D2.3 1ST RELEASE OF THE UNITY-6G ARCHITECTURE

Work package	WP2
Task	T2.2, T2.3
Due date	31/03/2026
Submission date	31/03/2026
Deliverable lead	TID
Version	1.0
Dissemination Level	Public
Editor	Javier Velázquez Martínez (TID)
Authors	Javier Velázquez Martínez (TID), Luis M. Contreras Murillo (TID), Harilaos Koumaras (NCSRD), Stefanos Plastras (NCSRD), Cristian Brunetto (LINKS), Carla Amatetti (UNIBO), Maria A. Serrano (NBC), Michalis Dalgitsis (NBC), Alice Piemonti (Martel), Engin Zeydan (CTTC), Jorge Baranda (CTTC), Luis Blanco (CTTC), Łukasz Kułacz (PUT), Paweł Płaczkiewicz (PUT), Adrian Kliks (PUT), Jetmir

	Haxhibeqiri, Vasilis Maglogiannis, Dries Naudts, Jeroen Hoebeke (IMEC), Nikolaos Kanakaris, Nikolaos Stamatiou (IPTO), Aitor Alcazar Fernández (TID) , Balaji Kirubakaran (DLR), Purva Joshi (DLR), Berk Buzcu (HES-SO), Sotirios Spantideas (FDI), Anastasios Giannopoulos (FDI), Marilina Bartsioka (FDI), Panagiotis Trakadas (FDI), Efi Dvir (CER), Eddi Gonzalez (HSA)
Reviewers	Gil Kedar (CER), Sihem Cherrared (Orange), Tomaso de Cola (DLR), Matteo Pagin (KEY), Hua Wang (KEY), Nikolaos Kanakaris (IPTO), Javier Velázquez Martínez (TID) Gianluca Fontanesi (NokiaBL), Aitor Alcazar Fernandez (TID)

Abstract	This deliverable presents the first release of the UNITY-6G architecture, outlining its main components, interfaces, and layers. It describes the architecture's defining principles as well as the support it provides to the UNITY-6G use cases.

Keywords	service-based architecture, cross-domain orchestration, AI-native, semantic communications, digital twin, data continuum, zero-trust, sustainability

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	14/01/2025	1st version of the template for comments	Paula Ando (Martel)
V.0.2	9/12/2025	1st Version with ToC and listed contributors	Javier Velazquez (TID)
V.0.3	9/02/2026	1st Draft Version with all contributions included	All authors
V0.4	20/02/2026	1st Review	Reviewers
V0.5	9/03/2026	2nd Version with addressed comments	All authors
V0.6	16/03/2026	2nd Review	Reviewers
V0.7	20/03/2026	Final Version with addressed comments	All authors
V1.0	31/03/2026	Final review and submission	Javier Velázquez Martínez (TID), Engyn Zeydan, Luis M. Contreras Murillo (TID)

DISCLAIMER

Co-funded by the European Union

Project funded by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI

The **unity-6G** project has received funding from the **Smart Networks and Services Joint Undertaking (SNS JU)** under the European Union's **Horizon Europe research and innovation programme** under Grant Agreement No *101192650*. This work has received funding from the **Swiss State Secretariat for Education, Research, and Innovation (SERI)**.

COPYRIGHT NOTICE

EXECUTIVE SUMMARY

This deliverable presents the first release of the UNITY-6G architecture, designed to address the increasing complexity, heterogeneity, and intelligence requirements of emerging 6G networks. As current 5G management frameworks remain fragmented across isolated domains, UNITY-6G introduces a unified, AI-native, cross-domain orchestration approach capable of coordinating terrestrial, non-terrestrial, RAN, transport, core, edge, and vertical application environments. The architecture builds on state-of-the-art frameworks such as MonB5G, aeriOS MetaOS, 3GPP's early 6G directions, and TMForum's principles, taking these foundations forward into a coherent, scalable, sustainable, and trustworthy 6G system.

At the heart of UNITY-6G is a service-based management paradigm where all capabilities across domains are exposed and consumed as modular services. Intelligence is embedded throughout the system using a common MS–AE–DE–ACT tuple model (Monitoring, Analytics, Decision, Actuation), enabling distributed autonomous agents that learn, reason, and act across multiple layers of the network. The architecture also integrates semantic communications to reduce data load and improve task relevance, allowing applications to operate efficiently even under spectrum or latency constraints, particularly relevant when combining terrestrial and non-terrestrial segments.

A key innovation of the architecture is the Inter-Domain Management Orchestrator (IDMO), which provides intent-based end-to-end orchestration, supported by Domain-specific Management Orchestrators (DMOs) and Infrastructure Domain Managers (IDMs). This hierarchical structure ensures consistent lifecycle management, conflict handling, distributed AI workflows, and real-time adaptation across highly diverse network components. Furthermore, UNITY-6G integrates energy sustainability as a native architectural objective by introducing Virtual Power Plants and energy-aware orchestration strategies that optimize resource distribution based not only on performance needs but also on environmental conditions and renewable energy availability.

Trust and security are treated as first-class architectural elements rather than afterthoughts. The Trust Layer applies Zero-Trust principles, verifiable identities, and evidence-driven trust scoring, combined with DLT-based immutability and auditability. This ensures safe cross-domain interactions, protects AI-driven decision-making, and preserves integrity across the data continuum. Complemented by a fully federated semantic data layer, the architecture

enables unified data visibility, consistent learning pipelines, and cooperation between digital twins, AI agents, and orchestration functions.

Finally, the architecture is validated against four demanding 6G use cases: disaster-resilient connectivity, real-time XR holography, digital twin for network evaluation, and multi-RAT Industry 4.0 networks; demonstrating how UNITY-6G's cross-domain coordination, intelligence distribution, semantic optimization, and trust-driven governance can support highly dynamic and mission-critical environments. These use cases guide the specification of flows, operational lifecycles, and interface behaviors that will shape the next architectural release in Deliverable D2.4.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

3GPP	3rd Generation Partnership Project
5G	Fifth Generation
6G	Sixth Generation
AADE	Awareness, Analysis, Decision-making, and Execution
AI	Artificial Intelligence
AIaaS	AI/ML as a Service
ACT	Actuator
AE	Analytics Engine
AMF	Access and Mobility Management Function
ANP	Autonomous Network Project
AoI	Age of Information
API	Application Programming Interface
B5G	Beyond 5G
BSS	Business Support System
CFS	Customer-Facing Services
CN	Core Network
CNC	Central Network Controller
CNF	Cloud-native Network Function
CP	Control Plane
CU	Central Unit
CUC	Central User Controller
DB	Database
DCCF	Data Collection Coordination Function
DDF	Domain Data Fabric
DDoS	Distributed Denial-of-Service
DE	Decision Engine
DID	Decentralized Identifiers
DLT	Distributed Layer Technology
DMO	Domain Management Orchestrator
DT	Digital Twin
DU	Distributed Unit
E2E	End-to-End
EMS	Energy Management System
ETA	Estimated Time of Arrival
ETSI	European Telecommunications Standards Institute
FaaS	Function as a Service
FCAPS	Fault, Configuration, Accounting, Performance and Security
GEO	GEostationary Earth Orbit
GTP	GPRS Tunnelling Protocol
HAP	High Altitude Platform
HLO	High-Level Orchestrator

