidth while the underlying network infrastructure transports the telemetry, surveillance, and coordination traffic between the different functional components.

The testbed realises the functional roles through a deployment architecture that preserves the essential interaction patterns and data flows of the operational scenario, while remaining isolated, repeatable, and

configurable for experimentation. The following section describes how these functional components are realised within the physical and virtual infrastructure of the Collins testbed.

5.3 Verification concept capabilities and technologies

The Collins testbed provides the physical and virtual infrastructure used to realise the use-case applications and supporting services described in the previous section. The deployment architecture is designed to preserve the essential interaction patterns between airborne assets, ground systems, and coordination services while remaining isolated, repeatable, and fully controllable for experimentation.

The infrastructure supports both experimentation scenarios defined for the Collins use case. In its initial configuration the testbed realises the single-domain Zone LAN environment used in Scenario 1. The same infrastructure can be extended to instantiate additional virtual routing domains representing external service provider or cloud environments, enabling experimentation with the multi-domain architecture required for Scenario 2.

The testbed is deployed across a number of physical compute hosts which collectively support the execution of virtualised network infrastructure, use-case service components, simulation and traffic generation tools, and experiment coordination functions. While multiple components may be co-located on shared hosts within the laboratory environment, the deployment preserves the logical separation of roles that would typically be distributed across multiple systems in an operational setting. Figure 5.3 presents a high-level deployment view of the Collins testbed. The deployment is organised around three principal elements:

- an airborne endpoint representing the UAV and its associated telemetry source,
- a deployment substrate hosting use-case services and experiment support components, and
- a dedicated host providing a programmable virtual routing infrastructure emulating the airport Zone LAN.

This separation reflects a realistic operational decomposition while allowing the testbed to be scaled and reconfigured as experimentation progresses.

Table 5.1 summarises the principal compute resources available across the physical hosts supporting the Collins testbed. These resources support the execution of the baseline routing topology and use-case services while providing sufficient capacity to instantiate additional virtual routing domains and to deploy CASTOR framework components in later experimentation stages.

Node/Host	Role in testbed	Key software/ services	CPU	RAM	Storage	Technologies
H1 Zone LAN Infrastructure & Control Host	Zone LAN routing domain (baseline); CASTOR domain control plane (future)	Cisco XRd virtual router group (containers)	I9, 24 cores	64GB	4TB	KVM Virtualization, GPU, headroom for: • CASTOR orchestration • Trust assessment • Optimisation • Monitoring components

Node/Host	Role in testbed	Key software/ services	CPU	RAM	Storage	Notes
H2 Use-Case Services Host (part of pool)	Use-case services execution	Surveillance feed generation, ground control, coordination service stubs	I7, 8 cores	64GB	1TB	One of multiple hosts forming the Use-Case Services Host Pool
H3 Use-Case Services Host (part of pool)	Use-case services execution	Surveillance feed generation, ground control, coordination service stubs	I7, 20 cores	32GB	0.5TB	One of multiple hosts forming the Use-Case Services Host Pool
E1 Airborne Endpoint	UAV flight and telemetry endpoint	UAV flight control, physics engine and telemetry stack	-	-	-	Resource-constrained endpoint; not a scalability driver for the testbed. Scaled out to multiple E1 hosts in future experiments

Table 5.1: Collins Testbed Compute and Platform Resources

5.3.1 Overall Architecture

The Collins testbed is currently operated within a private laboratory network environment used for development and integration of the use-case software components. This environment consists of a physically connected local area network providing connectivity between the compute hosts used for UAV simulation, service components, and development activities. The physical laboratory network provides a stable platform for developing and validating the use-case software while maintaining full control over the testbed infrastructure.

For the experimentation phases of the project, the network environment will be emulated using containerised Cisco XRd virtual routers deployed on a dedicated x86-based host with hardware virtualisation support. The XRd platform enables programmable routing environments to be instantiated within the testbed and allows complex network topologies to be reproduced in a controlled and repeatable manner.

Within the Collins experimentation scenarios, groups of XRd router instances will be used to represent individual network domains. In the single-domain scenario (Scenario 1), a router group represents the airport Zone LAN environment in which telemetry, command-and-control, and surveillance information are exchanged. In the multi-domain scenario (Scenario 2), multiple router groups will be instantiated to represent distinct administrative network domains such as airport infrastructure networks, service provider networks, or remote or cloud-hosted service environments. This approach enables the testbed to emulate both single-domain and multi-domain network architectures using the same underlying infrastructure.

At the time of writing, the XRd platform has been validated within the Collins testbed environment and preliminary router instances have been deployed to confirm compatibility with the host infrastructure and container runtime environment. The full routing topologies required for the experimentation scenarios will be instantiated using deployment templates provided by CASTOR partners. These templates allow groups of routers representing complete routing topologies to be deployed in a consistent and repeatable manner.

The testbed architecture therefore combines a stable physical laboratory environment used for software development with a programmable virtual routing environment used to emulate operational network topologies. This approach allows the Collins testbed to reproduce realistic network behaviour while maintaining the flexibility required to introduce CASTOR trust-aware routing, orchestration, and policy mechanisms during later stages of experimentation.

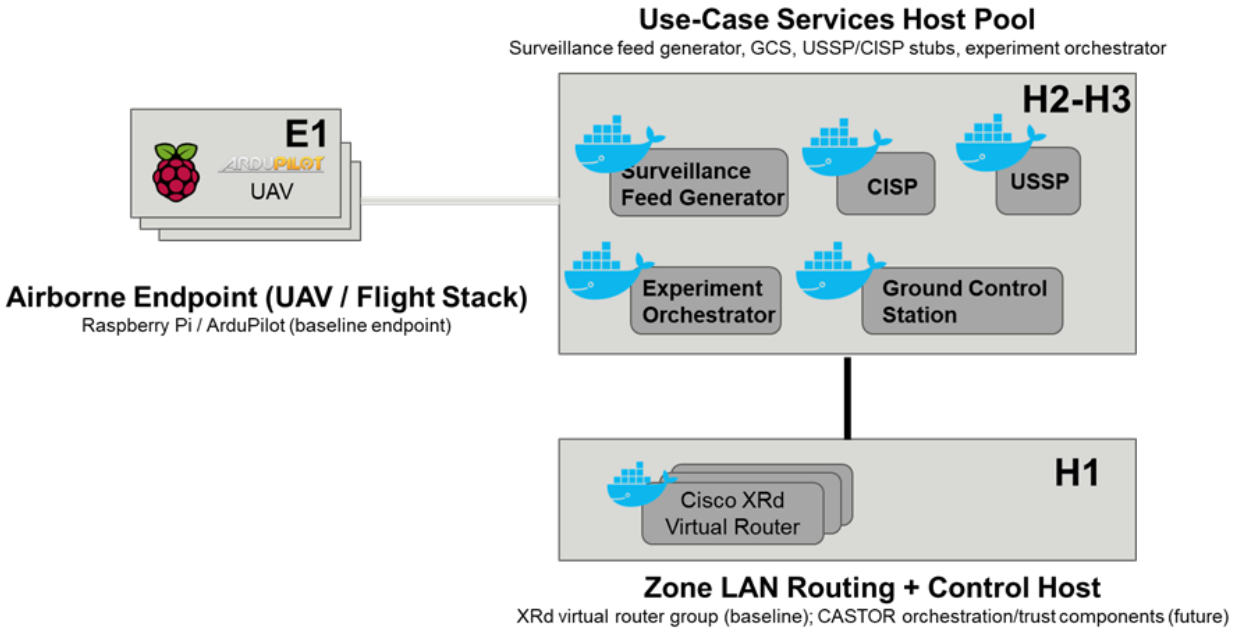

Figure 5.3: High-level deployment view of the Collins single-domain testbed

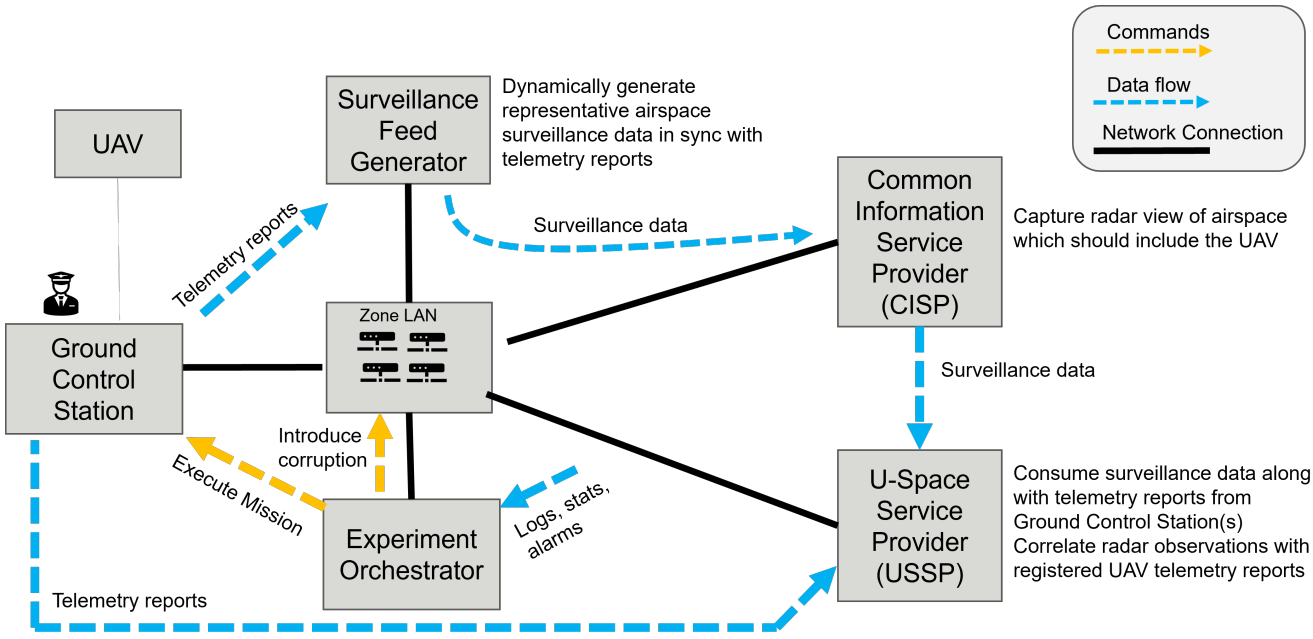

Figure 5.4: Functional decomposition of the Collins testbed for Scenario 1, showing separation between use-case components and network infrastructure within a single Zone LAN domain

5.3.2 Requirements and Technologies

The testbed is supported by a small number of dedicated hosts connected to the segregated network:

- A primary infrastructure host used to deploy the virtual routing topology and supporting network services.
- Additional auxiliary hosts available for hosting use-case components such as surveillance data generators, ground control functions, analytics, and monitoring tools.

This hosting model allows functional components to be co-located or distributed as required by the experiment configuration, while preserving architectural separation between network infrastructure and use-case elements.

The testbed environment relies on a combination of containerised services, virtualised routing infrastructure, and UAV simulation software in order to reproduce the operational data flows required by the Collins use case. Container-based deployment approaches allow services and routing components to be instantiated and reconfigured in a controlled and repeatable manner while running on shared laboratory hardware.

Airborne platform behaviour is reproduced using a UAV simulation environment capable of generating telemetry and radar streams representative of an aerial platform operating within the airport operational environment. These telemetry streams are consumed by coordination services deployed within the testbed, enabling realistic bidirectional information exchange between airborne assets and ground-based systems.

The use of virtualised routing infrastructure together with containerised service deployment provides a flexible platform for constructing both single-domain and multi-domain network environments required by the Collins experimentation scenarios.

5.3.3 Realization of each UC application service

An airborne platform is emulated using a Raspberry Pi running a flight control stack with local physics engine, connected directly to the testbed network. The platform generates live telemetry streams, including position, status, and health information, representative of an aerial asset operating within the airport perimeter.

Telemetry and command-and-control information are exchanged between the airborne platform and ground-based control functions across the emulated Zone LAN. This allows the testbed to exercise realistic bidirectional communication flows traversing the routing infrastructure, without embedding assumptions about specific middleware or tooling.

To represent situational-awareness traffic within the airport environment, the testbed incorporates a surveillance data source producing airspace observation tracks in a standardised format. In the baseline configuration, this data source is capable of generating representative observation streams that reflect the aircraft presence and movement within the monitored area.

The surveillance component is designed to evolve from replay-based data generation towards dynamic track construction, aligned with live telemetry from the airborne platform. This supports increasingly realistic coupling between telemetry flows and surveillance observations, while maintaining full control over data generation for experimental purposes.

5.4 Use-Case Software Readiness and Staging

The testbed is intended to support the progressive realisation of the CASTOR on-airport trusted routing use case, beginning from a baseline operational configuration and evolving towards full CASTOR-enabled experimentation in later project phases. As such, the availability and maturity of use-case software components is intentionally staged.

At the time of this deliverable, the primary objective is to establish a credible baseline environment against which CASTOR-enabled behaviour can later be compared, in line with the KPI definitions introduced in Deliverable D2.1. This baseline reflects a conventional, non-CASTOR operational setup for on-airport surveillance and command-and-control traffic.

In Figure 5.5, we present the current status of the testbed UI that we have been developing to showcase the operation of the Collins use case scenarios alongside CASTOR. Our current focus is on exposing the operational flow so that we can easily observe and demonstrate the future effects of network compromises. The functional architecture of the principal use-case components and their interactions is

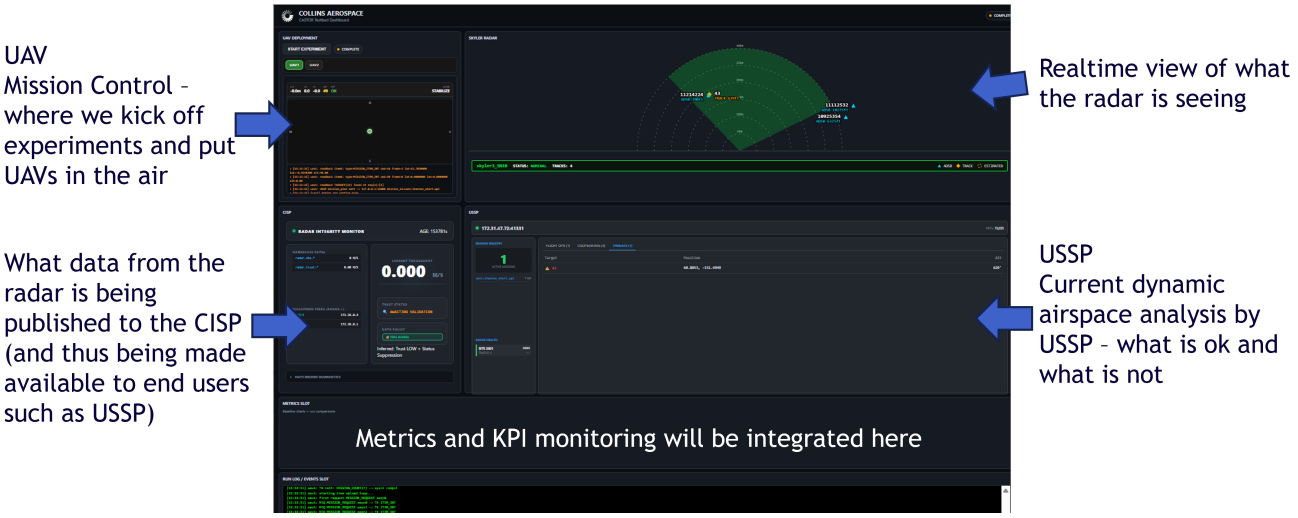

Figure 5.5: Experimentation dashboard for observing use case scenarios in operation

illustrated in Figure 5.2. The following subsections describe the current implementation status of these components within the Collins testbed.

5.4.1 Baseline software capabilities

A number of core use-case components are currently operational within the Collins testbed and provide the baseline functionality required for experimentation.

- **UAV platform and telemetry generation** An operational UAV platform based on the open-source PX4 flight control stack is deployed on a Raspberry Pi connected to the testbed network. The platform generates telemetry streams including position, velocity, and system health information representative of an airborne surveillance asset operating within the airport environment. Current development activities focus on a single UAV platform in order to establish the baseline telemetry flows and service interactions within the testbed. The architecture has been designed to support multiple UAV endpoints, although multi-UAV experimentation has not been a primary focus of the initial development stage.
- **Telemetry distribution infrastructure** Telemetry generated by the UAV is routed through a MAVLink hub acting as an intermediary broker between the UAV, the ground control station, and other services within the testbed. This architecture enables the insertion of observation points within the

telemetry path and allows telemetry streams to be forwarded simultaneously to multiple consumers, including the ground control station, the USSP, and other experimentation components.

- **Ground control functionality** Ground control capabilities are provided through a standard UAV ground control station capable of monitoring vehicle status and issuing flight commands. This provides representative command-and-control behaviour within the experimental environment while remaining independent of CASTOR functionality.
- **Common Information Service Provider (CISP)** A lightweight prototype of the Common Information Service Provider (CISP) has been implemented to support the distribution of surveillance and situational-awareness information within the testbed. The prototype uses the open-source NATS messaging system to provide a publish–subscribe architecture in which radar observation data and associated system information are distributed via topic-based messaging channels.

  Separate messaging topics are used to organise airspace observation reports, radar health information, and metadata associated with the surveillance feed. In addition to operational data, the architecture allows the inclusion of information describing the trustworthiness of the network path between surveillance data sources and the CISP. In the current prototype this trust indicator defaults to a high-trust state, but the structure is designed to support future integration with CASTOR mechanisms that will enable real-time trust assessment of the communication path.
- **U-space Service Provider (USSP)** A prototype USSP service has been implemented within the testbed environment. The service consumes UAV telemetry reports and surveillance observations received via the CISP and performs basic correlation between known UAV positions and radar observations. The current prototype highlights aircraft detected by radar that are not reporting telemetry, providing an initial representation of airspace awareness functionality.
- **Surveillance data generation** To represent situational awareness information within the airport environment, the testbed incorporates a radar data adaptor capable of ingesting airspace observation tracks from a Collins Skyler radar source. The adaptor can operate using pre-recorded radar sweeps, simulated radar inputs, or potentially live radar feeds, allowing representative surveillance traffic to be introduced into the system.
- **Mission coordination** An experimental mission coordinator service supports the configuration and upload of flight missions to deployed UAV platforms and provides monitoring of mission progression during flight operations.
- **Experimentation dashboard** An initial experimentation dashboard prototype provides a unified view of the components operating within the testbed and their real-time activity, enabling operators to observe the behaviour of the experimental environment during scenario execution.
- **Monitoring and instrumentation** Initial work has begun to introduce observability capabilities within the testbed. Selected services are being instrumented using OpenTelemetry in order to capture service interaction data across the network. The intention is to collect detailed metrics during experimental runs, including latency, bandwidth utilisation, and message exchange timing between components. A distributed tracing and metrics visualisation platform (e.g. Jaeger) is planned to support the analysis of these metrics, and key indicators will be integrated into the experimentation dashboard to provide a consolidated operational view of system behaviour during experiments.

5.4.2 Components under development or represented as stubs

While several core services are operational, certain aspects of the use-case software platform continue to evolve as the experimentation environment matures.

One area of ongoing development concerns the coupling between UAV telemetry streams and surveillance observations. Current experiments rely on deterministic radar data inputs or replayed radar sweeps. Future iterations will introduce tighter integration between UAV telemetry and synthetic radar track generation, enabling more realistic closed-loop simulation of airspace activity.

In addition, the functionality of the prototype CISP and USSP services will continue to evolve as experimentation progresses. The current implementations focus primarily on interface behaviour and data exchange patterns rather than full operational functionality. This approach allows the testbed to capture representative service interactions while avoiding unnecessary complexity in the baseline configuration.

5.4.3 Evolution towards CASTOR-enabled experimentation

The staged development approach adopted for the Collins testbed allows the baseline operational environment to be established independently of CASTOR-specific functionality. This separation ensures that baseline network behaviour, service interactions, and traffic flows can be characterised without the influence of trust-aware routing or orchestration mechanisms.

In later project phases, CASTOR domain-local components including orchestration, trust assessment, optimisation, and monitoring services will be deployed alongside the existing testbed infrastructure. These components are expected to operate in close interaction with the programmable routing infrastructure described earlier in this section.

The modular structure of the testbed enables CASTOR functionality to be introduced incrementally on top of the existing baseline environment. This will allow experimentation to explore how trust-aware routing decisions and policy enforcement mechanisms influence service interactions and network behaviour across both single-domain and multi-domain deployment scenarios.

UC1 Evaluation Plan			
User Story	**Description**	**1st Evaluation**	**2nd Evaluation**
CA.US1a	Nominal Operation	The first experimentation round will focus on the evaluation of the network-centric requirements in a CASTOR-enabled ecosystem where trusted paths are been constructed and enforced by the PCE. This will allow for the identification of a baseline of the resource utilization. As aforementioned, this will consider testing in the COLLINS testned in a single domain topology. This will include end-to-end latency; packet loss; jitter; and throughput metrics.	During the second round of experiments, all infrastructure elements will also be equipped with TNDE-enriched capabilities leveraging the verifiable sharing of trustworthiness evidence through CASTOR's crypto agility layer. Furthermore, composite attestation for path-centric trust assessment will also be tested and compared against the baseline evaluation.
CA.US1b	Performance Degradation	Focus will be given on the time needed from the identification of an SSLA policy violation to the enforcement of a new path (over routing elements exhibiting the Required Trust Level). This will include measurement of the degradation impact on latency over the data plane.	During the second round of experiments, this will be elevated to a cross-domain topology including also the sharing of (abstracted) trust-related information through CASTOR's Blockchain infrastructure as well as the DORE crypto protocol.

CA.US1c	Trust Degradation Alert	Focus will be given on time to detect trust degradation (ATL dropout) from the Global TAF. Metrics will include alert generation latency (NetOps visibility); Accuracy of trust classification; Stability of trust state (no flapping).	Focus will be elevated to a cross-domain topology so as to measure the impact on establishing a trusted path over multiple domains.
CA.US2a	Baseline consumption (blind trust)	Application behaviour without trust awareness where all input data ar treated with the same level of certainty/uncertainty.	
CA.US2b	CASTOR tier enforcement (network side)	Focus will be given on time needed to enforce a change in the trust level of an established path profile as well as post-enforcement QoS (latency/loss/jitter versus baseline)	End to end SSLA enforcement across domains
CA.US2c	Receiver-side trust interrogation	When trust metadata is available at the received-side, measure the time needed for the interpretation of the produced trust opinions. Application adaptation to trust level.	End to end trust provenance continuity across domains and ability to distinguish per-domain trust contributions.

Table 5.2: UC1 Evaluation plan

Chapter 6

Trustworthy Communications of First Responder Mobile Units and the Compute Continuum in the ORO Testbed

The V2N ecosystem established in the ORO Testbed implements and validates CASTOR Use Case 2 (UC2), as specified in D2.1 [4], which addresses the trustworthy communication of first-responder mobile units and their interaction with distributed backend services. The use case is motivated by the operational reality that first-responder vehicles – such as police motorbikes or light emergency units – must maintain reliable and verifiable connectivity to backend infrastructure even when traversing administrative and radio access boundaries that were not designed with coordinated trust policies in mind. The use case focuses on ensuring that safety-critical vehicles equipped with compact On-Board Units (OBUs) can securely access essential services, such as Public Key Infrastructure (PKI) credential provisioning and Over-the-Air (OTA) software updates, across heterogeneous, potentially multi-operator environments.

A defining characteristic of the use case is the tight coupling between communication security and operational safety. To appreciate this coupling, it is important to understand the broader V2X communication context, even though direct V2X messaging falls outside of the deployment scope. In currently deployed first-responder operations, OBUs continuously emit Cooperative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs), which are signed using short-lived, pseudonymous certificates issued by a backend PKI. This credential refresh cycle provides both message authenticity and location privacy through pseudonymity. Any disruption or integrity compromise in the PKI provisioning workflow, therefore, directly degrades the trustworthiness of V2X safety messages, creating a causal link between network-level trust properties and road safety outcomes. Equally, OTA firmware delivery must satisfy end-to-end integrity and authenticity guarantees, since a tampered or downgraded image could silently alter safety-critical communication behaviour. These interdependencies motivate the need for enforceable trust guarantees at the path level, not merely at individual protocol endpoints.

The compute continuum relevant to the use case spans from resource-constrained OBUs – connected to handheld devices via short-range interfaces such as Wi-Fi or Bluetooth – through cellular uplinks provided by potentially different mobile network operators, to cloud-hosted PKI and OTA backend services. Each segment introduces distinct trust exposure: the OBU-to-handset link is susceptible to proximity-based attacks; the cellular uplink traverses independently administered operator infrastructure; and backend services may span multiple administrative domains with no shared trust governance. No single domain in this path has end-to-end visibility over the aggregate trust properties of the full service delivery chain. While the present deployment does not physically incorporate heterogeneous radio access segments, CASTOR mechanisms operate at the service and orchestration layers above the access technology, and the results are directly applicable to heterogeneous access deployments, including multi-operator 5G and hybrid V2X environments in which first-responder vehicles already operate.

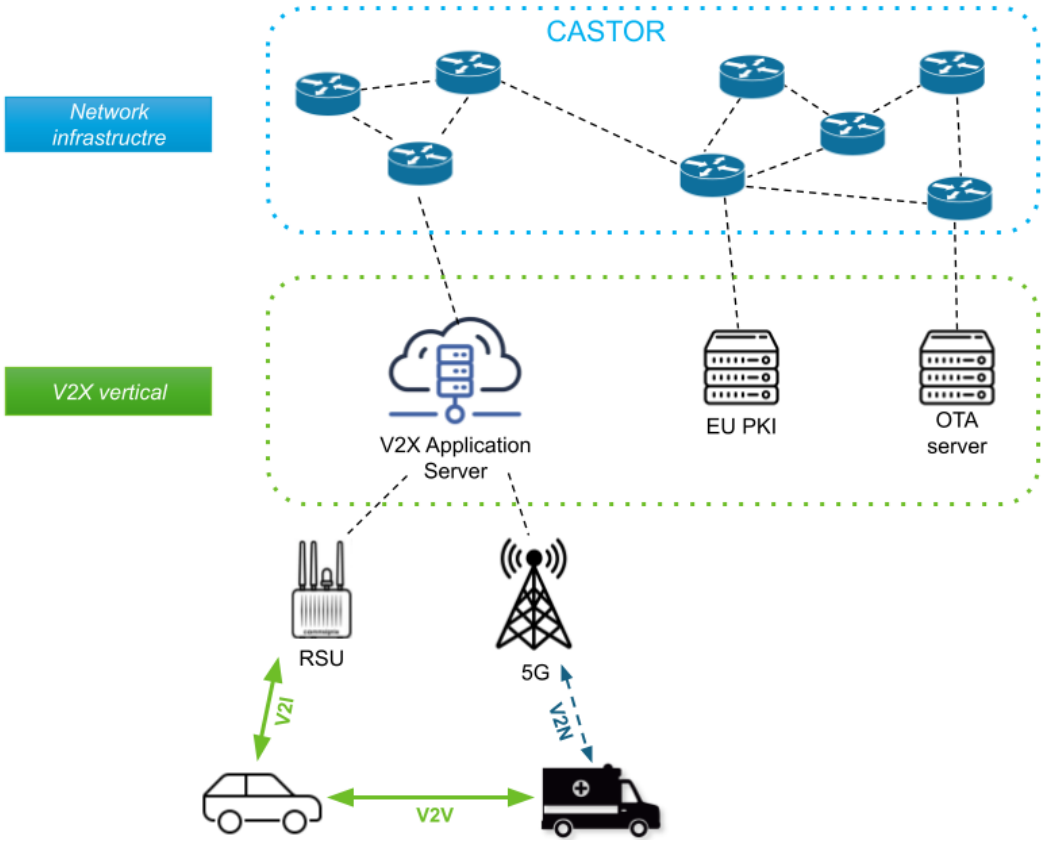

Figure 6.1: V2N ecosystem in CASTOR

The V2N ecosystem recreates the core of this operational setting by integrating OBU endpoints, OTA update servers, and PKI backend components across a network environment reflecting the multi-domain administrative structure of real deployments. Notably, the cellular network segment – representing the uplink through which OBUs reach backend services – is not part of the CASTOR-controlled domain in the current deployment and is treated as an external, uncontrolled network with opaque internal trust properties. This reflects a practical operational reality, as first-responder vehicles will typically rely on commercial mobile operators whose infrastructure cannot be assumed to fall under CASTOR's governance. The validation of the CASTOR framework in the routing plane serves as a foundation for future explorations that could encompass all network segments across the compute continuum. Setting this basis ensures that future research can move beyond current boundaries to provision fully trust-aware, end-to-end communication services. Within the domains currently under governance, CASTOR introduces Service Security Level Agreements (SSLAs), trust-aware path orchestration, cross-domain trust exposure, and continuous monitoring. SSLAs formalise per-service trust and performance obligations, providing the basis for path selection and reconfiguration. Cross-domain trust exposure enables partial attestations from individual domains to be composed into a coherent end-to-end trust view, even when intermediate segments, such as the cellular network, cannot directly contribute. Continuous monitoring detects runtime deviations and triggers re-orchestration accordingly.

As such, the deployed ecosystem provides the experimental foundation for assessing the transition from conventional best-effort V2N service delivery (as-is) to trust-orchestrated and policy-compliant communication paths (to-be), as defined in D2.1, with results and mechanisms that are directly applicable to the heterogeneous multi-operator and hybrid V2X access environments in which first-responder vehicles already operate.

The use case is decomposed into two user stories, which are directly implemented in the CMS deployment. Both share a common topology in which the OBU does not communicate directly with the PKI or OTA backend services but accesses them exclusively through the V2X Application Server, which serves

as a mediator and proxy for all interactions with the backend services. This architecture deliberately positions the V2X Application Server as the single integration boundary between the vehicle domain and the backend service domain and, consequently, as the natural entry point for CASTOR trust orchestration in the to-be configuration.

- **UC2.1 — Digital certificate download for secure V2X communication.** The OBU periodically requests a fresh batch of pseudonymous Authorization Tickets from the PKI backend via the V2X Application Server. This user story also realises the cross-domain scenario: the PKI backend is operated as a distinct administrative domain, and credential requests must traverse the inter-domain path under CASTOR trust orchestration in the to-be configuration. In the as-is scenario, this exchange occurs over a conventional best-effort network path with no trust-aware routing or SSLA enforcement.
- **UC2.2 — Over-the-Air firmware update of security-sensitive communication devices.** The OBU retrieves firmware or configuration update packages from the OTA server via the V2X Application Server. The update transaction must satisfy end-to-end integrity and authenticity requirements, as a compromised or downgraded firmware image would silently impair the vehicle's safety-critical communication behaviour. In the as-is scenario, update delivery relies on application-layer signing with best-effort transport and no path-level integrity monitoring.

6.1 Objectives, Scope and Planning

The deployment validates secure, trustworthy, and reliable V2N operations for first responder vehicles operating across heterogeneous network conditions characteristic of real deployments – conditions that exist in practice but are intentionally kept outside the direct scope of the current setup to maintain a controlled, reproducible experimental environment. Key objectives include:

- Ensure timely and reliable delivery of OTA firmware and security updates to OBUs, with end-to-end integrity and SSLA compliance.
- Maintain fresh and valid PKI credentials for secure V2X communication, including cross-domain certificate renewal.
- Demonstrate secure, auditable communication paths between OBUs, V2X Application Servers, PKI, and OTA servers, supporting dynamic path selection based on trust (CASTOR Trust Exposure Layer).
- Evaluate system performance under realistic V2N and V2N2V scenarios, including degraded or partially trusted network segments, with KPI monitoring for latency, throughput, availability, and integrity.
- Support integration of CASTOR orchestration to enforce SSLA-compliant routing, fallback SSLA mechanisms, and adaptive responses to degraded trust.
- Provide a test environment reflecting hybrid CCAM topologies, heterogeneous devices, and diverse communication technologies.

The current configuration of the V2N ecosystem deployment establishes a baseline environment that supports secure OTA updates and PKI certificate provisioning over a controlled V2N connectivity setup. While first-responder vehicles in practice already operate across heterogeneous access technologies – including 5G, ITS-G5, and hybrid V2X connectivity – these are intentionally not exercised directly within

the testbed deployment. Furthermore, the PKI system is treated as a black box, with no details on the connectivity between its components [1]. This scoping decision preserves experimental reproducibility and isolates the contribution of CASTOR mechanisms, while the architecture, methods, and results remain directly transferable regardless of the underlying access technology or the nature of the consumed service domain – whether that is a different radio access technology, a distinct PKI authority hierarchy, or any other externally administered backend service.

We are extending our solution to integrate with the CASTOR framework, allowing the V2X Application Server and its components to leverage trust-aware path orchestration, automated detection of network and trust events, and dynamic switching between Main and Fallback SSLAs. Through this interface, we can validate how CASTOR-managed paths and SSLA compliance enhance the reliability and security of OTA updates and PKI certificate management, without requiring the implementation of the underlying orchestration mechanisms.

Further planned evolution includes support for cross-domain PKI access scenarios, scalable simulation of large responder fleets for mass OTA and certificate-renewal testing, and enhanced KPI monitoring to compare baseline operations with CASTOR-enhanced behaviour. These enhancements will be applied incrementally on top of the CMS V2N ecosystem, enabling traceable interactions with CASTOR and measurable assessment of the benefits of trust-aware routing and policy-compliant network paths.

Planned enhancements to the baseline V2N ecosystem within CASTOR are:

- Advanced traffic steering and trust-aware path selection.
- Cross-domain PKI support with CASTOR-mediated trust negotiation and path orchestration.
- KPI-driven monitoring and evaluation framework for latency, throughput, SSLA compliance, and availability.

The results and experience gathered can support scalable simulation of large responder fleets for future mass OTA updates and certificate renewals, serving as fuel for large-scale fleet deployment planning and execution. The system's performance can be further improved by leveraging mechanisms beyond the CASTOR scope, such as 5G QoS and other advanced APIs, to support latency-sensitive PKI operations and throughput-sensitive OTA updates under various conditions.

6.2 High-level overview of UC applications

At a functional level, the use case involves first responder vehicles equipped with OBUs, backend service components such as a V2X Application Server, an OTA server, and a PKI, and heterogeneous communication networks interconnecting these entities, as shown in Figure 6.2. The functional architecture defined in D2.1 captures the interaction patterns required to support secure certificate provisioning, firmware updates, and continuous service availability under heterogeneous and potentially multi-domain network conditions.

[1] Because CASTOR-enabled domains ensure trust-aware Traffic Engineering (TE) provisioning up to the PKI boundary, the terminating PKI domain itself is assumed to be inherently trusted for the purposes of this validation. Consequently, the PKI system is treated as a black box, rendering the internal connectivity details between its components out of scope for the validation and evaluation of the CASTOR integrated framework.