HTTP	Hypertext Transfer Protocol
IDM	Infrastructure Domain Manager
IDMO	Inter-Domain Management Orchestrator
IE	Infrastructure Element
IMS	IP Multimedia Subsystem
IoT	Internet of Things
IP	Internet Protocol
ISM	In-Slice Management
KPI	Key Performance Indicator
KVI	Key Value Indicator
LCM	Lifecycle Management
LDAP	Lightweight Directory Access Protocol
LEO	Low Earth Orbit
LLO	Low-Level Orchestrator
LoT	Level of Trust
MANO	Management and Orchestration
MAPE	Monitor-Analyse-Plan-Execute
MEO	Medium Earth Orbit
ML	Machine Learning
MS	Monitoring System
NAS	Non-Access Stratum
NBO/NBA	Next Best Offer/Action
NFs	Network Functions
NFV	Network Function Virtualization
NFVO	NFV Orchestrator
NPN	Non-Public Network
NRF	Network Repository Function
NSC	Network Slice Controller
NTN	Non-Terrestrial Networks
NWDAF	Network Data Analytics Function
ODA	Open Digital Architecture
OSS	Operations Support System
PCFP	Packet Forwarding Control Protocol
PoC	Proof of Concept
PLMN	Public Land Mobile Network
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RAT	Radio Access Technology
REST	Representational State Transfer
RF	Radio Frequency
RFS	Resource-Facing Services
RIC	RAN Intelligent Controller
RL	Reinforcement Learning

RU	Radio Unity
SA	Service Adapter
SBA	Service-based Architecture
SBMA	Service-Based Management Architecture
SDLC	Service Development Lifecycle
SDN	Software-Defined Networking
SEPP	Security Edge Protection Proxy
SMB	Service-based Management Bus
SMO	Service Management and Orchestration
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TSO	Transmission System Operator
TN	Transport Network
UE	User Equipment
URLCC	Ultra Reliable Low Latency Communications
UP	User Plane
UPF	User Plane Functions
VC	Verifiable Credentials
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
VPN	Virtual Private Network
VPP	Virtual Power Plant
WAB	Wireless Access Backhaul
XR	eXtended Reality
ZSM	Zero-touch network and Service Management
ZTA	Zero Trust Architecture

1 INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

The evolution toward sixth generation (6G) networks introduces unprecedented requirements for scalability, openness, and sustainability across heterogeneous domains, including the Radio Access Network (RAN), Core Network (CN), and Transport Network (TN). Emerging use cases such as extended reality, disaster response, digital twins, and industrial automation demand not only ultra-reliable, low-latency communication but also intelligent and sustainable service delivery across distributed infrastructures. Current 5G management frameworks, though advanced, are limited in their ability to provide holistic, cross-domain orchestration, particularly regarding end-to-end (E2E) service continuity, dynamic adaptability, and environmental considerations.

This deliverable D2.3 focuses on developing an initial release of a comprehensive architecture for the integrated UNITY-6G network environment, including the different components, interfaces, and protocols that will be used.

The main motivation for this work is to address the fragmentation of current management and orchestration systems and to enable cross-domain intelligence for 6G networks. Existing frameworks operate in isolation, focus on specific domains (such as the RAN or CN), and often use static interfaces and hierarchical control models that limit scalability and flexibility. This fragmented approach limits seamless end-to-end orchestration and prevents dynamic service continuity when resources must be relocated or reconfigured across domains.

Additionally, current frameworks lack standardized mechanisms for integrating AI-driven closed loops that coordinate learning and decision-making across different domains. As a result, the potential of distributed intelligence is underutilized, with most optimizations restricted to local or domain-specific scopes. Meanwhile, sustainability is becoming a fundamental design criterion for future networks, yet existing orchestration mechanisms still provide limited support for energy-aware decisions or environmentally optimized service management.

These gaps motivate the design of a Service-Based Management Architecture (SBMA) that enables intelligent, cross-domain, and sustainable orchestration. SBMA adopts a service-based paradigm to abstract domain heterogeneity, integrates distributed AI-driven control, and provides a foundation for realizing the unified management vision outlined in recent 6G architectural work.

1.2 DELIVERABLE STRUCTURE

This deliverable is structured as follows:

- Section 2 presents some relevant State-of-the-Art architectures that helped shaping UNITY-6G framework such as MonB5G [1], aeriOS [2], 3GPP [3] and TMForum [4].
- Section 3 shows a high-level description of the architecture with its general components outlining its main architectural principles: service-based approach, AI-native approach, semantic communications, scalability and sustainability.
- Section 4 deeps into the 4 layers of the architecture: management and orchestration layer, unification layer, data continuum and trust layer, while detailing the subcomponents and domains that define it.
- Section 5 links the developed architecture with the four use cases that UNITY-6G addresses, defining the functional flows and operational lifecycle for each use case.
- Section 6 concludes the deliverable by summarizing the main architectural contributions of the UNITY-6G framework as well as listing the planned enhancements for the final release of the architecture

1.3 RELATION TO OTHER WORK PACKAGES

D2.3 contains relationships with most of the work packages (WPs), as the architecture is the core basis for most of the work. This deliverable is part of WP2, where use cases, requirements and KPIs are defined, as well as the architecture is developed through three tasks, involving two deliverables D2.3 and D2.4, containing the first and the final release of the UNITY-6G architecture, respectively.

WP5 has a strong relationship with this deliverable, as it contains and specific task for developing a Service Based Architecture (SBA), which is one of the core principles of the architecture. WP3 develops in detail all the components and interfaces presented in the architecture along this document. WP4 focuses on working on the AI intelligence of the architecture. Lastly, WP6 works on developing several scenarios and proof of concepts that implement the architecture defined in this deliverable. Figure 1-1 shows how the work packages in UNITY-6G are related.