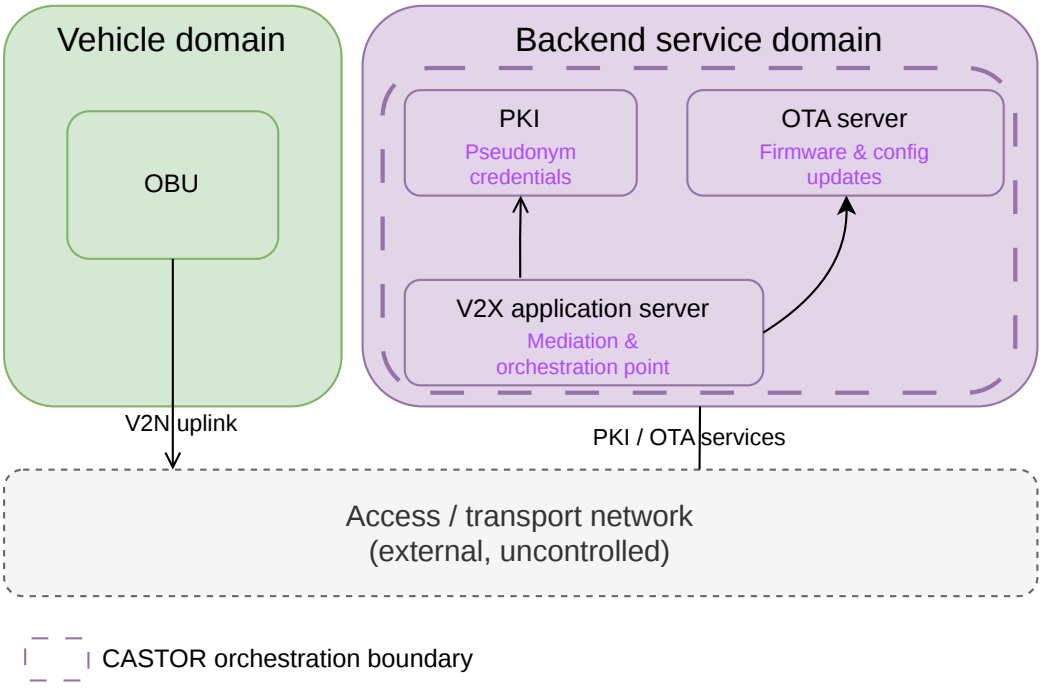

Figure 6.2: Functional overview of the CMS ecosystem in CASTOR

The deployed ecosystem fulfils these functional roles while preserving the essential communication paths and control flows of the operational scenario. Vehicle OBUs interface with the V2X Application Server over V2N connectivity. The backend components – the OTA server and PKI – operate within distinct administrative or trust domains with respect to each other. Details of the domains and support for the multi-domain scenario are described later in Section 6.3.1. The V2X Application Server acts as the mediation and orchestration point, enabling certificate management, update coordination, and integration with CASTOR trust-aware mechanisms.

The logical separation between the vehicle domain, access network, and backend services reflects the realistic operational boundaries of the target scenario, while remaining configurable for controlled experimentation. This mapping ensures that the deployment captures the architectural characteristics relevant for validating trust-aware routing, SSLA enforcement, cross-domain communication, and performance evaluation, without binding the implementation to a specific operational fleet or commercial infrastructure.

6.3 Verification concept Capabilities and Technologies

The CMS ecosystem integrates the vehicle On-Board Unit, backend services running on cloud and edge computing infrastructure, and the network fabric provided by partner organizations, to evaluate secure V2N services and validate the goals and architecture described in the preceding sections. It is designed to realistically model the operational entities and communication relationships of the use case while maintaining a carefully controlled and reproducible experimental environment. The deployment supports all user stories defined for the CMS use case, covering both single-domain and multi-domain operation within the CASTOR framework. Table 6.1 summarises the primary components, their functional roles, and the key technologies and resources associated with each.

Node/ Host	Role in testbed	Key software/ services	CPU	RAM	Storage	Technologies
V2X Application Server	Coordination and mediation point for V2X backend services; primary integration point for CASTOR trust-aware mechanisms	V2X client session management, OTA coordination, PKI request proxying	2 vCPU	4 GB	40 GB	Docker container runtime; REST/HTTPS interfaces
On-Board Unit (OBU)	Vehicle-side V2X client; initiates PKI credential refresh and OTA firmware update transactions	Embedded Linux environment, OTA client, PKI client	4 core CPU	4 GB	30 GB	5G capability, Proprietary V2X stack, Docker runtime
OTA server	Secure management, storage, and distribution of firmware/software/-configuration updates for the OBU	Software repository/artefacts, Device repository, Update server	2 vCPU	4 GB	40 GB	RESTful APIs and AMQP-based message brokering
PKI server	Certificate authority for OBU devices	ETSI TS 102 941-compliant PKI backend [2]	*To be determined*			

Table 6.1: Resource summary of deployed components

6.3.1 Overall Architecture

The deployed components of the V2N ecosystem are distributed across physically and administratively distinct computing resources, unified into a coherent test environment through virtualized network overlays. This distribution is intentional: it reflects the multi-domain communication structure of the operational scenario, in which vehicle endpoints, access infrastructure, and backend services belong to different administrative and trust domains, and do not share a common network management plane. Figure 6.3 shows a high-level deployment overview of the CMS V2N ecosystem. Figure 6.4 details the deployment further by mapping the V2N ecosystem components onto the resources available at the ORO testbed. The V2X Application Server is hosted on the ORO Lab edge computing platform, placing it close to the CASTOR orchestration components and minimising communication overhead for trust signalling and path management. The PKI backend and OTA server are each connected to this environment via dedicated VPN tunnels, a topology that deliberately realises the multi-domain scenario defined in the use case: each backend service appears as a logically distinct domain with its own administrative boundary, reachable via a controlled inter-domain path rather than a shared local network. To support the Single and multi-domain scenarios, the V2X Application Server shares the same CASTOR domain with the OTA server, and the PKI service is kept in a separate domain (as shown in Figure 6.3). In the primary deployment configuration, the OBU, as the vehicle-side endpoint, connects to the backend services via the V2X Application Server over the 5G access network. For earlier integration testing and scenarios where radio access conditions are not under evaluation, the OBU can alternatively be connected through the VPN overlay, allowing functional correctness of OTA and PKI workflows to be validated independently of the access network segment.

[2] Since CASTOR evaluates the network path up to the PKI domain rather than the PKI itself, the specific backend implementation does not critically affect the framework's assessment. Therefore, an internally operated CA (e.g., EJBCA [12]) provides an initial baseline for evaluating the PKI scenario, while exploring a transition to a sandboxed third-party C-ITS authority as a parallel activity.

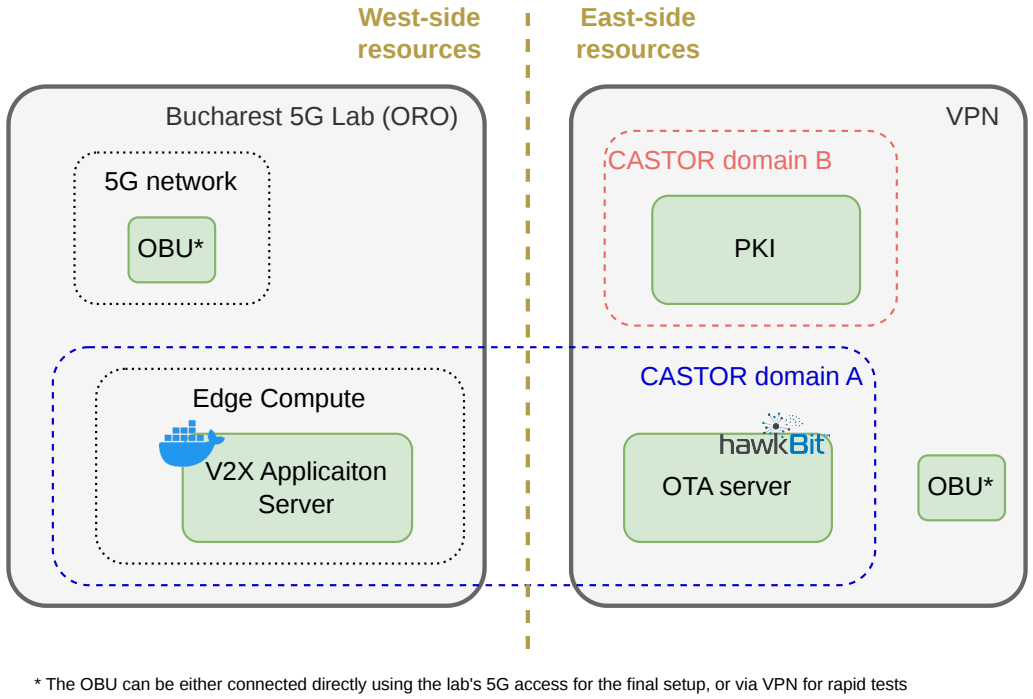

Figure 6.3: High-level V2N ecosystem deployment diagram

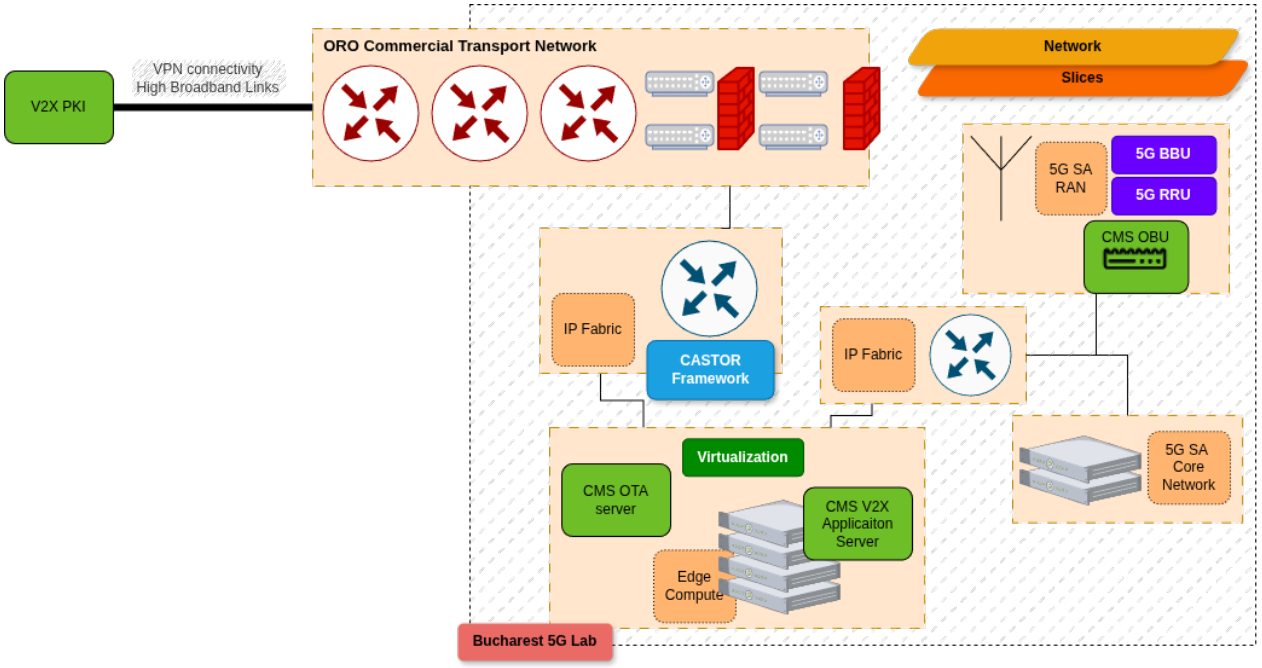

Figure 6.4: Mapping of V2N ecosystem components onto ORO testbed infrastructure[3]

[3]Note that the communication between the V2X Application Server and the OTA server is done via the CASTOR framework (i.e., CASTOR-enabled network topology.

6.3.2 Requirements and Technologies

The deployed ecosystem is built around four foundational infrastructure requirements that, together, enable the use-case experimental objectives.

The first is an edge computing platform capable of hosting the V2X Application Server with sufficient processing capacity and network proximity to the CASTOR orchestration components. This platform must support containerized service deployment to ensure portability, isolation, and reproducibility across experimental runs.

The second is a 5G access network providing the primary V2N uplink for the OBU in operational test scenarios. As noted in Section 6.1, this cellular segment constitutes an external, uncontrolled domain from the perspective of the CASTOR framework in the current deployment. It provides a realistic network bearer over which trust properties must be inferred or estimated rather than directly attested.

The third is a VPN infrastructure that connects the PKI and OTA server instances, the V2X Application Server, and the CASTOR testbed infrastructure. The VPN acts as a connectivity facilitator, enabling the CMS components to participate in the testbed regardless of whether the underlying network topology spans a single domain or multiple Autonomous Systems. It also provides the fallback connectivity path for OBU access during integration and pre-deployment testing, allowing the OBU to be remotely connected to the testbed without requiring the 5G access network to be active.

The fourth is a flexible, controllable experimentation platform that supports scripted, repeatable triggering of the use-case workflows. This ensures that OTA update and PKI credential provisioning transactions can be initiated under defined, reproducible conditions, enabling systematic KPI measurement across baseline and CASTOR-enhanced configurations.

6.3.3 Realisation of UC Application Services

The primary client actor in the use case is an OBU running an embedded Linux environment. The device is representative of the compact, lightweight V2X communication stations targeting micromobility and light first responder vehicles that do not rely on extended OEM backend systems.

In its PKI operational role, the OBU is enrolled in a sandbox European C-ITS PKI and is capable of requesting and refreshing pseudonymous certificates using the standardised ETSI TS 102 941 credential management protocol. This standard defines the interaction between Intelligent Transport System (ITS) stations and the PKI components of the European C-ITS security credential management system, including the Enrolment Authority (EA) for initial device registration and the Authorisation Authority (AA) for the issuance of short-lived pseudonymous Authorisation Tickets (ATs). These pseudonymous credentials are the basis for privacy-preserving V2X message signing, and their timely provisioning is therefore operationally critical.

In its OTA operational role, the OBU can receive and apply firmware and configuration update packages from the OTA server. The update process encompasses package retrieval, cryptographic integrity verification, and controlled application, ensuring that only authenticated and unmodified update payloads are applied to the device.

During experimentation, both OTA update transactions and pseudonym download cycles are triggered through managed scripts, allowing precise control over transaction timing, frequency, and sequencing. This scripted triggering approach enables the systematic measurement of end-to-end latency, delivery success rate, and integrity verification outcomes under both baseline and CASTOR-enhanced network conditions, and supports the reproducibility required for KPI-driven comparative evaluation.

The high-level description of the key functional components is collected in Table 6.2.

Platform	Description
Vehicle OBU	In-vehicle computing platform for secure V2X communication, PKI handling, and OTA updates
V2X Application Server	Mediates V2N2V communications, PKI interface, OTA orchestration, cross-domain gateway, and Trust Awareness API integration
OTA Server	Manages firmware/software lifecycle, integrity protection, secure delivery to OBUs; supports SSLA-aware orchestration
EU PKI	Trust anchor for secure communication; manages certificate/pseudonym issuance, renewal, and revocation

Table 6.2: Overview of the V2N ecosystem services

6.4 Use-Case Software Readiness and Staging

The objective of this section is to describe the implementation state and staged evolution of the deployed software components. The target is to deploy and integrate the functional components to support the use case and progressively adopt CASTOR-enabled network capabilities via SSLA negotiation and CASTOR API interfaces. The baseline environment realises the functional architecture without CASTOR-specific enhancements, corresponding to the as-is scenario defined in D2.1 [4]. The implementation properties described below reflect the capabilities of the existing solution components.

6.4.1 Baseline software capabilities

The baseline configuration comprises the four core use case application components introduced in Section 6.3.3: the V2X Application Server, the OBU, the OTA server, and the PKI backend. Each component is independently operational in the as-is scenario, with inter-component communication over standard transport protocols and without trust-orchestrated path selection or SSLA enforcement.

V2X Application Server. The V2X Application Server is the central coordination component for all backend service interactions on behalf of the OBU. Its primary functions in the baseline configuration are device session management, service request routing, and acting as the single point of contact through which the OBU accesses the PKI and OTA services. By consolidating all backend connectivity through the V2X AS, the architecture naturally positions it as the entry point into the CASTOR domain in later project phases: CASTOR API consumption, SSLA negotiation, and trust-aware path selection can be integrated at this layer without requiring modifications to the OBU or the backend service components themselves.

Over-the-Air Update Capabilities. OTA update management is realised through a server-side campaign management platform that provides rollout control, deployment state tracking, and artefact delivery. Update packages are prepared, signed, and, if requested, encrypted at the OTA server prior to distribution. The server exposes a device-facing API over HTTPS through which OBU controllers poll for pending deployment actions, retrieve update metadata, and download firmware or configuration artefacts in chunks. Each artefact is accompanied by a cryptographic hash that the OBU verifies upon download before applying the update, providing application-layer integrity assurance independent of the transport path. The OBU controller reports the deployment action state back to the server at each stage of the update lifecycle, enabling centrally managed rollout monitoring and failure detection. In the baseline configuration, the OBU accesses the OTA server via the V2X Application Server, which acts as a proxy. Control messages are exchanged over HTTPS; artefact data is transferred over the same verified transport channel. Figure 6.5 illustrates the functional service architecture of the OTA subsystem. In the

CASTOR-enhanced configuration, the V2X AS will additionally invoke the CASTOR orchestration layer to select trust-compliant paths for OTA traffic, monitor end-to-end integrity, and activate fallback SSLA routing in the event of path degradation or trust threshold violations.

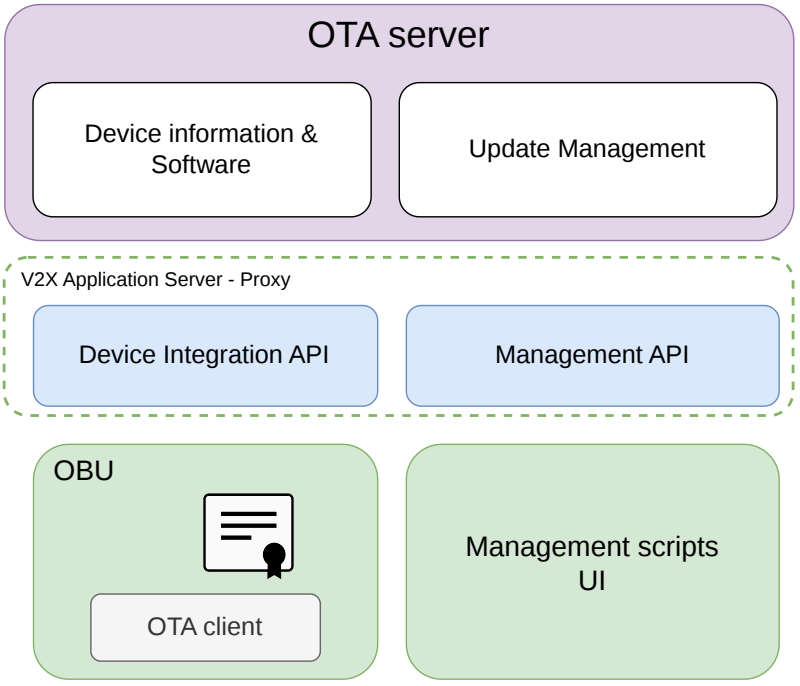

Figure 6.5: Overview of the OTA service architecture

PKI Credential Management Capabilities. Currently, there are two options under consideration for realising the PKI subsystem. It is primarily envisioned to be a sandboxed environment fully compatible with the ETSI TS 102 941 C-ITS security credential management standard [11]. The standard defines a two-tier authority structure: an Enrolment Authority (EA), responsible for the initial registration and long-term identity provisioning of ITS stations, and an Authorisation Authority (AA), responsible for issuing short-lived pseudonymous Authorisation Tickets (ATs) used for V2X message signing. In the CMS use case, the OBU is assumed to be a pre-enrolled device that holds a valid enrolment credential. The active operational scenario, therefore, focuses on the periodic AT refresh cycle: the OBU requests a new batch of pseudonymous ATs from the AA via the V2X Application Server, which proxies the credential request and response along the network path.

Pseudonym batches are rotated on a defined schedule to preserve location privacy, and the timely delivery of fresh credentials is operationally critical: a failure to renew certificates before expiry would directly impair the OBU's ability to generate authenticated V2X messages. A more detailed overview of the message flow during the certificate renewal procedure is provided in Figure 6.6. Table 6.3 summarises the service properties and quality objectives associated with PKI credential management in the use case.

From a trust orchestration perspective, the EA–AA validation exchange – in which the AA verifies the requesting device's enrolment credential with the EA before issuing ATs – represents an inter-authority communication path that is conceptually within scope for CASTOR-managed trust routing. In a fully CASTOR-governed PKI topology, this internal path could be subject to the same SSLA-based integrity and availability guarantees applied to the OBU-facing service paths, providing end-to-end trust assurance across the entire credential issuance chain. However, the internal connectivity between PKI authority components falls outside the scenario boundary: the EA and AA are treated as a single logical backend trust anchor from the perspective of the V2N ecosystem, and CASTOR mechanisms are applied to the

Property	Description
Integrity	Certificate payloads must not be tampered with in transit; cryptographic signatures are verified end-to-end.
Confidentiality	Sensitive credential material is protected from unauthorised access at rest and in transit.
Auditability / Traceability	All PKI operations are logged to support forensic analysis and compliance verification.
Throughput	Sufficient data rate to support timely batch issuance and renewal without queuing delays.
Latency	Request and response round-trips must complete within operational deadlines to prevent certificate expiry.
Availability	PKI services must remain reachable under degraded network conditions to avoid V2X service interruption.
Scalability	The system must support concurrent credential sessions for large first-responder or mixed-fleet deployments.

Table 6.3: PKI service properties and quality objectives

paths between the V2X Application Server and the PKI as a whole, rather than to the intra-PKI authority interactions. This scoping decision reflects the operational reality that PKI infrastructure is typically administered as an independent, internally secured system, and that the primary trust exposure relevant to first-responder operations lies in the network path between the vehicle and the credential service boundary, not within the credential service itself.

The second implementation option is a self-hosted simulated PKI environment deployed within the testbed using open-source CA software (e.g., EJBCA). In this configuration, both the EA and AA roles are instantiated as distinct CA entities within the same deployment, preserving the two-tier authority structure of ETSI TS 102 941 even in a simulated context. The instance must be configured with ETSI TS 103 097-compliant certificate profiles to ensure OBU client compatibility without modification to the vehicle-side software stack. As the OBU is assumed to be pre-enrolled, the self-hosted option additionally requires a one-time enrolment provisioning step – scripted as part of the testbed setup procedure – to issue the long-term enrolment credential against the simulated EA prior to experimentation. A practical advantage of this option for the multi-domain scenario is that the PKI's network exposure within the testbed VPN topology is fully controllable: the self-hosted instance can be explicitly positioned as a distinct network domain with defined connectivity boundaries, whereas the third-party sandboxed option arrives with its own connectivity configuration that may constrain the testbed topology. The PKI subsystem produces ETSI TS 102 941-compliant certificates in both cases, ensuring fully standard OBU client behaviour in the user story regardless of which option is adopted. The final implementation choice is subject to ongoing evaluation and partner negotiation at the time of writing; crucially, the choice does not affect the validity or comparability of the CASTOR evaluation, as the integration boundary exposed to the V2X Application Server and the CASTOR orchestration layer is identical in both cases.

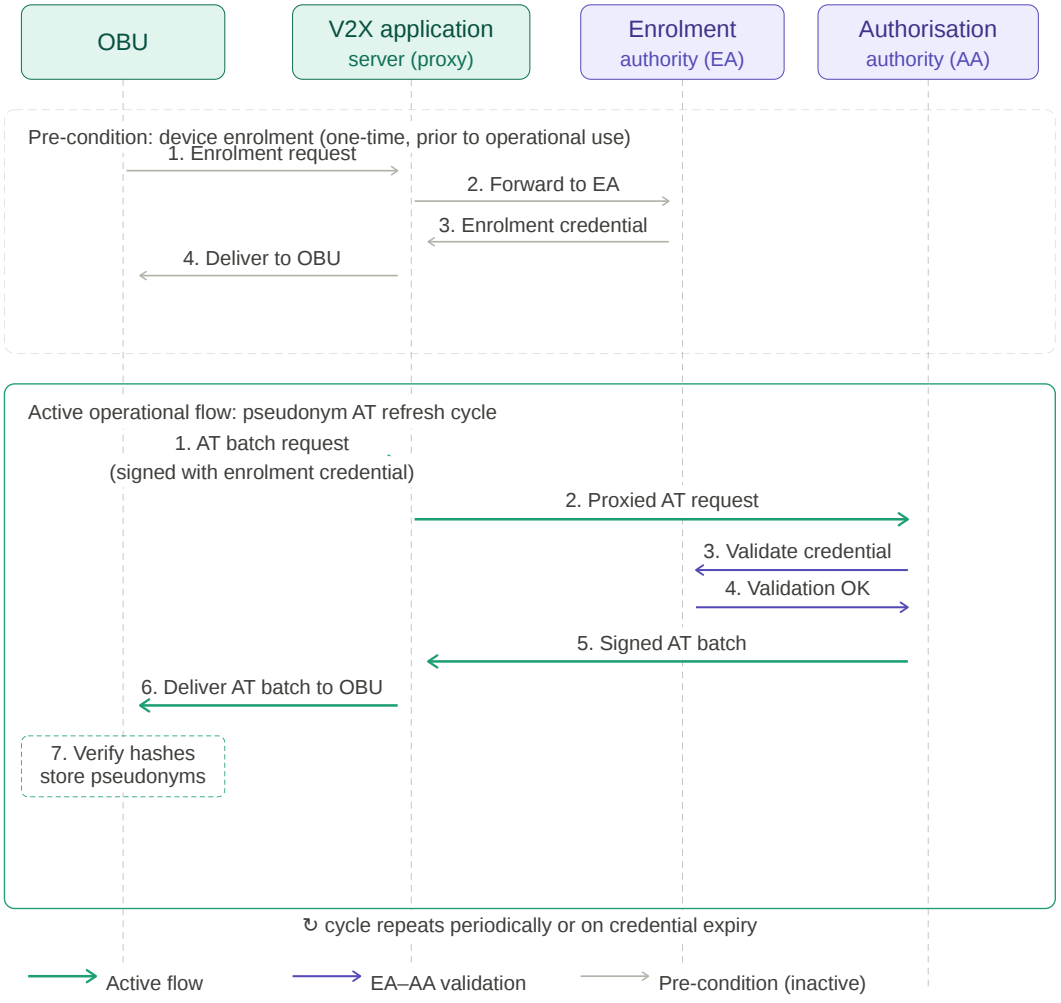

Figure 6.6: PKI credential provisioning flow (ETSI TS 102 941)

6.4.2 Components under development or represented as stubs

The as-is baseline configuration of the CMS testbed – comprising the OBU, V2X Application Server, PKI backend, and OTA server – is functionally complete and deployable in its current state. The existing components require no structural modification to support the CASTOR-enabled use case; rather, they need to be staged in a deployment configuration that exposes the correct integration points for CASTOR interaction. The primary development work remaining for the first release concerns the CASTOR-facing extension of the V2X Application Server: specifically, the reactive control logic that enables the V2X AS to respond to CASTOR runtime notifications – such as SSLA violation alerts or path degradation events – by triggering service-level actions such as session suspension or operator alerting. Until these extensions are implemented, the CASTOR integration points are represented as stubs within the V2X AS, allowing the as-is service workflows to be exercised and measured independently while the to-be control interfaces are developed and integrated incrementally.

6.4.3 Evolution towards CASTOR-enabled experimentation

The integration of CASTOR capabilities into the V2N ecosystem follows an iterative, progressive adoption model. In each integration phase, new CASTOR modules are interfaced with the V2X Application Server, which serves as the stable integration boundary between the use-case application layer and CASTOR's orchestration and trust management functions. Initially, the integration relies on pre-agreed SSLAs and utilises the Trust Awareness API to receive real-time notifications regarding path deviations or SSLA violations. In the second release of the integrated framework (see the overview in Chapter 2), CASTOR will explore dynamic service request capabilities, enabling the V2X Application Server to request new connectivity paths based on the domain's Path Profile Catalogue. Subsequent phases extend this to cross-domain trust exposure and dynamic fallback path activation, as defined in the to-be scenario.

UC2 Evaluation Plan			
User Story	**Description**	**1st Evaluation**	**2nd Evaluation**
UC2.US1a	Secure update delivery based on trusted network paths	Focus will be given on the time required for the establishment (and maintenance) of a trusted path compliant to the agreed SSLA. Essentially, impact on the throughput and FW upgrade latency that may be introduced considering the operation of the Global TAF for path-centric trust awareness.	During the second round of experiments, trust awareness will also be enhanced with trust opinions originate from the Local TAF deployed in ech routing element. All auxiliary crypto constructions, tracing mechanisms required for the verifiable monitoring and sharing of trustwrthiness evidence will be considered as part of the baseline operation.
UC2.US1b	Secure certificate renewal based on trusted network paths	Same as above but considering the extended set of trust properties required for the case of certificate renewal.	To also consider the scalability of CASTOR deployed trust sources (Attestation and FSM) when calculating trustworthiness evidence based on an extended set of raw traces extracted.
UC2.US2	Update adapts to degraded trust	Focus will be given on the time needed from the identification of an SSLA policy violation to the enforcement of a new path (over routing elements exhibiting the Required Trust Level). End-to-end experiments will be conducted for the routing path alternative establishment.	During the second round of experiments, this will be elevated to a cross-domain topology including also the sharing of (abstracted) trust-related information through CASTOR's Blockchain infrastructure as well as the DORE crypto protocol. Focus will be given on the additional overhead imposed by the CASTOR artifacts on the routing plane - especially on how these can impact the safety profile of the application layer.
UC2.US3	Secure Routing to PKI Server in a different domain	During the first experimentation period, focus will be given on path establishment in a cross-domain topology in the case where only one path is available.	To be elevated in topologies where multiple path exist across multiple domains.

Table 6.4: UC2 Evaluation plan

Chapter 7

Priority-based Trusted Messaging & Scalable Performance for CCAM Applications in the ORO testbed

The verification concept developed by TUIASI that will run on the Orange testbed provides the UC3 experimentation environment used to validate CASTOR support for safety-critical message exchange in intelligent transportation systems (ITS). The verification concept combines an OMNeT++-based simulation environment, a Geo Service Provider (GSP) processing core, and operator-facing dashboard services deployed on the Orange infrastructure described in Section 4.1.2.

The verification concept addresses the three UC3 user stories defined in Deliverable D2.1 [4]. In the Traffic Operator story, virtual vehicles generate Cooperative Awareness Messages (CAM) that are aggregated by the GSP into an environmental model consumed by a Traffic Operator dashboard. In the emergency response story, Decentralized Environmental Notification Messages (DENM) are classified and forwarded to a Local Emergency Responder (LER), with escalation to a Central Emergency Responder (CER) when severity thresholds are exceeded. In the Vulnerable Road User (VRU) story, detection nodes generate Collective Perception Messages (CPM) that are correlated with CAM traffic in order to produce low-latency warnings for vehicles at risk.

The deployment described in this chapter is provisioned to support both the simulation-side experimentation workflow and the operator-side dashboard workflow. This allows the TUIASI testbed to evolve from a baseline UC3 validation environment toward richer CASTOR-enabled experimentation without changing the fundamental separation between message generation, message processing, operator interaction, and observability.

7.1 Objectives, Scope and Planning

The TUIASI concept has been designed to support the experimentation scenarios defined for CASTOR Use Case 3. Its primary objective is to provide a controlled experimental environment capable of reproducing the generation, transport, processing, and visualisation of safety-critical V2X messages required for the evaluation of CASTOR trust-aware routing and service behaviour in intelligent transportation systems.

The concept supports the three UC3 workflows described in Deliverable D2.1 [4]. The first workflow focuses on traffic monitoring, where CAM traffic generated by virtual vehicles is aggregated by the GSP and presented to the Traffic Operator application. The second workflow focuses on emergency response, where DENM alerts are classified, routed to the LER, and escalated to the CER when the simulated incident severity requires cross-domain coordination. The third workflow focuses on VRU protection,

where CPM traffic generated by detection nodes is correlated with CAM data in order to identify collision risk and trigger timely warnings.

The key objectives of the TUIASI concept are therefore:

- To realise a representative UC3 deployment comprising virtual vehicles, detection nodes, a GSP processing core, operator-facing dashboards, and supporting data services.
- To provide a controlled and repeatable environment for exercising CAM, DENM, and CPM flows under measurable latency, delivery ratio, throughput, and processing constraints.
- To support both the simulation-side communication environment and the operator-side workflows needed to validate traffic monitoring, emergency response, and VRU safety scenarios.
- To provide a stable baseline configuration onto which CASTOR trust-aware routing, QoS-aware communication behaviour, and richer observability mechanisms can be progressively introduced.

The infrastructure supporting the TUIASI concept combines simulation hosts, dashboard services, dedicated data services, and observability components deployed on the Orange infrastructure. This separation keeps the operator path compact while preserving clear boundaries between workload types and ensuring that simulation outputs, event persistence, and monitoring data remain attributable throughout experimentation.

As the project progresses, the TUIASI concept will evolve from a baseline UC3 validation environment toward a richer CASTOR-enabled experimentation platform. The staged approach adopted in this chapter preserves the core message flows and operator interactions needed for consistent experimentation while leaving room for more detailed 5G QoS modelling, higher-scale scenarios, and trust-aware path selection.

7.2 High-level overview of UC applications

Figure 7.1 summarises the UC3 component architecture formed by a pipeline of data producers, a central processing hub, and role-specific consumers connected through explicit message flows. At the producer side, the verification concept uses virtual nodes to synthesise realistic V2X traffic under controlled experimental parameters. Virtual Vehicle Nodes generate periodic CAM messages describing kinematic state and event-triggered DENM messages representing hazards or accidents, while Detection Nodes generate CPM messages to emulate perception inputs such as VRU detections. All these streams converge at the Geo Service Provider, which acts as the functional core by ingesting and normalising messages, performing spatio-temporal correlation and deduplication, maintaining short-term contextual state, and generating higher-level outputs such as aggregated traffic and environment models, incident notifications, and collision-risk warnings.

The GSP outputs the data required by the operator applications and supporting backend services. For the Traffic Operator, the GSP publishes aggregated traffic and environment updates that are visualised on the Traffic Operator dashboard to support monitoring and traffic management decisions. For the emergency response workflow, the GSP forwards incident alerts to the LER dashboard and triggers escalation toward the CER dashboard when severity thresholds are exceeded. For the VRU workflow, the GSP issues vehicle-facing warnings derived from CAM-CPM correlation while also surfacing the situation to operator views for awareness. In this way, the dashboards close the loop shown in Figure 7.1 by transforming derived GSP outputs into actionable operational displays while preserving end-to-end measurability for UC3 KPIs.

Geo Service Provider and core simulation entities

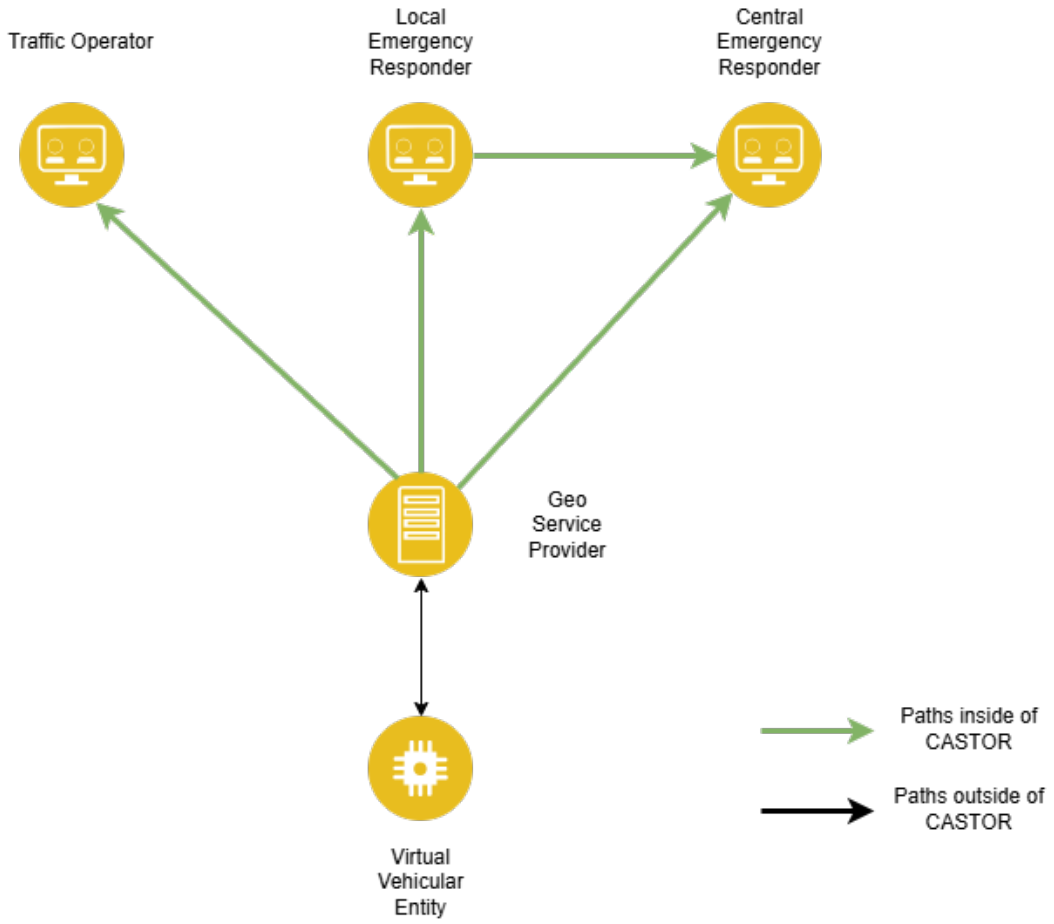

Figure 7.1: Component architecture for Use Case 3

The Geo Service Provider represents the central processing entity of the UC3 simulation architecture. It is modelled as a server-side software service deployed on a generic compute platform and integrated into the OMNeT++ simulation environment. Within the UC3 scenario, the GSP is responsible for ingesting, interpreting, and disseminating V2X-related information generated by simulated vehicles and infrastructure-based sensing components. It receives CAM-derived traffic information aggregated by infrastructure-side components together with DENM messages generated by vehicles and sensor streams provided by detection nodes. These inputs are normalised and correlated in space and time in order to construct a consistent representation of traffic conditions and environmental events, after which the GSP applies decision logic that determines how messages are processed and through which communication path they are forwarded.