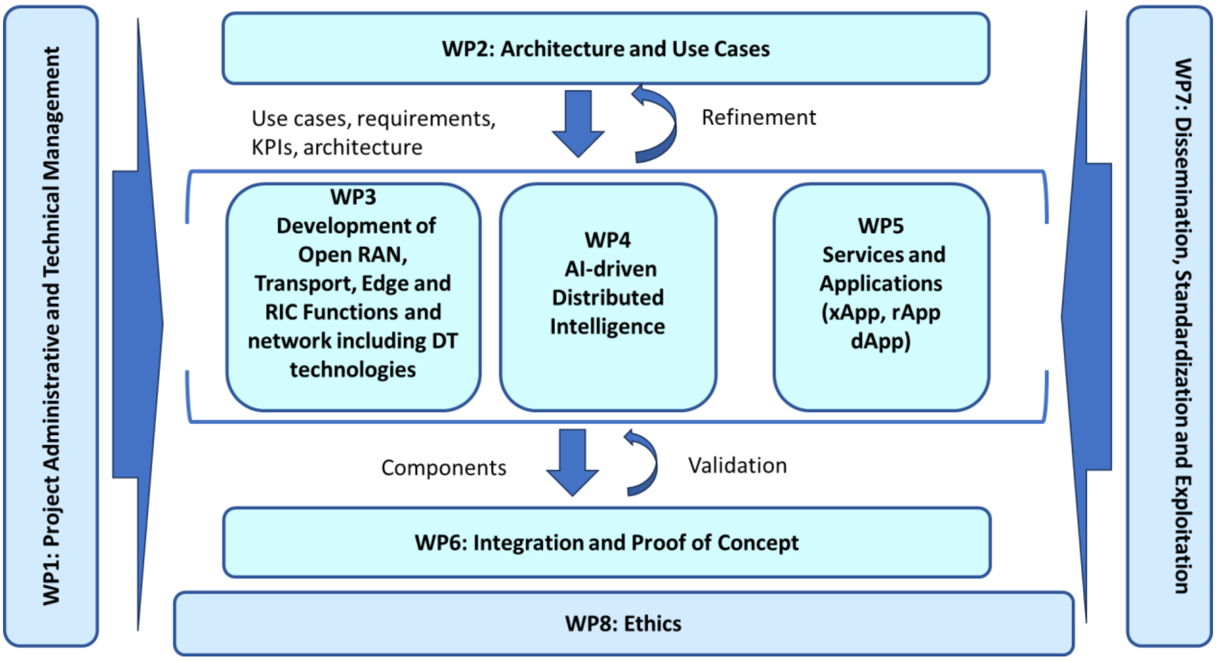


Figure 1-1. UNITY-6G Work Packages

2 STATE-OF-THE-ART

The State-of-the-Art underlying UNITY-6G spans several complementary frameworks, architectures, and standardization efforts that each address different aspects of next-generation network automation, intelligence, and continuum-scale integration. Before examining them individually, it is important to provide a clear, high-level overview to help the reader understand why these specific entries were selected and how they collectively shape the foundations of UNITY-6G.

MonB5G brings a fully distributed, AI-driven, hierarchical management framework based on the MAPE paradigm and tailored for large-scale, multi-domain network slicing. Its decomposition of monitoring, analytics, decision, and actuation functions into autonomous subsystems directly inspires UNITY-6G's adoption of distributed MS-AE-DE-ACT tuples and its emphasis on multi-level, zero-touch orchestration.

aeriOS and the broader MetaOS continuum ecosystem contribute an architectural vision that unifies IoT, edge, cloud, and programmable networks into a single federated execution substrate. aeriOS, in particular, provides continuum-native abstractions, observability, secure federation, and 5G-aware programmability (NEF/CAPIF, SDN/NFV, TSN), all of which strongly align with UNITY-6G objectives. These concepts help shape UNITY-6G's data continuum, cross-domain orchestration patterns, and integration of digital twins, trust, and sustainability mechanisms.

3GPP's early 6G studies introduce an evolution of the 5G Service-Based Architecture toward a more modular, data-centric, and AI-assisted core network. They explore optionalized network functions, enhanced user-plane programmability, intelligent exposure, and multi-agent coordination models, directions that directly correlate with UNITY-6G's focus on intent-based orchestration, AI-native agents, and multi-domain programmable control.

TM Forum ODA and its Autonomous Networks Project (ANP) offer a cloud-native, componentized operating architecture built around standardized Open APIs, intent management, and closed-loop automation. These principles support UNITY-6G's need for consistent cross-domain lifecycle management, multi-vendor interoperability, and the clean separation of layers.

Together, these SotA pillars define the technological baseline upon which UNITY-6G is built: AI-native control loops, continuum-wide orchestration, service-based and intent-driven management, modularity and programmability inspired by 3GPP evolution, and trust and data-centric operations. The following subsections provide the detailed analysis of each framework, highlighting where UNITY-6G adopts, extends, or generalizes their concepts.

2.1 MONB5G

The MonB5G [5] [1] architecture is a hierarchical, fault-tolerant, and AI-driven framework designed to achieve Zero-touch network and Service Management (ZSM) for massive-scale network slicing in Beyond 5G (B5G) environments. By capitalizing on artificial intelligence, the architecture enables networks to gain self-configuration, self-monitoring, self-healing and self-optimization capabilities, eliminating the need for human intervention.

The architecture is fundamentally built upon the MAPE (Monitor-Analyse-Plan-Execute) paradigm, implemented in a distributed and programmable manner through multiple AI-driven operations. To ensure carrier-grade performance and scalability, MonB5G decomposes its logical entities into distributed, interacting components executed at various levels: the OSS/BSS level, inside the virtualized infrastructure, and embedded directly within network slices.

A central feature of this design is the hierarchical control scheme, which utilizes fast local control loops for immediate management and slow wider-scope loops for broader coordination. This time-scale decomposition limits interference between different feedback-based decisions and ensures system stability in a non-linear environment.

The MonB5G system is organized into four primary functional blocks that collaborate to manage the lifecycle and security of network slices:

- **Monitoring System (MS):** This block gathers detailed information on network slices across multiple technical and administrative domains. It is built as a cloud-native application using microservices architecture (e.g., Kubernetes) to facilitate real-time monitoring and linear scalability.
- **Analytics Engine (AE):** The AE employs AI techniques (e.g., Federated Learning, Graph Neural Networks, and Distributed Deep Neural Networks) to provide slice-specific KPI predictions and anomaly detection. It focuses on distributing analysis tasks to local entities near the data source to minimize communication overhead and response latency.

- **Decision Engine (DE):** Leveraging inputs from the AE, the DE uses data-driven algorithms (like Deep Reinforcement Learning) for zero-touch lifecycle management, including slice admission control, scaling, and migration. These engines can be distributed flexibly at various granularities, such as per tenant, per slice, or per Virtual Network Function (VNF).
- **Actuators (ACT):** These are responsible for executing the management actions decided by the DE to maintain the performance and security of the network.

Some key architectural features are the following:

1. **Distribution of management operations:** Management tasks are AI-driven and distributed across nodes, slices, and orchestration domains. This local processing reduces the need for constant management information exchange, enhancing overall system scalability.
2. **Strong separation of concerns:** The framework distinguishes between the Lifecycle Management (LCM) handled by domain OSS/BSS and the runtime management of slices. OSS/BSS focuses on resource management and LCM but remains agnostic to the specific runtime operations of individual slices.
3. **In-Slice Management (ISM):** A novel concept where each slice template includes its own management platform responsible for FCAPS (Fault, Configuration, Accounting, Performance, and Security). This provides isolation of the management planes, a feature not natively provided by standard ETSI NFV MANO or 3GPP architectures.
4. **Hierarchical, end-to-end orchestration:** The architecture incorporates multiple domain-level orchestrators and a single master orchestrator. This multi-tier approach supports domain-specific slice templates, further contributing to the scalability of orchestration tasks.
5. **Security and energy efficiency by design:** Security is bolstered by the isolation of management spaces through ISM and AI-based closed control loops that detect and mitigate attacks (e.g., DDoS or LTEr attacks). Additionally, the architecture includes energy-aware orchestration that can dynamically deploy services to minimize energy consumption and latency through techniques like VNF clustering.

The MonB5G management and orchestration framework is designed to satisfy the essential requirements of ETSI ZSM. It also supports Management as a Service (MaaS), enabling the creation of dedicated "management slices" for runtime management by third-party Slice

Management Providers. A dedicated Slice Tenant Portal is provided to allow tenants to request slice deployments based on templates and manage their slices' lifecycles.