The GSP supports the three UC3 user stories through distinct but related workflows. First, CAM-derived information is aggregated to produce the traffic and environmental representation used in the Traffic Operator scenario. Second, DENM messages describing hazards or accidents are analysed to classify events by severity and determine alert dissemination behaviour for local and central emergency responders. Third, the GSP supports VRU awareness by ingesting sensor streams from infrastructure-based detection nodes and generating low-latency alerts toward vehicles potentially exposed to the detected event.

The vehicular part of the system is represented by the *VehicleNode*, which models a connected vehicle equipped with an On-Board Unit (OBU). Vehicles periodically broadcast CAM messages for local awareness and infrastructure-side aggregation, generate DENM messages in response to simulated hazardous events, and receive alerts produced by the Geo Service Provider. Infrastructure sensing is represented by the *DetectionNode*, which abstracts roadside sensing systems such as cameras or radar units. These nodes generate sensor streams directed toward the GSP, simulate the detection of events such as the presence of vulnerable road users, and provide controllable traffic load at the GSP input when evaluating

higher throughput or tighter latency constraints.

Layered architecture of the Geo Service Provider

The internal architecture of the Geo Service Provider follows a layered structure that separates runtime behaviour, communication mechanisms, data processing logic, trust and quality-of-service policies, and application behaviour. This organisation maintains architectural modularity and facilitates the mapping of conceptual components to OMNeT++ modules and message flows.

- **Platform runtime layer.** The simulation is implemented using OMNeT++ with C++ modules executed on standard Linux-based development platforms. Within OMNeT++, this layer is represented through processing delays, queue behaviour, and overload effects within the *GeoServiceNode*, enabling the simulation to capture server-side processing latency and congestion under increased message load.
- **Communication layer.** Vehicles transmit DENM messages toward the GSP through a simulated V2X communication infrastructure representing either cellular connectivity or infrastructure-assisted communication. CAM-derived traffic information reaches the GSP through infrastructure-side aggregation components, while detection nodes provide additional sensor input through the same simulated communication environment. Parameterised channels capture delay, packet loss, bandwidth variation, and path switching events, providing the abstraction used for trusted and non-trusted communication alternatives.
- **Data processing layer.** CAM-derived information is interpreted and aggregated to construct a coherent view of traffic conditions and environmental context. DENM messages are analysed by a rule-based severity component that evaluates the type of event and its contextual indicators. The same layer performs deduplication, timestamp alignment, geo-filtering, and event correlation in order to prevent repeated or spatially irrelevant reports from influencing later decisions.
- **Trust and QoS layer.** Communication constraints influence message handling inside the GSP by classifying traffic according to communication requirements, prioritising safety-related messages over routine awareness traffic, and evaluating conditions such as latency, delivery ratio, throughput, and path availability. Together with path-selection logic, this layer represents the abstracted trust-aware routing behaviour defined by the CASTOR architecture.
- **Application layer.** The application layer translates lower-layer outputs into the behaviours required by the UC3 workflows. It produces aggregated traffic information for monitoring, alert behaviour for accident-related workflows, and VRU warnings derived from vehicle and detection-node context. In the OMNeT++ implementation, these behaviours are realised through dedicated message classes such as *TrafficModelMsg*, *AccidentAlertMsg*, and *VRUAlertMsg*, while the corresponding KPIs are collected through OMNeT++ signal recording configured in the *omnetpp.ini* simulation file.

7.3 Verification concept capabilities and technologies

The TUIASI verification concept provides the physical and virtual infrastructure used to realise the UC3 applications and supporting services described in the previous section. The deployment architecture is designed to preserve the essential interaction patterns between virtual vehicles, sensing nodes, GSP processing, operator dashboards, and observability services while remaining controlled, repeatable, and measurable for experimentation.

The infrastructure combines simulation hosts, operator-facing dashboard services, dedicated data services, and an observability plane. In its current form the testbed supports baseline UC3 validation on the

Orange infrastructure, while the same architectural split leaves room for richer trust-aware routing, more detailed 5G QoS emulation, and higher-scale scenario execution in later experimentation stages.

7.3.1 Overall Architecture

The TUIASI UC3 deployment is organised around four cooperating domains that separate simulation, operator-facing applications, persistent data management, and observability:

- A GSP simulation host that executes the OMNeT++-based experimentation environment together with the software toolchain required to model vehicle, sensing, and GSP behaviour.
- A dashboard application node that runs the single-tenant, edge-oriented dashboard stack used by the Traffic Operator, LER, and CER workflows in the UC3 pilot.
- A dedicated data node that provides the durable PostgreSQL event store and the edge-local Valkey service used for low-latency cache and channel-layer coordination.
- An observability node that centralises metrics, traces, dashboards, and alert views across both the simulation-side and dashboard-side services.

This decomposition keeps the time-sensitive operator path short while preserving clear boundaries between workload types. The dashboard stack remains optimised for edge deployment and pilot operation by colocating the browser-facing services, API, WebSocket gateway, and outbox dispatcher on a compact application node, while the GSP simulation side remains a software-based experimentation platform built around OMNeT++. In this way, the TUIASI concept can evolve the simulated communication and processing environment independently from the operator dashboard runtime, without obscuring the infrastructure resources required by each part of the UC3 workflow.

Before detailing the runtime topology and services, it is important to state the principles that shape the dashboard deployment in the TUIASI verification concept. The UC3 pilot favours a compact edge-oriented architecture over a generic multi-tenant cloud setup because the supported workflows are time-sensitive and must meet tighter latency budgets. Three constraints shape the deployment:

- An edge-local in-memory cache is maintained for latency-sensitive lookups such as trust posture and tier state.
- The hot path for operator actions avoids dependence on remote streaming components, while the streaming and audit planes remain available for durability and compliance.
- The system follows a fail-loud philosophy: when key dependencies degrade, the dashboards must surface that degraded state explicitly instead of silently masking missing or delayed information.

The infrastructure is further decomposed into frontend delivery, backend API and WebSocket handling, event storage, cache and channel management, outbox dispatching, and an observability plane. This separation allows each layer to be developed, tested, deployed, and troubleshot independently. If the cache layer is impaired, the event store remains authoritative; if WebSocket delivery is delayed, the audit trail is still preserved.

The dashboard runtime is organised around a compact set of services that keep the operator path short while preserving a durable audit history and real-time event dissemination. Figures 7.2 and 7.3 summarise the deployment view used for the UC3 pilot. At the centre of the deployed runtime sits the Django backend, which serves both traditional HTTP requests and persistent WebSocket connections through a single ASGI endpoint. This single-port approach simplifies ingress routing because the reverse proxy

or load balancer does not need separate routing logic for REST traffic and WebSocket traffic. Operator browsers connect over HTTPS to the ingress layer (i.e., the component that sits at the edge of web application), which then routes page delivery to the SvelteKit frontend and API or WebSocket traffic to the backend. The backend communicates with PostgreSQL as the durable event store and with Valkey for both caching and real-time channel management, while the external trust exposure service provides trust posture information over HTTPS.

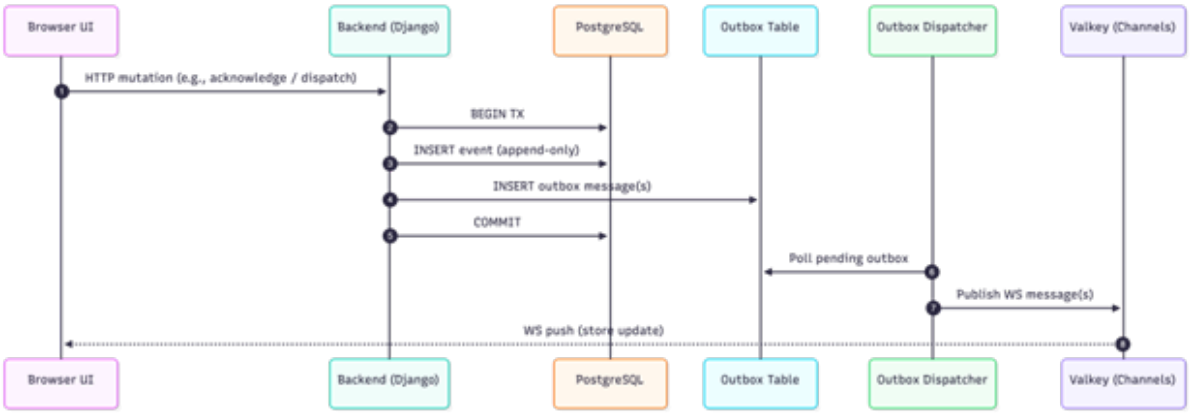

Figure 7.2: High-level container topology of the CASTOR Dashboard platform

Valkey has a dual role in this topology. It acts as the Django Channels channel layer, allowing the backend to broadcast updates to groups of connected clients, and it also provides a short-TTL cache for hot-path lookups that must complete with minimal latency.

One of the key consistency patterns used in the dashboard runtime is the transactional outbox. Every operator-visible event and every WebSocket push is tied back to the same durable write path so that real-time updates do not diverge from the audit record.

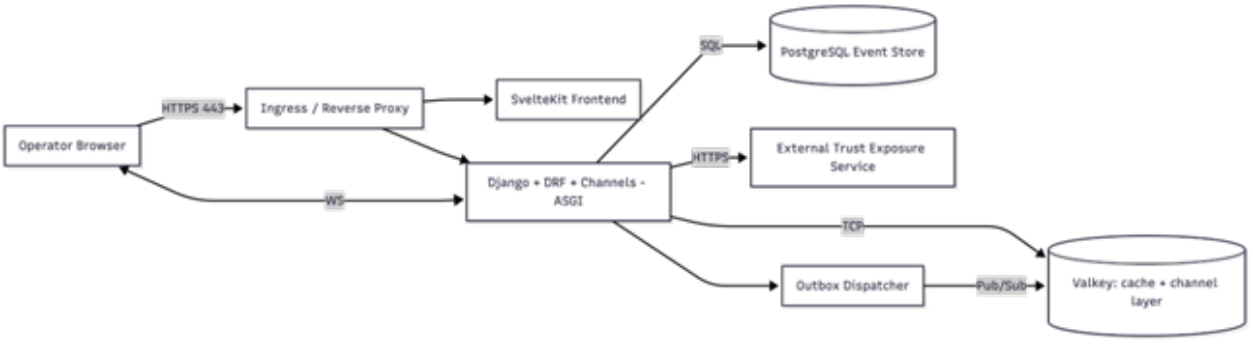

Figure 7.3: Transactional outbox from operator action to WebSocket delivery

When an operator performs a mutation such as acknowledging an alert or dispatching emergency services, the backend opens a database transaction and writes both the append-only event and the corresponding outbox message within that same transaction. This guarantees that the system cannot record an event without a matching notification, nor emit a notification for an event that was not durably committed. A separate Outbox Dispatcher process continuously polls the outbox table for pending messages and publishes them to Valkey, which then fans them out to the relevant WebSocket clients. This separation is operationally important because transient publish failures in the WebSocket delivery path do not compromise the core transaction processing or the integrity of the audit record.

Table 7.1 summarises the principal compute resources available across the physical nodes supporting the TUIASI verification concept. These resources support the execution of the baseline UC3 topology

while providing sufficient capacity for richer communication modelling, more capable operator services, and extended observability in later experimentation stages.

Node/ Host	Role in testbed	Key software/ services	CPU	RAM	Storage	Technologies
GSP Simulation Host	Runs the OMNeT++-based UC3 simulation environment and Geo Service Provider processing workflow.	Ubuntu 22.04 LTS, OMNeT++ 6.2.0, INET 4.5.4, Simu5G 1.4.0, gcc/g++, CMake/Make, Python 3.10+, SUMO 1.20.0, Veins 5.3.1.	8 vCPU	16 GB	100 GB SSD	OMNeT++, INET, Simu5G, SUMO, Veins, Linux build toolchain.
Dashboard App Node	Serves the frontend, backend API, WebSocket gateway, and outbox dispatcher for UC3 operator workflows.	Ubuntu Server 24.04 LTS, Docker Engine with Compose, Python 3.13+, Django, DRF, Channels, Daphne, Node.js 20/22+, optional reverse proxy.	4 vCPU	8 GB	80 GB SSD	SvelteKit, Django ASGI stack, Docker Compose, WebSocket delivery.
Data Node	Provides the durable event store, outbox persistence, and low-latency cache and channel-layer support.	Ubuntu Server 24.04 LTS, PostgreSQL 17+, Valkey 8.1+, backup tooling including pg_dump and storage snapshots.	4 vCPU	16 GB	200 GB SSD	PostgreSQL, Valkey, backup and recovery services.
Observability Node	Collects telemetry and exposes metrics, traces, dashboards, and alert views for the full UC3 deployment.	Ubuntu Server 24.04 LTS, OpenTelemetry Collector, Prometheus, Grafana, optional Alertmanager.	2–4 vCPU	8 GB	100 GB SSD	OpenTelemetry, Prometheus, Grafana, alert visualisation.

Table 7.1: Resources of TUIASI in ORO testbed

Operationally, browser access enters the dashboard stack over HTTPS on port 443, while PostgreSQL on port 5432 and Valkey on port 6379 should remain internal-only wherever possible. All nodes depend on reliable NTP synchronisation in order to preserve timestamp consistency between simulation outputs, event persistence, and observability data. The observability node is intended to cover both the OMNeT++/GSP side and the dashboard runtime, while a separate audit-pack or artifact-storage node may be added later if long-term experiment retention or signed evidence packaging becomes necessary.

7.3.2 Requirements and Technologies

The dashboard infrastructure is designed to run across multiple environments, each serving a specific purpose in the software lifecycle while keeping the core topology consistent from development to the live UC3 pilot. Table 7.2 summarises the deployment modes used to support fast iteration, automated validation, integration testing, and the single-tenant pilot environment.

Environment	Purpose	Deployment method	Key characteristics
Local development	Fast iteration and feature development	Docker Compose + native dev servers	Postgres and Valkey run in containers; backend and frontend run natively on the developer's machine.
CI (PR validation)	Automated quality checks on every pull request	GitHub Actions runners	Builds and tests the codebase; may spin up ephemeral containers for integration tests.
Staging	Integration testing in a production-like environment	Kubernetes (dev overlay)	Mirrors the production topology with lower resource allocations for cost efficiency.
UC3 Pilot / Production	Live operational environment for real users	Kubernetes (prod overlay)	Single-tenant instance with edge-local dependencies for optimal latency.

Table 7.2: Environment deployment modes

The local development environment is designed to keep developer iteration fast while still reflecting the production topology. Docker Compose manages the shared infrastructure dependencies, specifically PostgreSQL for the event store and Valkey for caching and channel management. The application services themselves are typically run natively on the developer workstation to benefit from hot reload, while optional containerised variants can still be used when parity with CI or staging is needed. The local workflow uses Docker Compose for shared infrastructure dependencies and optional containerised application services:

- `postgres`: event store container
- `valkey`: cache and channel layer container
- optional containers for backend, frontend, and outbox services when testing CI or staging parity

Typical development workflow:

1. Bring up infrastructure: `docker compose up -d postgres valkey`
2. Run the backend locally: `cd backend && python manage.py runserver`
3. Run the frontend locally: `cd frontend && npm run dev`
4. Run the outbox dispatcher when testing WebSocket delivery: `cd backend && python manage.py run_outbox`

The CASTOR dashboard stack is composed of a small number of application and data services, each with a clearly defined role. Together they provide browser delivery, synchronous and asynchronous operator workflows, durable event storage, low-latency distribution of live updates, and operational visibility.

Service	Runtime	Responsibilities	Scaling (Pilot)
Frontend	Node/SvelteKit	UI delivery, client-side routing, state management, WebSocket client	1 replica
Backend	Django + DRF + Channels + Daphne (ASGI)	REST APIs, WebSocket consumers, RBAC enforcement, domain services	1 replica
Outbox Dispatcher	Python worker (Django management command)	Poll outbox table, publish messages to Valkey for WebSocket delivery	1 replica

Table 7.3: Service runtimes and responsibilities

Frontend. The frontend is built with SvelteKit and is responsible for everything the operator sees and interacts with in the browser. It performs page delivery, handles client-side routing, manages local state, and maintains the WebSocket connection used to receive real-time updates. A single replica is sufficient for the UC3 pilot.

Backend. The backend is implemented with Django, Django REST Framework, and Django Channels running on the Daphne ASGI server. It exposes the REST API for operator actions, manages WebSocket consumers for live communication, enforces role-based access control, and coordinates the domain services that manage incidents, trust posture, and alert workflows. A single replica is sufficient for the pilot deployment.

Outbox Dispatcher. The outbox dispatcher is a Python worker implemented as a Django management command. Its sole responsibility is to poll the outbox table in PostgreSQL for pending messages and publish them to Valkey for WebSocket delivery. Running it as a distinct process makes event delivery resilient to transient failures without risking the integrity of the event store.

Service	Runtime	Responsibilities	Scaling (Pilot)
PostgreSQL	Managed or in-cluster	Append-only event store, outbox, projections / read models	1 instance (HA optional)
Valkey	In-cluster/edge	Cache + Django Channels channel layer for WS fanout	1 instance (HA optional)
Observability stack	OTel collector + Prometheus + Grafana	Traces, metrics, dashboards, alerts	Shared

Table 7.4: Platform services

PostgreSQL. PostgreSQL serves as the durable system of record for the platform. It stores the append-only incident and event tables that form the immutable audit trail, the outbox message table used for reliable WebSocket delivery, and the read-model and projection tables that power dashboard queries. In operational deployments, these read models must be indexed appropriately to meet the query latency expected by operators, and pilot or production deployments should include regular backups and point-in-time recovery.

Valkey. Valkey is the in-memory data service used by the dashboard runtime. First, it acts as the Django Channels channel layer and supports WebSocket group fanout to connected browsers. Second, it provides a cache-aside layer for hot-path lookups such as trust posture and tier state with short time-to-live values. Valkey is treated as non-authoritative: if it becomes unavailable, cache lookups and WebSocket fanout may degrade, but PostgreSQL remains the source of truth and state can be rebuilt once Valkey recovers. For the UC3 pilot, Valkey must remain edge-local to satisfy latency targets.

Observability stack. The observability stack consists of an OpenTelemetry collector, Prometheus, and Grafana. These components are shared across the deployment and provide the metrics, traces, dashboards, and alert visualisations needed to understand system behaviour, diagnose faults, and track service-level objectives during the UC3 pilot.

The simulation-side environment relies on OMNeT++ 6.2.0 together with INET 4.5.4, Simu5G 1.4.0, SUMO 1.20.0, Veins 5.3.1, and the standard Linux build toolchain. Within this environment, the radio access network conditions used in the TUIASI verification concept are represented through simulation models rather than a physical radio infrastructure. OMNeT++ networking modules derived from the INET framework provide configurable communication channels capable of representing latency, packet loss, bandwidth constraints, and link variability, allowing the concept to reproduce the radio conditions that influence message transmission between vehicles, infrastructure nodes, and backend services.

The simulated RAN environment supports the transmission of CAM, DENM, and CPM traffic. By controlling channel parameters and traffic generation patterns, the verification concept enables the evaluation of communication performance indicators including end-to-end latency, packet delivery ratio, and throughput under different network conditions. Edge computing functionality is primarily represented by the Geo Service Provider, which performs near-real-time processing of V2X information generated by vehicles and infrastructure sensing components. In the current simulation environment, the GSP is implemented as a server-side software module integrated within OMNeT++ and is responsible for deriving higher-level traffic context models, hazard notifications, and collision-risk alerts from incoming messages.

Simulation-based virtualisation is used throughout the TUIASI verification concept to reproduce large-scale vehicular communication scenarios in a controlled experimental environment. Instead of deploying physical vehicles or sensing infrastructure, the system relies on virtual entities implemented within the OMNeT++ simulation framework. Multiple types of nodes can be instantiated and configured dynamically, including virtual vehicles, infrastructure sensing nodes, and backend processing components. Each virtual node executes its own communication logic and interacts with other nodes through simulated communication channels, allowing the platform to reproduce complex V2X communication scenarios in a repeatable way.

7.3.3 Realization of each UC application service

The functional roles introduced in Figure 7.1 are realised through a combination of simulation modules, backend services, dashboards, and monitoring tools. Each UC3 user story is therefore mapped onto a concrete set of concept services and message flows.

- **Traffic Operator workflow.** Virtual vehicles generate CAM traffic inside the OMNeT++ environment. Infrastructure-side aggregation components forward the CAM-derived information to the GSP, which produces an aggregated traffic and environmental model. That model is exposed to the backend and visualised through the Traffic Operator dashboard to support situational awareness and traffic management decisions.
- **Emergency response workflow.** Vehicles generate DENM messages when simulated hazardous events occur. The GSP analyses these messages through its severity-classification logic, produces incident alerts, and forwards them to the LER dashboard. When severity thresholds indicate that broader coordination is required, the same workflow supports escalation toward the CER dashboard over a cross-domain path.
- **VRU workflow.** Detection nodes emulate roadside sensing systems and provide CPM traffic to the GSP. The GSP correlates those CPM inputs with CAM data received from virtual vehicles in order to identify predicted trajectory intersections and generate low-latency warnings for vehicles potentially exposed to a VRU-related hazard.
- **Operator dashboards and backend services.** The frontend, backend, and outbox dispatcher collectively realise the operator-facing part of UC3. The frontend provides the browser-based user interface, the backend enforces role-based access control and domain services, and the outbox dispatcher ensures that operator-visible events are delivered consistently over WebSocket channels without diverging from the durable event log stored in PostgreSQL.
- **Persistence and observability services.** PostgreSQL provides the authoritative audit trail and projection storage, while Valkey supplies low-latency cache and WebSocket fanout support. The observability stack collects traces, metrics, dashboards, and alert views across the simulation-side and dashboard-side services so that UC3 behaviour can be analysed during and after each experiment.

Figure 7.4 shows the supervisor-facing incident feed used in the UC3 emergency-response workflow, illustrating how operator dashboards surface live incidents, trust labels, and degraded realtime-channel state during pilot operation.

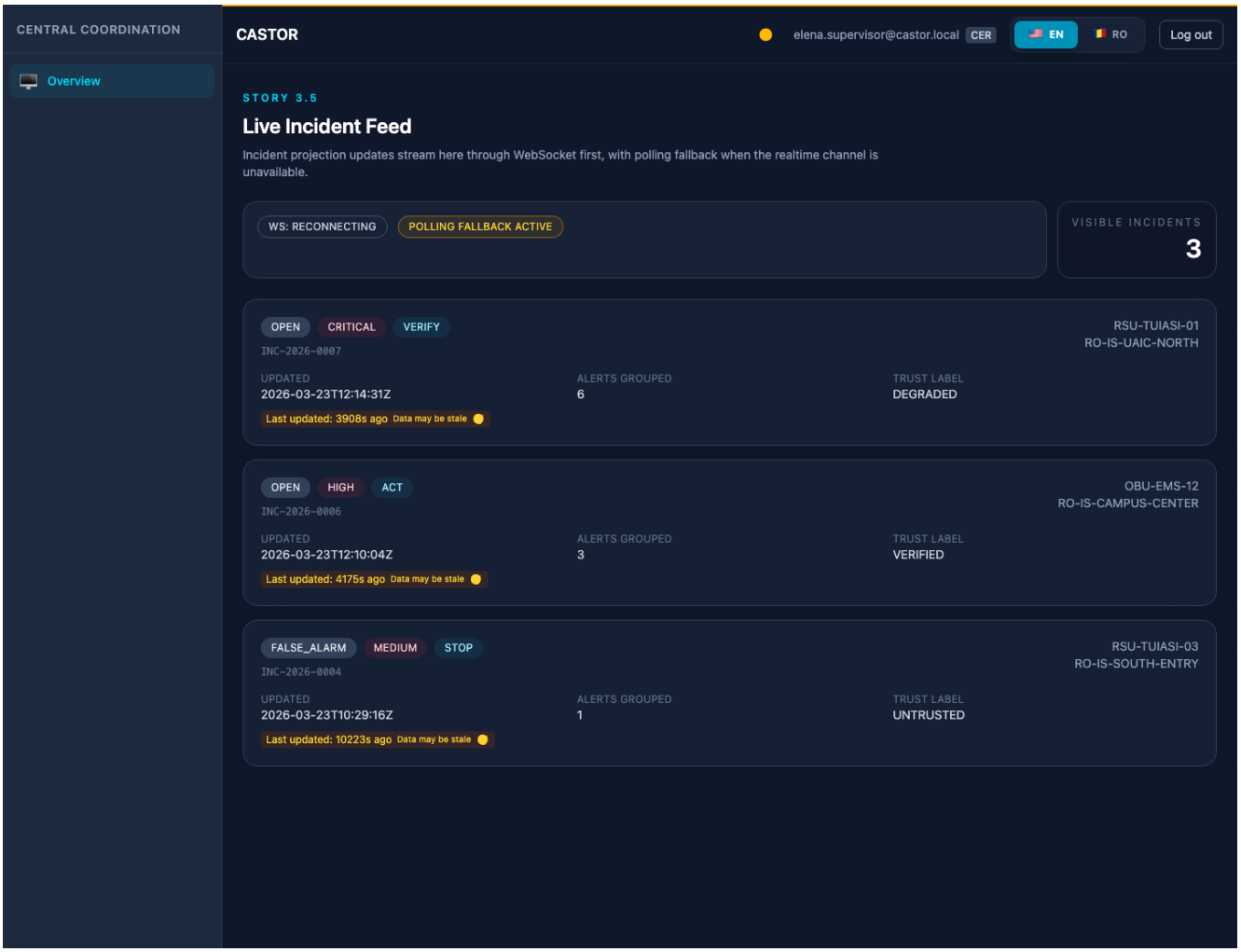

Figure 7.4: Supervisor incident feed view for the UC3 emergency-response workflow

Monitoring within the TUIASI concept operates at two complementary levels. At the simulation level, OMNeT++ signal recording mechanisms capture runtime performance indicators associated with individual modules. These include end-to-end latency, packet delivery ratio, message throughput, and processing delays at the Geo Service Provider, with results written to scalar and vector output files for subsequent analysis. At the application level, the dashboard infrastructure allows operators and researchers to observe traffic situations, incident alerts, and system responses during experiments, providing a broader view of UC3 workflow behaviour across the different user stories.

Service-level interaction within the TUIASI verification concept is organised around the outputs produced by the Geo Service Provider, which exposes processed information derived from incoming CAM, DENM, and CPM messages in the form of aggregated traffic context, hazard notifications, and collision-risk alerts. These outputs are consumed by application-level components supporting the UC3 workflows, allowing operators and researchers to monitor traffic situations, observe incident propagation, and evaluate system responses across the different user stories. This interaction layer enables the validation of application workflows and the analysis of system behaviour under different simulated traffic and network conditions.

7.4 Use-Case Software Readiness and Staging

The TUIASI verification concept is intended to support the staged realisation of UC3, beginning from a baseline simulation and dashboard environment and progressing toward richer CASTOR-enabled experimentation. At the time of this deliverable, the primary objective is to establish a credible baseline for message generation, processing, operator visualisation, and KPI collection against which later trust-aware and QoS-aware enhancements can be evaluated.

The current software platform therefore combines operational UC3 components with elements that remain under active development. The following subsections distinguish between the baseline capabilities that are already available, the components that are still being extended or represented in simplified form, and the evolution path toward later CASTOR-enabled experimentation.

7.4.1 Baseline software capabilities

A number of core use-case components are currently operational within the TUIASI verification concept and provide the baseline functionality required for experimentation.

- **OMNeT++-based UC3 simulation environment.** The testbed runs an operational simulation environment based on OMNeT++, INET, and the associated Linux toolchain. This environment provides the execution context for virtual vehicles, detection nodes, aggregation components, and the Geo Service Provider processing logic.
- **Vehicle and sensing entities.** The *VehicleNode* and *DetectionNode* abstractions provide the baseline producers for CAM, DENM, and CPM traffic. Together they enable the controlled generation of safety-critical V2X message flows needed to exercise the three UC3 user stories.
- **Geo Service Provider processing workflow.** The GSP currently performs the core functions required for UC3 validation, including message ingestion, normalisation, deduplication, severity classification, geo-filtering, event correlation, and application-specific output generation for traffic monitoring, emergency response, and VRU safety workflows.
- **Operator-facing dashboard stack.** The Traffic Operator, LER, and CER applications are supported by a single-tenant dashboard runtime comprising the SvelteKit frontend, Django backend, and outbox dispatcher. This stack provides browser delivery, REST and WebSocket handling, real-time event dissemination, and durable event persistence for operator workflows.
- **Data persistence and cache services.** PostgreSQL and Valkey are available as the baseline data services for the UC3 pilot. PostgreSQL stores the append-only event log, outbox entries, and projection tables, while Valkey provides the channel layer and low-latency cache required for responsive operator interaction.
- **Observability and KPI collection.** OMNeT++ signal recording together with the OpenTelemetry, Prometheus, and Grafana stack provides the initial monitoring baseline. This allows simulation metrics, service behaviour, and dashboard-side telemetry to be observed during experiments and reviewed afterwards.

7.4.2 Components under development or represented as stubs

While the baseline UC3 workflow is available, several parts of the TUIASI software platform continue to evolve as the experimentation environment matures.

One area of ongoing development concerns the fidelity of the communication environment. The current simulation relies on configurable channel abstractions for latency, delivery ratio, and throughput, but tighter integration with Simu5G is planned in order to represent 5G NR QoS behaviour, dynamic latency variations, and congestion scenarios in greater detail. In the same way, the current GSP processing logic already captures the main UC3 workflows, but richer event-correlation and traffic-context modelling remain under active development for more complex urban scenarios with multiple sensing sources and simultaneous incidents.

The deployment model also contains elements that are intentionally provisional. The simulation runtime is currently centred on a software-based OMNeT++ environment rather than a containerised execution model, and the pilot dashboard deployment prioritises a compact edge-oriented stack over a broader multi-site packaging approach. Likewise, the observability node is intended to cover both the OMNeT++/GSP side and the dashboard runtime, while a separate audit-pack or artifact-storage capability remains optional for later experiment retention and signed evidence packaging.

7.4.3 Evolution towards CASTOR-enabled experimentation

The staged development approach adopted for the TUIASI verification concept allows the baseline UC3 environment to be established independently of the most advanced CASTOR-specific mechanisms. This separation ensures that baseline message flows, operator interactions, and system behaviour can first be characterised in a controlled setting before more complex trust-aware behaviour is introduced.

In later project phases, the TUIASI concept will evolve toward richer CASTOR-enabled experimentation by tightening the relationship between communication conditions, path selection, and application behaviour. More detailed 5G QoS modelling, more explicit trusted and non-trusted communication alternatives, and higher-scale simulation scenarios will allow the project to study how degraded or shifting communication conditions influence UC3 workflows and service outcomes.

The same evolution path applies to the operator-facing side of the platform. The dashboard runtime, data services, and observability plane are designed so that CASTOR trust exposure, policy-driven routing decisions, and larger validation campaigns can be introduced incrementally on top of the existing baseline deployment. This will allow experimentation to explore how trust-aware routing, policy enforcement, and service-level visibility affect traffic monitoring, emergency response, and VRU protection across progressively more demanding UC3 scenarios.

UC3 Evaluation Plan			
User Story	**Description**	**1st Evaluation**	**2nd Evaluation**
UC3.US1	Traffic operator	Using the TUIASI concept and Orange infrastructure, the availability of the connection between the GEO Service Provider and the Traffic Operator (TO) will be verified. In addition, the integrity of the traffic model packets received by the TO will be measured. Finally, the latency of the link between the GSP and the TO will be measured.	The CASTOR framework will be evaluated to determine its impact on the availability, integrity, and latency of the TO application.

UC3.US2	Emergency Responder	Using the TUIASI concept and Orange infrastructure, the availability of the connection between the GEO Service Provider and the Local Emergency Responder (LER) will be verified. In addition, the integrity of the accident notifications received by the LER will be measured. Finally, the latency of the link between the GSP and the LER will be measured.	The CASTOR framework will be evaluated to determine its impact on availability, integrity, and latency. Moreover, the cross-domain scenario required for severe accident notifications which must reach the Central Emergency Responder will also be evaluated.
UC3.US3	Vehicle Driver	Using the TUIASI concept and Orange infrastructure, the availability, the integrity and the latency for the connection between detection nodes, GSP and vehicles will be measured.	The CASTOR framework will be evaluated to determine its impact on the availability, integrity, and latency of the vehicle driver application.

Table 7.5: UC3 Evaluation plan

Chapter 8

Future-Proofing Next-Generation Unmanned Aerial Vehicles Communications towards Critical Infrastructure Sustainability in the K3Y Testbed

The K3Y testbed will be used to establish a baseline operational network environment for the K3Y use case scenarios validation. The testbed, as presented in this deliverable, focuses on the applications that realize the end-to-end UAV operational lifecycle and does not yet include CASTOR trust-aware mechanisms. Its integration with the CASTOR-enabled forwarding plane provisioned in the ORO testbed will be documented and evaluated in D6.2. The current setup focuses on baseline testing with conventional policies regarding routing and connectivity, and is aligned with the K3Y use case scenarios defined in D2.1 [4], specifically with Scenario 1: CASTOR in the Data Network, Scenario 2: External Risk Indices as input to CASTOR in Data Network, as well as the nice-to-have Scenario 3: CASTOR in the Shared Back-haul Infrastructure. The respective deployment diagrams are depicted in Figure 8.1 and in Figure 8.2. These deployment topologies aim to replicate telemetry, mission control and mission inspection of data flows, within the Data Network or a shared backhaul network infrastructure and will be the core where CASTOR trust-aware mechanisms will be introduced. For each scenario, the vRouter topology is placed between the respective use case components through VPN connections that force the traffic of the use case components to pass through the vRouter topology, that is hosted centrally at ORO.