The main feature of the MonB5G framework is the decomposition of the management system into a set of cooperating/federated and autonomous AI-driven distributed management subsystems (see Fig. 2.1). Each of the Management and Orchestration subsystems is focused on a specific goal. Such an approach fits well with the multi-provider/multi-operator virtualization ecosystem (see Fig 2.2)

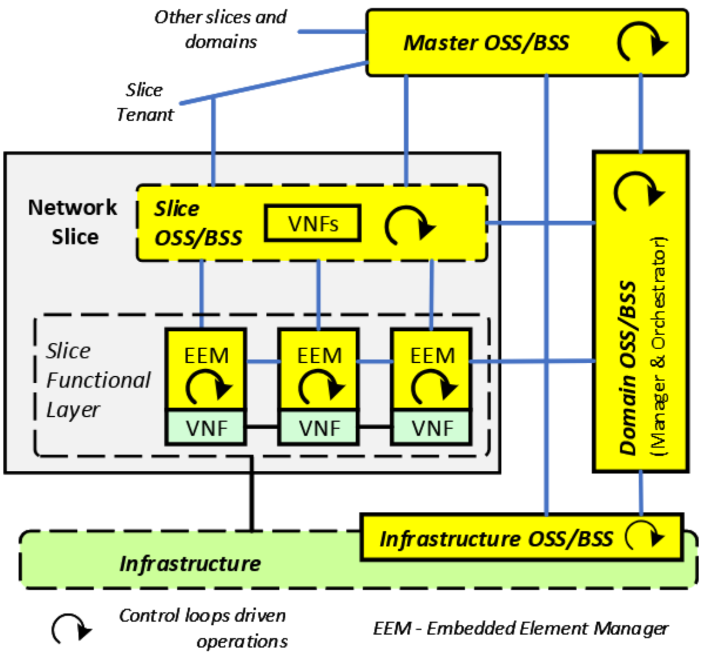


Figure 2.1. Vision of the MonB5G project. The yellow blocks show distributed management plane. A single domain case.

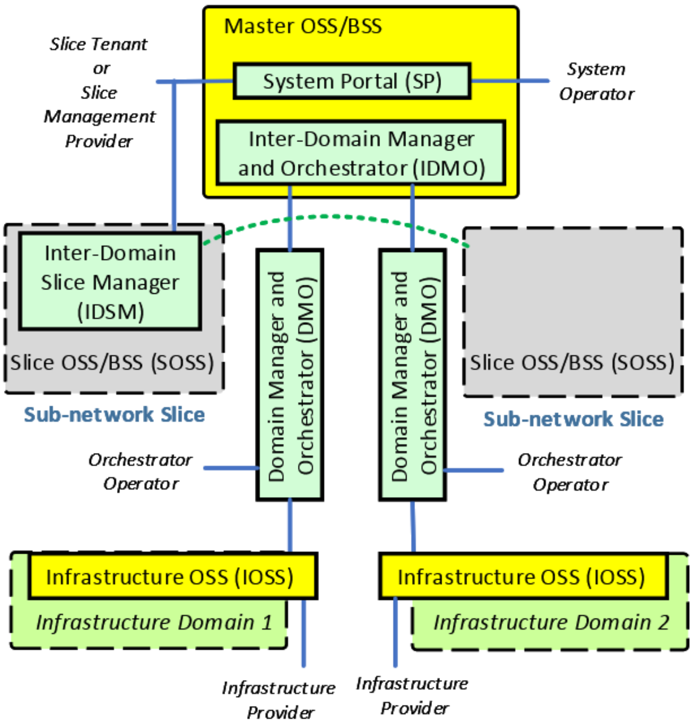


Figure 2.2. Multi-domain architecture in a multi-provider environment.

2.2 AERIOS CONTINUUM

The European Meta Operating System (MetaOS) landscape has been significantly shaped by a set of Horizon Europe projects funded under the HORIZON-CL4-2021-DATA-01-05 call, all of which address the cloud–edge–IoT continuum from complementary architectural and strategic perspectives. Among these, aeriOS, NEMO[1], NEPHELE[2], ICOS[3] , FLUIDOS[4] and NebulOuS[5]all propose meta operating system–like abstractions over heterogeneous

[1] https://cordis.europa.eu/project/id/101070118

[2] https://nephele-project.eu/

[3] https://cordis.europa.eu/project/id/101070177

[4] https://cordis.europa.eu/project/id/101070473

[5] https://nebulouscloud.eu/

infrastructures [6] [7].However, they differ substantially in how “OS-like” their approach is, whether their primary driver is AIoT enablement, sustainability, decentralisation or brokerage, and to what extent they integrate programmable 5G/6G networking capabilities as Table 1 depicts.

From a pure continuum perspective, all six projects provide relevant building blocks. aeriOS targets a unified MetaOS for the IoT–edge–cloud continuum, building a virtualised, platform-agnostic layer above heterogeneous Infrastructure Elements (IEs). NEMO focuses on an AIoT–edge–cloud continuum with strong emphasis on AI-native support and secure execution environments. NEPHELE emphasises orchestration of hyper-distributed applications over programmable cloud–edge–Internet of Things (IoT) infrastructures. ICOS concentrates on volatility, heterogeneity and energy-aware operation across the continuum. FLUIDOS promotes a decentralised and flexible MetaOS to exploit idle edge capacity through distributed resource sharing. NebulOuS adopts a brokerage-oriented approach to manage fog and multi-cloud continuums.

While all can be categorised as “continuum projects”, their design drivers range from AI/ML integration and energy efficiency to decentralised orchestration and multi-cloud brokerage. Importantly, only some explicitly address tight coupling with 5G/6G-grade networking environments and telco infrastructures.

When refining the comparison specifically in terms of 5G/6G relevance, clearer differentiation emerges. aeriOS explicitly frames its work in the context of the 6G era and aims to exploit 5G-native capabilities such as NEF/CAPIF exposure, programmable network fabrics, SDN/NFV and Time-Sensitive Networking (TSN) to support latency-critical and time-sensitive applications across the continuum. NEMO provides strong AIoT-oriented orchestration with secure execution and TSN-based clustering, but with a more compute- and AI-centric orientation rather than deep telco/RAN integration. NEPHELE is strongly 5G-ready at orchestration level, focusing on programmable infrastructures, yet is not explicitly positioned as a 6G-native MetaOS. ICOS, FLUIDOS and NebulOuS contribute significantly to sustainability, security and brokerage dimensions, but treat networking largely as a generic connectivity layer rather than as an exposed, programmable substrate tightly integrated with RAN/core components.

This distinction becomes critical when viewed against the objectives of SNS JU and the 6G vision it promotes: a cloud-native networking paradigm with tightly coupled compute and communication fabrics across the cloud–edge continuum. In this context, UNITY-6G defines a particularly demanding set of requirements. UNITY-6G aims at an AI-native, scalable 6G architecture unifying terrestrial and non-terrestrial networks (TN/NTN), Open RAN, xHaul, Non-Public Networks (NPNs), edge, core and cloud. It further integrates digital twins, DLT and semantic communications to enable trustworthy, state-aware and sustainable 6G services, targeting use cases such as disaster-resilient connectivity, real-time XR/holography and time-sensitive Industry 4.0 NPN deployments.

Project	Years	Programme / Call	Continuum scope	Project website
aerOS	2022-2025	Horizon Europe, HORIZON-CL4-2021-DATA-01-05 (Meta Operating Systems	IoT–edge–cloud continuum with unified meta-OS layer over heterogeneous Infrastructure Elements	https://aeros-project.eu/
NEMO	2022-2025	Horizon Europe, HORIZON-CL4-2021-DATA-01-05	AIoT–edge–cloud continuum with emphasis in AI-native support & secure execution environments	https://cordis.europa.eu/project/id/101070118
NEPHELE	2022-2025	Horizon Europe, HORIZON-CL4-2021-DATA-01-05	Programmable cloud-edge-IoT continuum for hyper-distributed services & end-to-end orchestration	https://nephele-project.eu/

ICOS	2022-2025	Horizon Europe, HORIZON-CL4-2021-DATA-01-05	Cloud-edge-IoT continuum focusing device volatility, heterogeneity & energy-aware execution	https://cordis.europa.eu/project/id/101070177
FLUIDOS	2022-2025	Horizon Europe, HORIZON-CL4-2021-DATA-01-05	Decentralised cloud-to-edge continuum, utilization of idle edge capacity through flexible, distributed Meta-OS	https://cordis.europa.eu/project/id/101070473
NebulOuS	2022-2025	Horizon Europe, HORIZON-CL4-2021-DATA-01-05	Cloud/fog continuum through brokerage-oriented meta-platform for multi-cloud & fog resources.	https://nebulouscloud.eu/

Table 1. Continuum frameworks comparison

A continuum MetaOS suitable for UNITY-6G must therefore provide:

1. A unified resource model spanning IoT, edge, core and cloud
2. Tight interaction with programmable 5G/6G network functions and exposure interfaces
3. Support for AI-driven orchestration and closed-loop control
4. Observability and policy hooks enabling digital twin integration, sustainability optimisation and trust enforcement.

In this context, Eclipse aeriOS the open-source industrialisation of aerOS aligns particularly well with UNITY-6G requirements as Table 2 depicts. aeriOS provides a virtualised, platform-agnostic MetaOS spanning IoT–edge–cloud infrastructures, integrating smart networking capabilities such as 5G-native exposure (NEF/CAPIF), programmable network fabrics, SDN/NFV and TSN. Compared with other MetaOS initiatives, aeriOS is the most explicitly 5G/6G-aware, offering built-in hooks towards programmable telco infrastructures rather than abstracting the network as a generic transport layer.

Project	**5G/6G Networking**	**TSN /SDN-NFV**	**Telco/RAN Integration**	**6G/SNS Goals**
aeriOS	Explicitly mentions 6G era and 5G-native capabilities for secure, scalable services	High: uses 5G NEF/CAPIF, programmable fabric, SDN/NFV, TSN for latency-critical continuum apps	High: designed to cooperate with 5G infra and telco edge, suitable for network-centric use cases	High: referenced in EU MetaOS material and co-organises Meta-OS/6G events with SAFE-6G, positioning it as a reference Meta-OS for 6G-ready architectures
NEMO	Indirect: AIoT and secure meta-OS focus rather than explicit 5G/6G APIs	Yes: mainly via SLO-driven orchestration, secure execution and TSN-based clusters	Medium: more compute/AI-centric than RAN/core-centric	Medium-high: aligns with AI-native 6G vision but is less telco-oriented than aerOS
NEPHELE		Yes:	Medium-high: orchestration-level	Medium-high:

	No “6G” label, but strong 5G and programmable infra focus	5G-ready orchestration over programmable network/compute, less detail on TSN/NEF/CAPIF	integration with 5G networks	good 6G-ready orchestration substrate, not framed as 6G-native meta-OS
ICOS	No specific 5G/6G focus	Limited: mainly virtualization and energy/resource efficiency	Low: generic cloud-edge-IoT	Medium: relevant for sustainable 6G, but not network-native
FLUIDOS	No explicit 5G/6G orientation	Mainly security-oriented (zero-trust, anomaly detection, deception)	Low: generic continuum resources	Medium: complements 6G security aspects, not a 6G-native meta-OS
NebulOuS	No explicit 5G/6G branding	Focus on brokerage and SLA/SLO orchestration, not low-level networking	Low-medium: multi-cloud/fog brokerage over networks	Medium: useful for 6G-era multi-cloud/fog service mgmt, but less suitable as core 6G meta-OS than aerOS/aeriOS

Table 2. Continuum frameworks comparison based on 6G ecosystem

At its architectural core, aeriOS introduces Infrastructure Elements (IEs) and domains. IEs represent the fundamental computing units, Kubernetes workers, VMs, bare-metal servers or capable IoT gateways, hosting workloads [8]. A domain groups multiple IEs under a common administrative and orchestration context, sharing MetaOS services such as orchestration, monitoring and security. Multiple domains are federated through a continuum fabric mediated by an NGSI-LD context broker (e.g., Orion-LD), enabling peer-to-peer collaboration and cross-domain orchestration. This federated approach allows UNITY-6G partners to operate autonomous local domains while participating in a unified continuum where services, AI components and network functions can be placed and migrated according to latency, energy or resilience policies.

UNITY-6G Objective	aeriOS Support
AI-native, scalable 6G architecture (AI/ML-driven control and services across the network)	aeriOS provides a meta-OS with unified abstractions and orchestration APIs across IoT–edge–cloud, suitable to host AI pipelines (functions, microservices) close to data and integrate with AI-driven orchestration/closed loops in UNITY-6G.
Energy efficiency and sustainability for integrated TN/NTN, edge, core, cloud	Continuum-aware placement and scaling in aeriOS/meta-OS can minimise data movement, exploit edge resources and allow UNITY-6G optimisers to enforce energy-aware policies over a unified resource view.
Unified TN/NTN, Open RAN, xHaul, NPN, edge–core–cloud integration	aeriOS offers an abstraction layer over heterogeneous Infrastructure Elements and domains; 6G components (O-RAN xApps/rApps, NTN gateways, edge cores) can be modelled as IEs and orchestrated consistently, giving UNITY-6G a single continuum control plane to bind network and compute.
Trust, security and data integrity via DLT and semantic communications	aeriOS can host DLT/semantic components as services at edge/cloud, while its meta-OS view (context broker, observability, policy framework) gives UNITY-6G a place to plug trust/semantics into the lifecycle of services and network functions.

Real-time, state-aware 6G network with digital twins	With monitoring/observability and a continuum-wide resource model, aeriOS can feed digital twins with near real-time metrics from all layers (edge nodes, RAN controllers, cores) and support co-deployment of twin models and control functions.
Support for time-sensitive XR/holography and Industry 4.0	aerOS/aeriOS explicitly consider TSN and time-sensitive networking plus smart networking (5G exposure, programmable fabric), which can be exploited by UNITY-6G to guarantee E2E latency/throughput constraints for XR and TSN-aware industrial NPNs

Table 3. aeriOS continuum framework support on UNNITY-6G

Within each domain, aeriOS implements a two-layer orchestration model separating global decision-making from local enforcement. A High-Level Orchestrator (HLO), implemented in Python, provides a global resource and policy view, while Low-Level Orchestrators (LLOs), implemented as Kubernetes operators, enforce decisions at IE level. The system thus appears logically centralised but operates in a distributed and federated manner, an approach well aligned with UNITY-6G's need to interconnect heterogeneous partner testbeds while avoiding single points of failure.

aeriOS further emphasises runtime heterogeneity and extensibility. It operates over Kubernetes-native environments, Docker-based clusters and can evolve towards additional runtimes such as containerd or WebAssembly. Communication and federation rely on widely adopted standards (e.g., REST/OpenAPI, NGSI-LD, VPN overlays such as WireGuard), simplifying integration with 5G/6G exposure mechanisms and external analytics or orchestration components.