A summary of data flows per scenario is provided bellow:

- For the first scenario, the data flows are focused on the interaction of the Swarm Manager and the 5G core in the Data network since in this scenario everything behind the 5G core will be considered a black box. The Swarm manager will issue MAVlink mission waypoints, control commands, and configuration updates for the UAVs, which will traverse the Data Network towards the 5G core entry point, where they will be forwarded to the UAVs without further visibility. Telemetry data from the UAVs, and mission inspection data will also exit the UPF towards the Swarm Manager. This bi-directional data exchange forwarded through the Data Network will be the focus of scenario 1. It should be noted that this scenario constitutes the main scenario that focuses on the performance evaluation (e.g., latency incurred, SSLA monitoring, communication restoration strategies) of the CASTOR integrated framework in the UAV Communication workflows presented in this UC.
- In the second scenario, the same data flows will be applied as in first scenario, but an MNO will introduce a risk index for its administrate domain which will be taken into account in the Data Network. This risk index will be included in the 5G core prior to traffic forwarding and will influence the handling of the bi-directional traffic flow in the Data Network. This can be further expanded to

account for additional risk information from the overall operational risk posture of the ecosystem, such as external risk indices from other network Autonomous Systems or source and target domains, which may in turn impact the risk and trust analysis conducted with the CASTOR integrated framework.

- The third scenario aims to evaluate CASTOR in a distributed and multi-tenant 5G deployment, where there are multiple Intermediary UPF (I-UPF) closer to the edge, and a central Anchor UPF (A-UPF) on the 5G core. Hence, the data flows of this scenario focus on the GPRS Tunneling Protocol for User Plane (GTP-U) traffic over the N9 interface, both when forwarding UAV telemetry and mission inspection data from the I-UPF to the A-UPF, and when UAVs receive trajectories and mission updates that are forwarded from the A-UPF to the I-UPF. This bi-directional flow in the shared backhaul infrastructure will be the primary point of evaluation within this scenario. While CASTOR's in-router trust enablers (i.e., the TNDE) and the novel cryptographic protocols presented in D3.2 [7] are evaluated across all scenarios, this specific multi-tenant environment provides a unique opportunity; it allows us to assess the forwarding plane when multiple TNDIs from different operators are provisioned within the same shared infrastructure element.

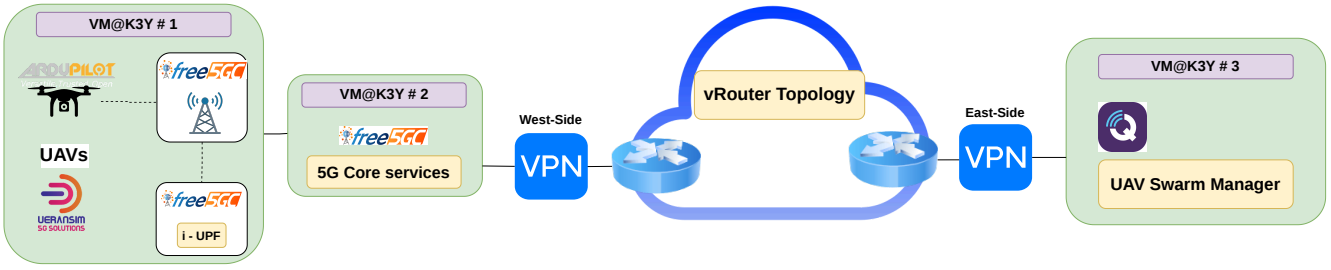

Figure 8.1: High-level deployment view of Scenarios 1 and 2

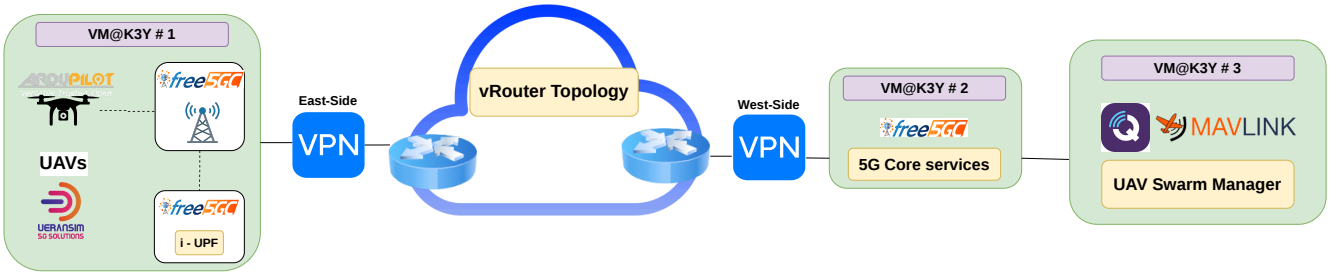

Figure 8.2: High-level deployment view of Scenario 3

8.1 Objectives, Scope and Planning

The K3Y testbed will support the experiment scenarios that are defined in D2.1 [4], with an environment that can reproduce all data exchange flows that are needed to evaluate CASTOR trust-aware routing and orchestration mechanisms, as well as support the respected deployment of the necessary components to support the scenarios.

The objectives of the testbed for this deliverable are:

- To realise a representative virtualised deployment supporting UAV swarm operations over a 5G network.

- Establish baseline communication flows between the UAV and 5G-related assets, under nominal conditions.
- To configure a system which will enable future evaluation of trusted path routing.

The K3Y testbed will focus on establishing normal operational status and will support end-to-end exchange of data such as telemetry, mission inspection data, waypoint trajectories for control and positioning, between UAV instances and radio access components, a 5G network, and a swarm manager that will be responsible for coordinating a UAV inspection mission. Supporting a modular architecture, the testbed supports enhancing components or replacing them with other components of similar functionality without changing in the underling infrastructure and can make use of any virtual routing infrastructure to connect the components.

The testbed will evolve along with the project. The baseline operational state that is defined in this deliverable will be extended with the integration of the CASTOR enabled forwarding plane to make use of trust-aware routing and orchestration components and explore more complex scenarios. This integration will mark the overall testbed for the evaluation UC scenarios to be documented in D6.2, whereas the evaluation of more complex scenarios in the forwarding plane will be explored in the second round of evaluations.

8.2 High-level overview of UC applications

At a functional level, the testbed involves simulated UAV assets, emulated RAN and 5G core network functions, ground control functions, and a mission-level swarm management responsible for coordinating UAV operations. These components are distributed across multiple virtual machines, and a vRouter topology at ORO will be used to forward and route traffic between the access, core, and application domains. The above will be included in a deployable architecture that will maintain the flow of data, while also will be isolated, and easy to configure for further experimentation.

From a workflow perspective, the use case scenarios will be based on a closed control loop with the UAVs on one end and the Swarm Manager on the other end. The mission will start with UAVs connecting to the Swarm Manager over the 5G network. The Swarm Manager is responsible for coordinating the mission and will transmit mission commands that include trajectory data such as waypoints to the UAVs. UAVs will send back to the Swarm Manager mission inspection data such as video streams as well as telemetry data such as position, attitude, system status to provide real time tracking. This communication between UAVs and Swarm Manager will be using MAVlink over dedicated PDU sessions provided by the 5G core. In parallel, QGroundControl will be connected to the Swarm Manager in order to monitor the mission and adjust additional parameter if needed.

8.3 Verification concept capabilities and technologies

The testbed core components will be hosted on K3Y premises on a Kernel Virtual Machine (KVM)-based virtualized environment, while the routing infrastructure is hosted on ORO's premises. The elements of the architecture are the following:

- A router topology that will be used for data forwarding between the core components.
- Software-based UAVs, gNB, and intermediate I-UPF generating telemetry and control traffic and forwarding user-plane data toward the 5G core.

- 5G Core control-plane functions and anchor UPF (A-UPF), which anchors UAV sessions and relays traffic between access and mission-management layers.
- The Swarm Manager logic issuing mission commands and receiving UAV feedback over the 5G core network.

Table 8.1 summarises the compute resources needed for the hosts that will support the use case components and the execution of the baseline routing topology.

Node/Host	Role in testbed	Key software/ services	CPU	RAM	Storage	Technologies
VM1 Edge network infrastructure	UAVs host and radio access	• Simulated UAVs • Emulated RAN component • Surveillance feed generation	4 cores	8GB	50GB	• UERANSIM • Free5gc • Ardupilot • Docker
VM2 5G core	provide the control and user plane of the mobile network and enable connectivity between UAVs, edge nodes, and service hosts	• Virtualized 5G core functions	4 cores	8GB	30GB	• Free5gc • Docker
VM3 Swarm manager	UAV mission inspection coordination	• UAV swarm management functions • UAV coordination service	4 cores	4GB	50GB	• MAVProxy • QGroundControl • Docker

Table 8.1: Testbed Node and Host Specifications

8.3.1 Overall Architecture

The K3Y testbed will be deployed in a K3Y private lab environment. It will consist of virtual machines that will act as different compute hosts where the UAV and 5G related services of the use cases will be deployed to, including simulated UAVs, 5G core network functions, and a Swarm Manager for mission control, while allowing any virtual router topology to be interconnected with the VMs via dedicated site-to-site VPN tunnels. This will provide a flexible environment to develop and test the use case components and the end-to-end communication flow.

The virtualized network topology that will be used to forward traffic between the VMs will be provided by ORO (see Chapter 4) and will be comprised of a set of Cisco XRd routers, that can also support the forwarding plane on top of which the CASTOR integrated framework will be deployed. Interconnectivity between the VM hosts should be made through the remote network topology, and will be established through several VPN sessions, forcing the VMs to forward network traffic destined to the rest VMs through the VPN tunnels, therefore passing through the vRouter topology.

8.3.2 Requirements and Technologies

The K3Y testbed will be comprised of a set of connected virtual machines that will host the needed components. More specifically:

- A virtual machine that will host the required software components for the Swarm Manager, such as a ground control station.
- A virtual machine that will host simulated UAVs, simulated RAN components, and a mission inspection data generator.
- A virtual machine that will host a virtualized 5G core implementation.

Hosting the components in different virtual machines is crucial to help preserve architectural separation between network infrastructure and use case elements. The components that will be used will be containerized, enabling them to be instantiated and reconfigured easily while running on K3Y lab hosts.

To generate telemetry and mission inspection data related to the UAVs, a simulated environment will be used that can generate this kind of traffic which will be used by the Swarm Manager to coordinate the UAV swarm for the execution of the mission. The combination of the above VMs and their containerized components, that can be connected using a virtualized router topology will form the platform that will comprise the necessary environments for K3Y use cases.

Access to the vRouter topology will be provided through a set of VPN connections. Thus, for the implementation of each scenario, two VPN sessions should be established from the K3Y lab, terminating at the east-side and west-side routers of the ORO's vRouter topology respectively. The appropriate routing advertisements should be pushed by the VPN servers, forcing the necessary traffic to pass through the vRouter topology. In more detail, for scenarios 1 and 2 (Figure 8.1), VM #2 will establish a VPN session with the east-side router, while VM #3 will establish a VPN session with the west-side router. Both scenarios assume that the vRouter topology is located on the Data Network, therefore the same topology is utilised for both scenarios. On the contrary, for scenario 3 (Figure 8.2), the vRouter topology needs to be placed between VM #1 and VM #2, therefore VM #1 and VM #2 are initiating VPN sessions with the east-side and west-side VPN endpoints.

It is noteworthy that this setup introduces some challenges in respect to performance evaluation and KPI measurements. In particular, the performance and SLA-related KPIs (e.g., bandwidth, latency) are measured on the VMs where the UAV and 5G core related services reside. However, the VPN tunnels between the VMs and the vRouter topology introduce an additional uncertainty factor (e.g., due to variant network latency and limited upstream bandwidth from the Internet providers), which in turn will introduce bias in the KPI measurements. To overcome this limitation, it will be investigated for the KPI measurements to be obtained on the ingress and egress nodes of the vRouter topology rather than at the K3Y-hosted virtual machine endpoints, ensuring that the VPN-related bias will not affect the KPI measurements.

8.3.3 Realization of each UC application service

The network functions of the 5G core that are required to support UAV connectivity and mission inspection traffic will be virtualized and will contain an anchor user plane function (A-UPF) that will form a chain with intermediate user plane functions (I-UPF) that are deployed on edge networks. The 5G core will act as the convergence point between the access-side environment which contains the UAVs and the mission-management domain which contains the Swarm Manager, establishing a complete end-to-end data path across access, core, and application layers, forming the backbone of the system, enabling consistent packet delivery between UAVs and the Swarm Manager.

UAV assets will be emulated using software-based UAV instances deployed along an emulated radio access point which will be used with an intermediate UPF (I-UPF) chained via the N9 interface to a PDU Session Anchor UPF (A-UPF) within the 5G core. These generate continuous operational data streams, including position updates, system status, and mission-related telemetry. Command messages

originating from the Swarm Manager will traverse the 5G core before reaching the UAVs, forming a closed-loop control path. This setup enables bidirectional communication between UAVs and mission control across the 5G network, supporting realistic command-and-telemetry interaction without dependency on specific hardware platforms.

The Swarm Manager produces mission directives and receives UAV feedback via the 5G core network. These exchanges represent typical swarm coordination workflows, including task assignment, progress reporting, and status monitoring. At this stage, traffic generation focuses on predictable operational patterns to validate end-to-end reachability and application-layer integration. More complex types of traffic such as video streams will be deferred to the second evaluation of the use case scenarios 8.2, ensuring that a clear operational baseline is established first.

The vRouter topology will contain a set of Cisco XRd virtual routers, that will be used to emulate either a backhaul infrastructure that connects an intermediate UPF to an anchor UPF of a 5G core, or they will emulate the Data Network that will forward traffic from the 5G core network to the Swarm Manager. This arrangement introduces multiple forwarding stages between UAV endpoints and mission control, enabling observation of packet traversal across access, core, and application layers. During the baseline operational state of the testbed, the routing topology will be configured with conventional mechanisms and routing policies.

8.4 Use-Case Software Readiness and Staging

Figure 8.3 represents a staging environment of the K3Y tesbed that was used to validate the operational state of the UAV communications over 5G, including the components that will be used for the emulated UAV mission inspection use case. In the initial phase, and prior to the deployment to the ORO testbed, the setup consists of an entirely emulated setup using the GNS3 emulator. The GNS3 project replicates the entire setup envisioned for the use case, including the VMs of the use case components interconnected with conventional Cisco routers (replacing the vRouters). The routers are configured with OSPF and BGP, representing the backhaul and data network domains respectively.

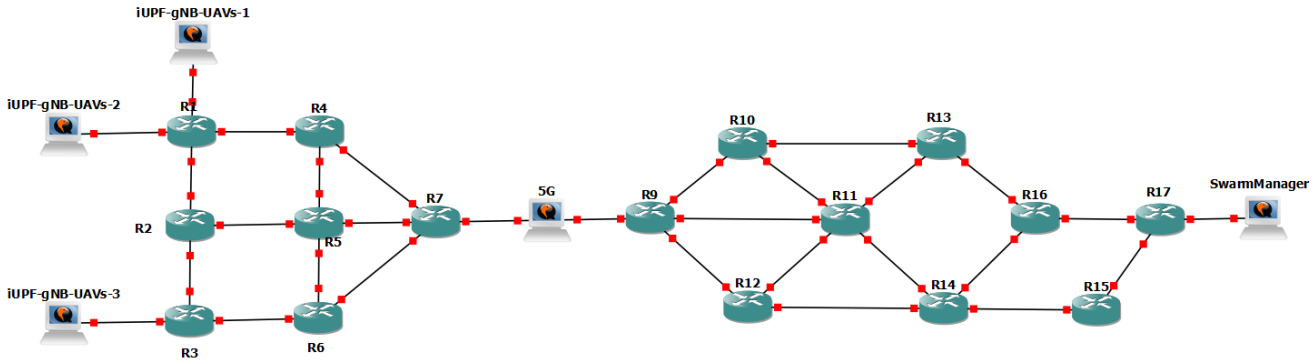

Figure 8.3: Initial implementation of the use case in GNS3

8.4.1 Baseline software capabilities

The following components are available or partially available for the baseline testbed:

- Swarm Manager: A lightweight component based on MAVProxy and QGroundControl that will be used for the coordination of UAVs and their respected mission. It will provide waypoint management, provide real-time monitoring analytics of the UAVs such as telemetry data, battery and health

metrics. It will include a GUI for supervising the mission, visualizing the UAV swarm and intervening manually when needed.

- 5G core network: A virtualized implementation of the free5gc 5G core network distribution that will provide software defined connectivity services for the UAVs of the use case. It will support scalable resource allocation to accommodate heterogeneous traffic profiles (e.g., video, telemetry, control), and enable experimentation with latency, reliability, and throughput optimization under realistic 5G conditions.
- UAVs: Emulated software-defined UAV instances using UERANSIM and Ardupilot that will replicate the behaviour, the dynamics, and the communication interfaces of real UAVs, simulating sensor outputs, telemetry data, and control responsiveness, enabling testing and validation of swarm management, mission planning, and network protocols in a controlled and repeatable environment.
- Edge network: A software-defined representation of a 5G edge network using UERANSIM and free5GC, combining key radio and core functions to provide a realistic testbed for UAV swarm operations. It integrates an emulated gNB for radio access and an emulated intermediate user plane function (I-UPF) for local traffic handling, enabling low-latency, high-throughput connectivity close to the UAVs.

To support the collection of metrics and KPI performance in this stage, a custom metrics collection software was developed, which is able to collect various metrics related to network performance, from the use case assets. The collection is performed directly from the swarm manager as well as from the 5G core. Figure 8.4 depicts the user interface of the metrics collection component, visualising all the collected data.