From a functional perspective, aeriOS exposes:

- Orchestration and resource APIs, providing unified, location-independent workload descriptions and IE capability queries;
- Monitoring and observability APIs, leveraging smart data models and telemetry (e.g. Prometheus exporters) to support KPI/KVI tracking, anomaly detection and energy-aware optimisation;

- Serverless/function services, enabling on-demand execution of analytics, ML inference or control logic across the continuum.

Trust and security are integral to the platform. aeriOS incorporates identity and access management (e.g. Keycloak, API gateways), encrypted communications, fine-grained access control, trust agents to assess IE reliability and enforce trustworthy resource selection. These mechanisms directly support UNITY-6G's trust, DLT and explainability objectives, providing a secure and auditable substrate for AI-driven orchestration decisions.

Mapping UNITY-6G objectives to aeriOS capabilities demonstrates a high degree of alignment as Table 3 depicts:

- AI-native scalable architecture: AI pipelines and closed-loop controllers can be deployed as microservices or serverless functions across domains, leveraging a unified resource abstraction and continuum-wide orchestration APIs.

- Energy efficiency and sustainability: Continuum-aware placement and scaling minimise unnecessary data movement and exploit edge proximity, enabling energy-aware optimisation across TN/NTN, edge, core and cloud.

- TN/NTN, Open RAN and xHaul convergence: Heterogeneous components (e.g. RAN controllers, NTN gateways, edge cores) can be modelled as IEs under a unified MetaOS control plane, binding network and compute resources coherently.

- Trust, DLT and semantic communications: Trust agents, policy frameworks and context management services provide hooks for integrating DLT and semantic models into service lifecycle management.

- Digital twin–driven, state-aware operation: Observability across IEs supplies near-real-time metrics to feed network digital twins and co-deployed control functions.

- Support for time-sensitive XR and Industry 4.0: Built-in consideration of TSN and programmable networking enables enforcement of end-to-end latency and throughput constraints required by XR and industrial NPN scenarios.

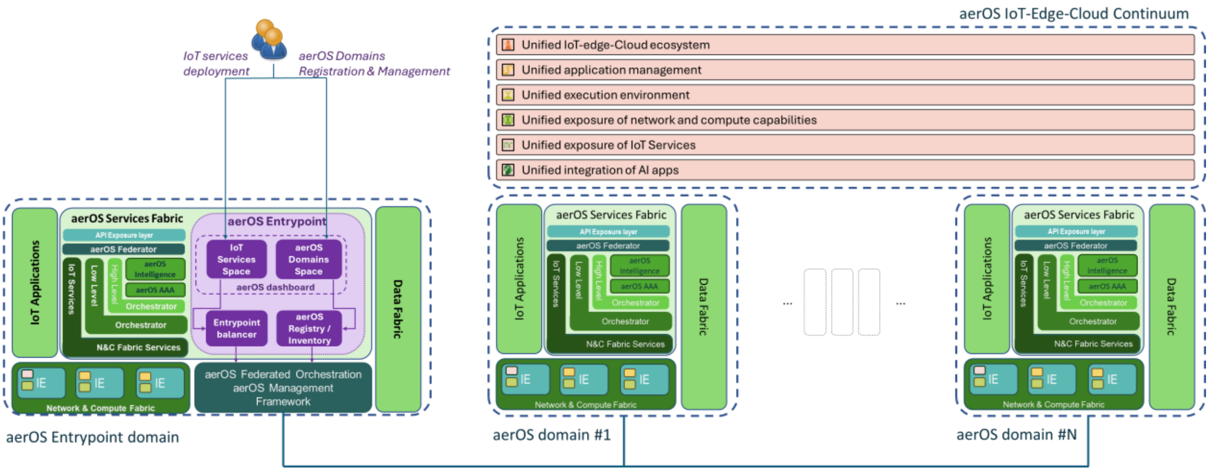


Figure 2-1. aeriOS generic IoT-Edge-Cloud Continuum

In summary, when comparing Horizon Europe MetaOS initiatives along both continuum and 5G/6G dimensions, aeriOS and its open-source evolution Eclipse aeriOS emerge as the most suitable continuum substrate for UNITY-6G. They combine continuum-native abstractions, explicit integration with programmable 5G/6G networking capabilities, federated cross-domain orchestration and integrated trust mechanisms, as figure 21 depicts. By adopting aeriOS as the underpinning MetaOS layer, UNITY-6G can concentrate innovation efforts on advanced 6G networking, AI-driven control and service intelligence, while building upon a proven, EU-driven and SNS-aligned MetaOS foundation.

2.3 3GPP

3GPP has initiated 6G studies with the two goals: simplification and flexibility, while preserving a pragmatic migration path from 5G. The envisioned evolution is data- and AI-centric, targeting new markets such as private networks, end-to-end automation, richer exposure of network capabilities, and the expected growth of AI traffic and AI-assisted operations [3].

The baseline direction is to evolve, not replace, the 5G cloud-native Service-Based Architecture. The microservice model, service-based interfaces, and CI/CD-friendly operations remain foundational, with a reinforced separation of control and user planes. In the user plane, there is emphasis on more precise and programmable flow handling, with clearer roles for N4/PFCP and fine-grained QoS enforcement. For continuity, key interfaces such as N1, N2, N3, N4, and the use of GPRS Tunnelling Protocol – User plane (GTP-U) are retained to ensure mobility and roaming work seamlessly during migration; any Non-Access Stratum (NAS) evolution is being approached cautiously to avoid unnecessary disruption [3] [9].

In the core network, modularity and optionality are central to accommodating diverse deployments, especially in private and localized networks. Proposals include making certain classical functions optional or decomposable. A frequently discussed example is splitting the AMF into an Access Management Function and a distinct Mobility Management Function [10], allowing mobility handling to be localized or omitted where stationary use cases make it unnecessary, while retaining the ability to provide a global mobility view when needed [3].

The user plane is trending toward deeper programmability across access, transport, and core. Concepts such as distinct Access-Network and Core-Network Data Plane Functions, coordinated via SDN principles, aim to deliver flow-level QoS, traffic steering, and deterministic behavior. While some scenarios may forgo GTP where mobility is not required, the consensus is that GTP-U remains the realistic default for mobility; attempts to rely purely on native IP mobility tend to reintroduce comparable complexity in a different form [3] [9].

Some contributions propose more detailed architectures to inform and improve the 3GPP 6G specifications for core networks such as the 3GPP 3GPP-6GWS-250014 [11] . For instance, as illustrated in Figure 2-2, Paper [10] presents a backward-compatible proposal for the 6G core network architecture. The design is guided by two goals: (i) decoupling mobility management from access control currently combined in the AMF while preserving core 5G functions and the existing NAS and GTP to minimize impact on UEs and the RAN; and (ii) adopting an agentic, intent-based approach by overlaying an AI-agent framework with a dedicated AI bearer to enhance core network flexibility.

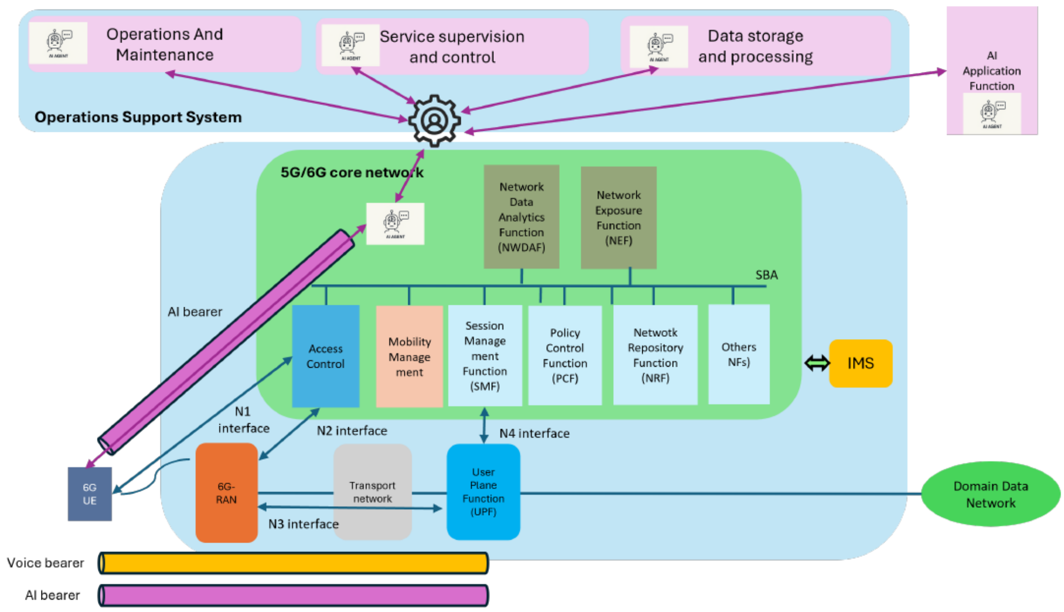


Figure 2-2 Overall architecture of the mobile core network

Co-funded by the European Union

On the RAN–core relationship, current thinking keeps the RAN outside the core SBA to prevent overloading service registries and to preserve the established N2/N3 coupling. O-RAN (including RIC and SMO) continues its own timeline, with coordination across RAN, core, transport, and cloud achieved through management systems and structured data exchange rather than by registering gNBs as SBA services.

The architecture becomes more data-centric by expanding observability beyond core signaling into RAN, transport, and cloud infrastructure. This is accompanied by a move toward coherent data frameworks, evolving NWDAF, leveraging functions such as DCCF, and adopting cross-domain data lakes, with consistent governance. Exposure is expected to mature into "intelligent exposure," aligning with open API ecosystems (e.g., CAMARA) to support intent-like interactions and to surface AI-derived insights safely and consistently [3] [12].

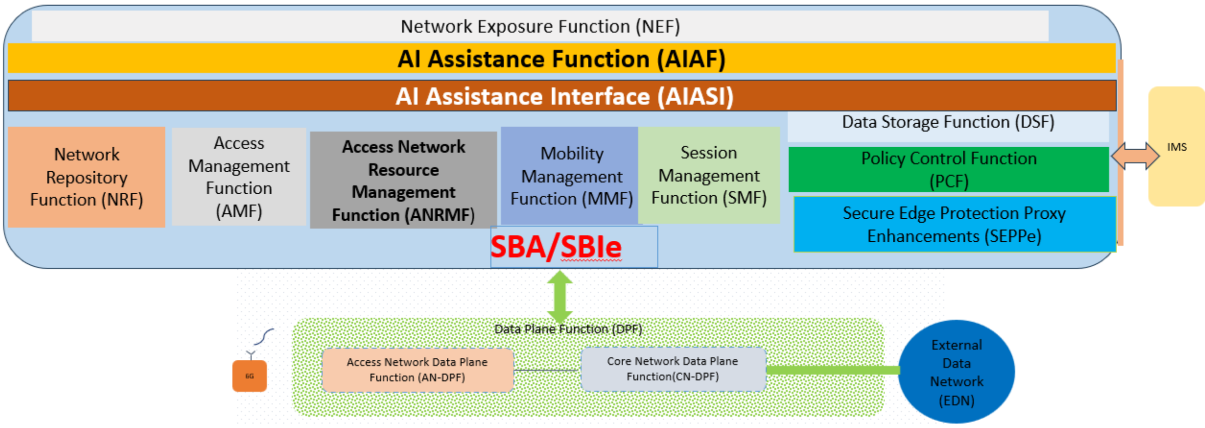


Figure 2-3 Service Based 6G Network Architecture (3GPP orange proposal)

AI is introduced as an overlay that augments, rather than replaces, existing network functions. Studies explore network AI agents and an AI Assistance Function, with an AI Assistance Interface to orchestrate multi-agent workflows spanning core, RAN/SMO, and cloud (Figure 2-2). Options discussed include a dedicated "AI bearer" with specific QoS for UE–network AI interactions, while leaving NAS and established interfaces intact. UE-side agents are recognized, and their coordination with the management plane is under study, alongside candidate inter-agent protocols developed in concert with external bodies [3].

Localization and roaming are being adapted to 6G demands. Localized deployments, such as local NRFs and a decoupling of NF logic from UE context-upport edge services and enterprise-specific use cases. Roaming and inter-PLMN exposure leverage SEPP enhancements to maintain trust and security in a more data- and AI-rich environment. In parallel, the IMS continues to evolve for voice and communications, and the core maintains a coherent multi-access posture across 3GPP and non3GPP access, including NTN, with consistent mobility and policy when required [3] [9].

Open issues under active study include architectural simplification and modularization (including SBA/SBI evolution, slicing simplification, and data-plane resource management), the AI-assisted architecture (agent roles, UE agents, and bearer options), and the data framework (cross-domain data lakes and governance). Interworking and migration paths, refined N4/PFCP semantics, scalable RAN–core data exchange, and the security and trust model for AI and agentic functions remain key topics [3].

In sum, the emerging 3GPP view favors an evolutionary 6G core: keep the 5G SBA foundation, add AI/agentic capabilities and a unified data fabric, make mobility and other features modular and optional, and strengthen user-plane programmability. In the UNITY-6G architecture that we will present in the following sections, we aimed to define multiple layers and domains to enable the modularity and the user-plane programmability including AI agents' capabilities aspects aligned with 3GPP specifications.

2.4 TMFORUM

The TMForum provides an architecture framework called Open Digital Architecture (ODA) [4] that can be applied to Telco operators and the whole ecosystem of its customers and providers. This framework offers a simplified representation around 3 systems in charge of:

- Interfaces with users, external systems (systems, IoT, ...)
- Operational processes and treatments
- Analytical processes.

TMForum's ODA aims to make network automation scalable, interoperable, and vendor-agnostic by defining a componentized, cloud-native, API-first architecture that cleanly separates concerns across engagement, records, and insight domains. It standardizes how OSS/BSS interact with orchestrators and controllers through Open APIs and model-driven catalogs, enabling intent-to-fulfillment-to-assurance closed loops. By decoupling commercial logic (products and offers) from technical realization (resources), ODA accelerates service design, provisioning, and lifecycle management across multi-vendor, multi-domain networks. It promotes event-driven operations, policy/intent governance, and integration of analytics/AI to support zero-touch operations and continuous optimization. Ultimately, ODA reduces integration complexity and OPEX, shortens time-to-market, and enables consistent automation patterns from service exposure to activation.