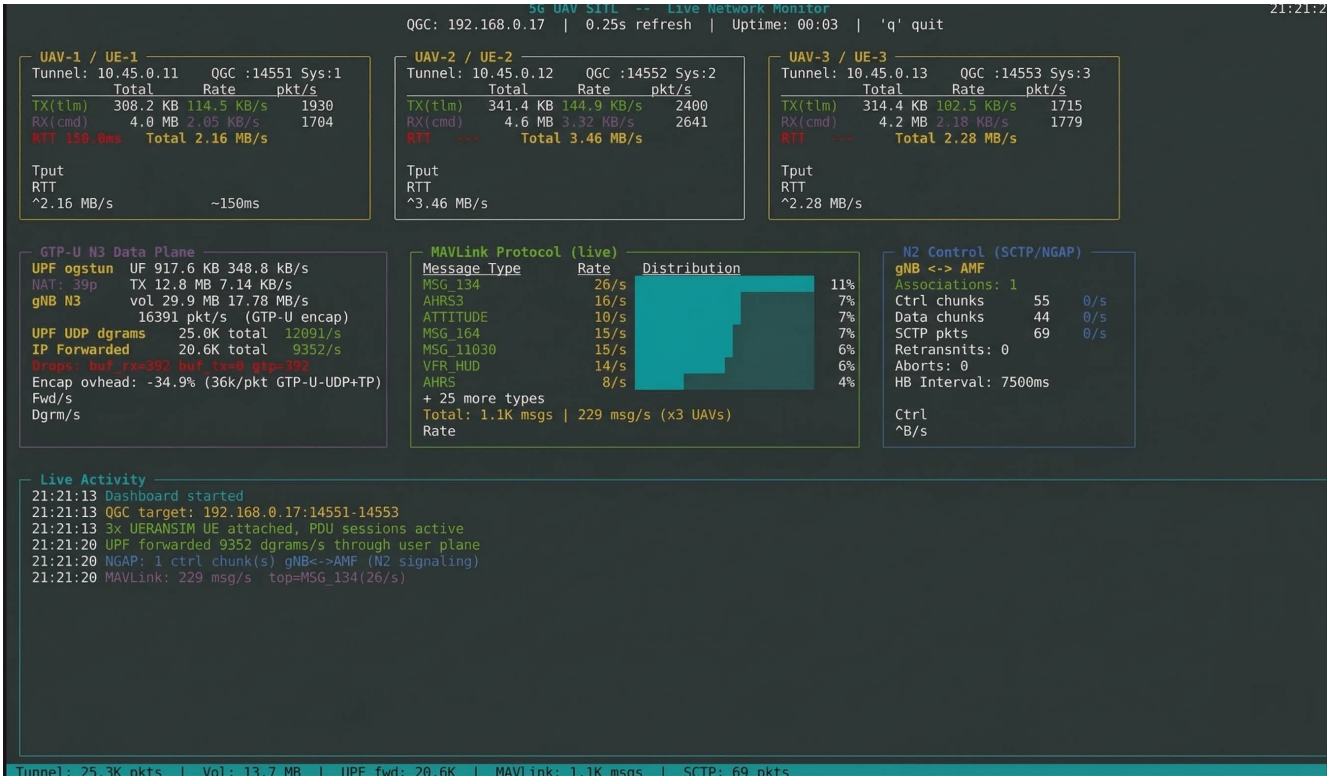

Figure 8.4: The user interface of the custom metrics collection component

In more detail, each drone communicates with QGroundControl via MAVLink protocol [14], with telemetry and command streams flowing through dedicated PDU sessions established by UERANSIM. The per-UE panels display bidirectional throughput and round-trip latency over the GTP-U user plane tunnels (N3

interface), giving immediate visibility into the quality of service experienced by each UAV. A dedicated MAVLink panel aggregates live packet captures across all three tunnel interfaces, breaking down message types (heartbeat, attitude, GPS, etc.) by rate and distribution to characterize the autopilot traffic profile in real time. At the infrastructure level, the dashboard monitors the GTP-U data plane between the gNodeB and the UPF. That includes encapsulated volume, IP forwarding rates, NAT translation counters, and socket-level drop statistics on port 2152. These provide us with insight into the tunneling overhead and potential bottlenecks within the 5G user plane. In parallel, the N2 control plane panel tracks SCTP/NGAP signaling between the gNodeB and the AMF, reporting control and data chunk rates, active associations, retransmissions, and aborts. Together, these views offer a comprehensive, protocol-layer-by-layer picture of a 5G-connected UAV swarm. From application-level drone telemetry down to the core network's control and user plane interfaces, all captured from a containerized Software in the loop (SITL) environment.

It should be noted that the SITL environment has currently been developed for monitoring network-related metrics experienced by the use case components participating in the UAV mission. For future evaluations that will need a complete end-to-end interaction with the CASTOR components, this software will be enhanced to integrate notifications and updates (e.g., SSLA violation notifications) from CASTOR.

8.4.2 Components under development or represented as stubs

- Mission inspection data generation: A realistic environment simulator will be investigated that will produce UAV mission inspection video streams. The simulator will model diverse operational scenarios (e.g., infrastructure inspection, surveillance, and search missions) and generate configurable video outputs in terms of resolution, frame rate, compression, and motion dynamics.

8.4.3 Evolution towards CASTOR-enabled experimentation

The K3Y testbed is designed in a modular way, accepting any routing topology to connect the services together, and does not require the functionality of CASTOR for the baseline operation. This approach decouples the development of UC workflow-specific artifacts (e.g., Swarm Manager, 5G core) from the network development and instantiation of the CASTOR integrated framework in the ORO testbed. This strategy allows both tracks to advance in parallel. Consequently, D6.2 marks the first release of the integrated CASTOR framework, where the overall application services leveraging CASTOR's trust-aware TE provisioning will be evaluated. This setup will enable experimentation in network behaviour when introducing CASTOR to the shared backhaul infrastructure, as well as the Data Network routing topologies in order to support the user stories defined in D2.1 [4].

UC4 Evaluation Plan			
User Story	**Description**	**1st Evaluation**	**2nd Evaluation**
UC4.US1a	Nominal Operation	Using the K3Y testbed connected to the ORO vRouter topology, the baseline latency between the 5G core and the Swarm Manager will be tested without using the CASTOR framework. In addition, the RTT of the telemetry commands from the Swarm Manager to the UAVs will also be measured.	During the second round of experiments, all infrastructure elements will also be equipped with TNDE-enriched capabilities leveraging the verifiable sharing of trustworthiness evidence through CASTOR's crypto agility layer. Furthermore, composite attestation for path-centric trust assessment will also be tested and compared against the baseline evaluation.
UC4.US1b	Fallback SSLA	Using the K3Y testbed connected to the ORO vRouter topology, latency inside the vRouter topology will be measured. Moreover, the behaviour of the RTL will be evaluated on its correctness, and the SSLA violation accuracy will be measured.	The CASTOR framework will be evaluated to measure its impact on latency and the establishment of alternative routing paths, while considering the overhead from the novel cryptographic schemes in the routing plane.
UC4.US1c	SSLA Compliance Audit	Using the K3Y testbed connected to the ORO vRouter topology, the latency of accessing SSLA compliance information will be evaluated in cases where the information is available either through the Trust Awareness API.	The CASTOR framework will be evaluated in its ability to establish alternative paths. Moreover, the SSLA violation mechanism through the DLT-based Trust Exposure Layer will also be evaluated.
UC4.US2a	Risk-aware path selection	Using the K3Y testbed connected to the ORO vRouter topology, the impact of the risk assessment process on the RTL calculation will be measured. The risk assessment will account for additional demands that may stem from a vulnerable profile of the source.	The CASTOR framework will be evaluated based on its capacity to select alternative paths based on the Risk Assessment.
UC4.US3a	Nominal Operation	This scenario will not be evaluated in the first phase.	Using the K3Y testbed connected to the ORO vRouter topology, the baseline latency between the I-UPF and the A-UPF of the 5G core will be tested without using the CASTOR framework. In addition, system reliability will be assessed along with scalability under operation conditions.
UC4.US3b	Optimal path selection	This scenario will not be evaluated in the first phase.	Using the K3Y testbed integrated with the ORO vRouter topology, latency within the data network between the I-UPF and the A-UPF will be systematically evaluated. In parallel, the reliability of communications will be measured to ensure consistent performance. The CASTOR framework will be evaluated in its ability to select the optimal path in the shared backhaul infrastructure.

Table 8.2: UC4 Evaluation plan

Chapter 9

Summary and Conclusions

In conclusion, Deliverable D6.1 represents the first significant milestone toward the realization of the CASTOR integrated framework, successfully transitioning the project from theoretical architectural modelling to a functional and verifiable prototype release. By consolidating the blueprint for all technical artifacts and establishing the PoC Playground for simulation-driven validation, the document operationalizes the CASTOR lifecycle and its associated trust-aware TE provisioning within a controlled environment. This progress is underpinned by the demonstrated readiness of a multi-tiered validation framework, anchored by a production-grade primary testbed and supported by specialized auxiliary sites for hardware-rooted trust and advanced orchestration. The strategic mapping of the four CASTOR use case scenarios onto this robust infrastructure ensures that the technical foundation is fully established for the evaluation phases scheduled for D6.2 and D6.3.

The interim verification results reported in this deliverable confirm that the CASTOR Integrated Framework is technically viable and capable of addressing the trust management challenges inherent in the compute continuum. The modular design allows for the incremental introduction of trust-aware functionalities without disrupting existing infrastructure. The framework's ability to dynamically adapt to degraded trust levels through reactive reconfiguration is critical for maintaining the resilience of safety-critical applications in multi-actor environments.

List of Abbreviations

Abbreviation	Translation
3GPP	3rd Generation Partnership Project
5GAA	5G Automotive Association
5GC	5G Core
AA	Authorisation Authority
ABAC	Attribute-Based Access Control
AMF	Access and Mobility Management Function
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
AS	Application Server / Autonomous System
ASGI	Asynchronous Server Gateway Interface
AT	Authorisation Ticket
ATL	Actual Trust Level / Actual Level of Trust
A-UPF	Anchor User Plane Function
AUSF	Authentication Server Function
BBU	Baseband Unit
BGP	Border Gateway Protocol
BGP-LS	Border Gateway Protocol Link-State
BSID	Binding Segment Identifier
CA	Certificate Authority
CAM	Cooperative Awareness Messages
CAV	Connected and Automated Vehicles
CC	Compute Continuum
CCAM	Connected, Cooperative, and Automated Mobility
CER	Central Emergency Responder
CFA	Control Flow Attestation
CFSM	Communicating Finite State Machine
CIM	Coherent Ising Machine
CISP	Common Information Service Provider
C-ITS	Cooperative Intelligent Transportation Systems
CIV	Configuration Integrity Verification
CLI	Command Line Interface
CMS	Certificate Management System
CN	Core Network
CNI	Container Network Interfaces
CPM	Collective Perception Messages
CPU	Central Processing Unit
CRL	Certificate Revocation List
CRL-CA	Certificate Revocation CA

CSC	Communication Service Customer
CSP	Communication Service Provider / Cloud Service Provider
DENM	Decentralized Environmental Notification Messages
DLT	Distributed Ledger Technology
DoS	Denial of Service
DRB	Data Radio Bearer
DRF	Django REST Framework
E2E	End-to-End
EA	Enrollment Authority
eBPF	extended Berkeley Packet Filter
ECA	Enrolment Certification Authority
EJBCA	Enterprise Java Beans Certificate Authority
ETSI	European Telecommunications Standards Institute
ETSI OSG OSM	ETSI Open Source MANO
EVPN	Ethernet VPN
FAD	Flexible Algorithm Definition
FIB	Forwarding Information Base
Flex-Algo	Flexible Algorithm
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GCS	Ground Control Station
GNB	Next Generation Node B
gNMI	gRPC Network Management Interface
GNS3	Graphical Network Simulator-3
GPU	Graphics Processing Unit
gRPC	gRPC Remote Procedure Call
GSP	Geo Service Provider
GTPU	GPRS Tunneling Protocol – User plane
HTTP	Hypertext Transfer Protocol
IBN	Intent-Based Networking
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IP	Internet Protocol
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
ITS	Intelligent Transportation Systems
I-UPF	Intermediate User Plane Function
JWT	JSON Web Token
KAFKA	Apache Kafka (distributed event-streaming platform)
KPI	Key Performance Indicator
L1	Layer 1
LAN	Local Area Network
LDP	Label Distribution Protocol
LER	Local Emergency Responder / Error
LSA	Link State Advertisements
MANO	Management and Orchestration
MANET	Mobile Ad Hoc Networks
MNO	Mobile Network Operator

MORP	Multi-Objective Routing Optimization Problem
MPLS	Multi-Protocol Label Switching
NaaS	Network as a Service
NEF	Network Exposure Function
NEP	Network Equipment Provider
NETCONF	Network Configuration Protocol
NFV	Network Function Virtualization
NGAP	Next Generation Application Protocol
NOP	Network Operator
NSO	Network Service Orchestrator
NTP	Network Time Protocol
OBU	Onboard Unit
OSPF	Open Shortest Path First
OSS/BSS	Operational Support System/Business Support System
OTA	Over-the-Air
PCC	Path Computation Client
PCE	Path Computation Element
PCEP	Path Computation Element Communication Protocol
PDU	Packet Data Unit
PKI	Public Key Infrastructure
PoC	Proof of Concept
QoS	Quality of Service
RAM	Random Access Memory
RAN	Radio Access Network
RAT	Remote Attestation Procedure
RBAC	Role Based Access Control
RCA	Root Certificate Authority
REST	REpresentational State Transfer
RIB	Routing Information Base
RIP	Routing Information Protocol
RoT	Root of Trust
RSU	Road Side Unit
RSVP-TE	Resource Reservation Protocol for Traffic Engineering
RTL	Required Trust Level
RTT	Round Trip Time
SCB	Security Context Broker
SCMS	Security Credential Management System
SCTP	Stream Control Transmission Protocol
SDN	Software-Defined Networking
SFC	Service Function Chain
SGX	Software Guard Extensions
SID	Segment Identifier
SITDS	Service Intent Translation & Decomposition Service
SITL	Software In The Loop
SLA	Service Level Agreement
SLO	Service Level Objective
SMD	Security Management Domain
SMF	Session Management Function
SNPN	Standalone Non-Public Network

SotA	State-of-the-Art
SPF	Shortest Path First
SR	Segment Routing
SRGB	Segment Routing Global Block
SR-MPLS	Segment Routing over MPLS
SR-TE	Segment Routing Traffic Engineering
SRv6	Segment Routing over IPv6
SSD	Solid State Drive
SSH	Secure Shell
SSI	Self-Sovereign Identity
SSLA	Security Service Level Agreements
SSLO	Security and Service Level Objective
SST	Single Slice Selection Assistance Information
SUMO	Simulation of Urban Mobility
TAF	Trust Assessment Framework
TCB	Trusted Computing Base
TCP	Transmission Control Protocol
TDX	Trust Domain Extensions
TE	Traffic Engineering
TED	Traffic Engineering Database
TEE	Trusted Execution Environment
TEPE	Traffic Engineering Policy Engine
TGCE	Topology Graph Composition Engine
TN-DSM	Trust Network Device Security Monitor
TNDE	Trust Network Device Extension
TNDI	Trust Network Device Interfaces
TNDI-SP	TNDI Security Protocol
TO	Traffic Operator
TPL	Trust Policy Language
TPM	Trusted Platform Module
TPR	Trusted Path Routing
TSM	Trust Source Manager
TTL	Time To Live
UAM	Urban Air Mobility
UAV	Unmanned Aerial Vehicle
UDM	Unified Data Management
UDR	Unified Data Repository
UE	User Equipment
UPF	User Plane Function
URLPC	Ultra Reliable Low Power Communications
USSP	U-Space Service Supplier
UTM	Unmanned Traffic Management
V2I	Vehicle-to-Infrastructure
V2N	Vehicle-to-Network
V2V	Vehicle-to-Vehicle
V2X	Vehicle to Everything
VC	Verifiable Credentials
VM	Virtual Machine
VNF	Virtual Network Function

VPN	Virtual Private Network
VRAM	Video Random Access Memory
VRF	Virtual Routing and Forwarding
VRU	Vulnerable Road User
VXLAN	Virtual Extensible LAN
WG	Working Group
WIP	Work In Progress
WP	Work Package
ZK	Zero Knowledge
ZKP	Zero Knowledge Proof

Bibliography

[1] 3rd Generation Partnership Project (3GPP). 3GPP TS 28.541: Management and orchestration; 5g network resource model (nrm); stage 2 and stage 3. Technical Specification (TS) 28.541, Release 15, 2020. Version 15.6.0.

[2] 3rd Generation Partnership Project (3GPP). 3GPP TS 28.531: Management and orchestration; provisioning. Technical Specification (TS) 28.531, Release 16, 2022. Version 16.11.0.

[3] CASTOR. Architectural specification of castor continuum-wide trust assessment framework. Deliverable 4.1, The CASTOR Consortium, 12 2025.

[4] CASTOR. Operational landscape, requirements and reference architecture - initial version. Deliverable 2.1, The CASTOR Consortium, 11 2025.

[5] CASTOR. Architectural specification of dynamic enforcement of trust-/network-aware path establishments. Deliverable 5.1, The CASTOR Consortium, 2 2026.

[6] CASTOR. Conceptual architecture of castor trusted computing base & composable attestation model specification. Deliverable 3.1, The CASTOR Consortium, 2 2026.

[7] CASTOR. Device-level & continuum-wide trust extensions and security controls (first release). Deliverable 3.2, The CASTOR Consortium, 03 2026.

[8] CASTOR. Trusted path establishment building blocks, optimization engine & crypto structures for trusted data sharing. Deliverable 4.2, The CASTOR Consortium, 4 2026.

[9] European Telecommunications Standards Institute (ETSI). ETSI GS NFV-SOL 005 V2.6.1: Network functions virtualisation (nfv) release 2; protocols and data models; restful protocols specification for the management and orchestration (mano) interface. Standard GS NFV-SOL 005 V2.6.1, 2019.

[10] European Telecommunications Standards Institute (ETSI). ETSI GS NFV-SOL 006 V2.7.1: Network functions virtualisation (nfv) release 2; protocols and data models; nfv descriptors based on yang specification. Standard GS NFV-SOL 006 V2.7.1, 2019.

[11] European Telecommunications Standards Institute (ETSI). Intelligent Transport Systems (ITS); Security; Trust and Privacy Management; Release 2. Technical Specification (TS) 102 941, November 2022. V2.2.1.

[12] Keyfactor. EJBCA - Open Source PKI Certificate Authority. https://www.ejbca.org/, 2026. Accessed: April 6, 2026.

[13] E. Mannie and D. Papadimitriou. Recovery (protection and restoration) terminology for generalized multi-protocol label switching (GMPLS). RFC 4427, Internet Engineering Task Force (IETF), 2006.

[14] Nakul Nair, K. B. Sareth, Rao R. Bhavani, and Ashish Mohan. *Simulation and Stabilization of a Custom-Made Quadcopter in Gazebo Using ArduPilot and QGroundControl*, page 191–202. Springer Nature Singapore, 2022.