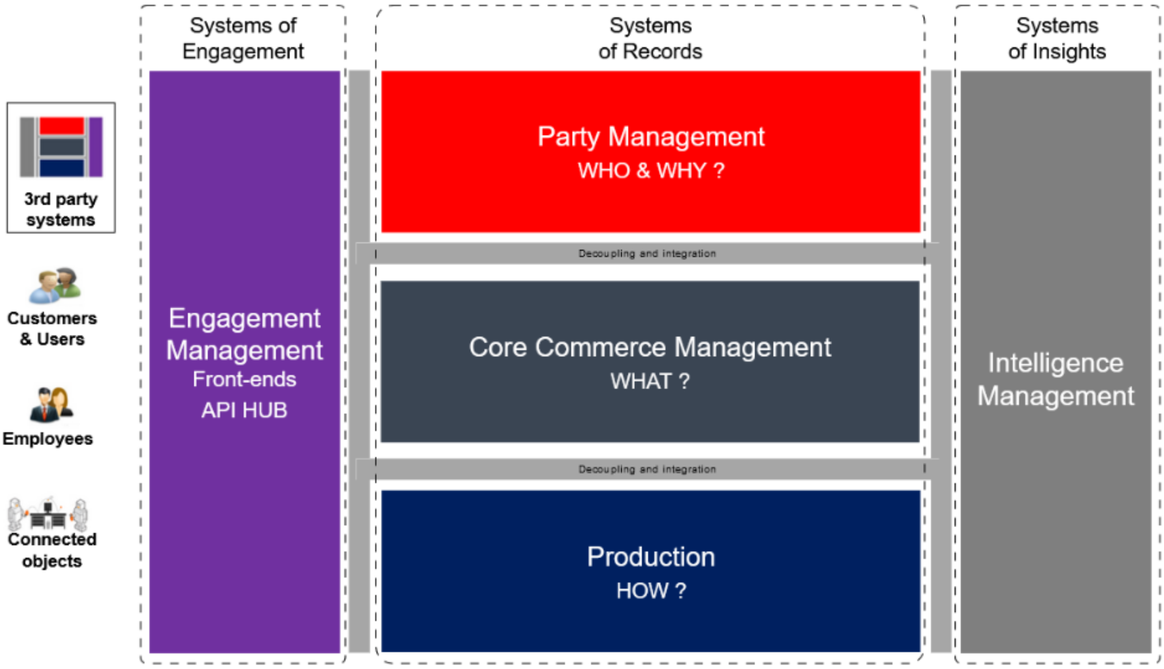


Figure 2-4. TMF ODA Overview

In the ODA model as represented in Figure 2-4 each functional block owns its data and processes and exposes them through standardized Open APIs. This modular responsibility and API-first approach allow clear boundaries, loose coupling, and consistent integration across the estate.

Within the System of Records, Party Management governs everything about “who” and “why” (customers, partners, employees; roles, intents, relationships). Core Commerce Management governs “what” (offers, products, and product specifications). Production governs “how” (services and resources) and runs the operational lifecycle, including fulfillment and assurance.

The System of Engagement is centered on Engagement Management, which orchestrates customer interactions across channels such as web UIs, chatbots, APIs, and physical stores. Regardless of the entry channel, the same underlying System of Records processes are invoked, ensuring consistency of business logic and data.

The System of Insight is anchored by Intelligence Management, which provides analytical and exploratory capabilities. It produces business activity insights, supports NBO/NBA (Next Best

Offer/Action), and leverages AI/ML. It processes information generated by services delivered to the customer organization and can enrich it with external data sources.

To decouple commercial and technical concerns within the System of Records, TM Forum defines Customer-Facing Services (CFS) as the “Know-How” that Production exposes to Core Commerce; these are technology-agnostic service abstractions. Core Commerce Management derives Product Specifications as constrained specializations of CFS and assembles Product Offerings that add commercial characteristics such as pricing, bundles, and terms. Production defines the Know-How as CFS and implements the technical realization using Resource-Facing Services (RFS) and resource elements required for delivery and assurance.

The TMForum proposes an Autonomous Networks Project (ANP) (Figure 2-5) reference architecture for the ODA architecture. The ANP is a functional blueprint that organizes the capabilities needed for autonomous network operations. It defines operational layers, partitions the autonomous network into autonomous domains, and specifies reference points for interfaces both between functions and across domains. The ANP introduces the concept of Autonomous Domain. As defined in [13], an autonomous domain is a system that is capable of autonomous behavior (e.g. resolve tasks, adhere to objectives) without manual human intervention. It is a domain with an administrative governance boundary that defines the scope of encapsulated autonomous behaviors. The autonomous domain does this by realizing self-management capabilities using:

- closed control loop mechanisms, using four key phases: awareness, analysis, decision-making, and execution (the so-called AADE closed control loop).
- Intent-driven interfaces (API or human interfaces like an UI)

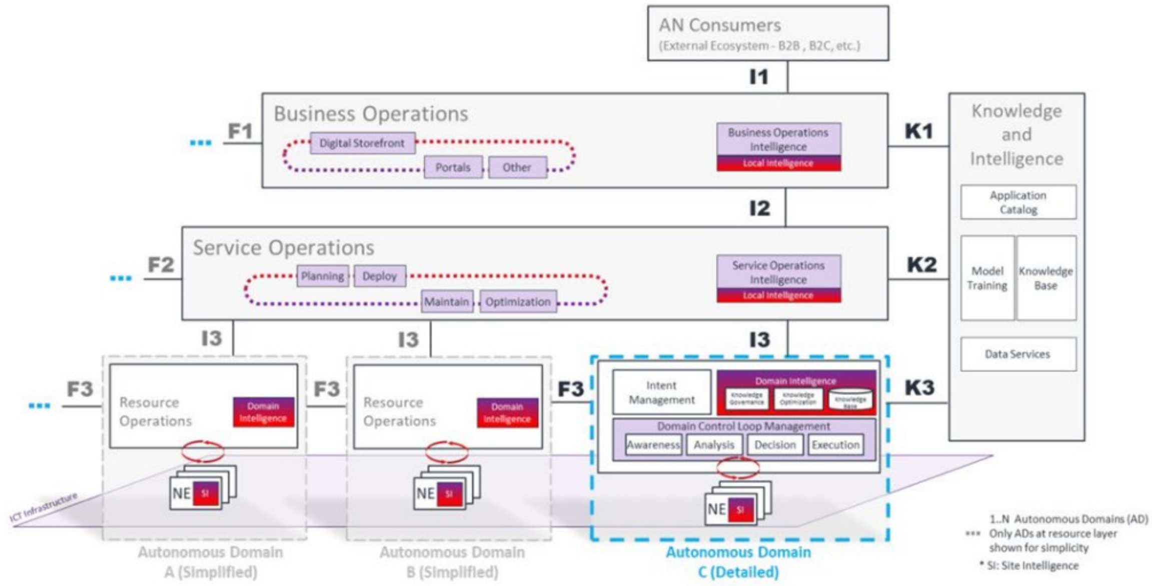


Figure 2-5 TMForum Autonomous Networks Reference Architecture [13]

In the TM Forum Autonomous Networks Reference Architecture, the Intent Management function (see Figure 2-6) accepts requirements expressed as intent via an intent interface/API and leverages measurement and analytics to observe the autonomous domain's state and behavior. Its core responsibility is to plan and execute strategies that drive the environment into compliance with the declared intent. Actions applied to the domain produce state changes that are validated through subsequent observations, thereby completing a closed loop [13].

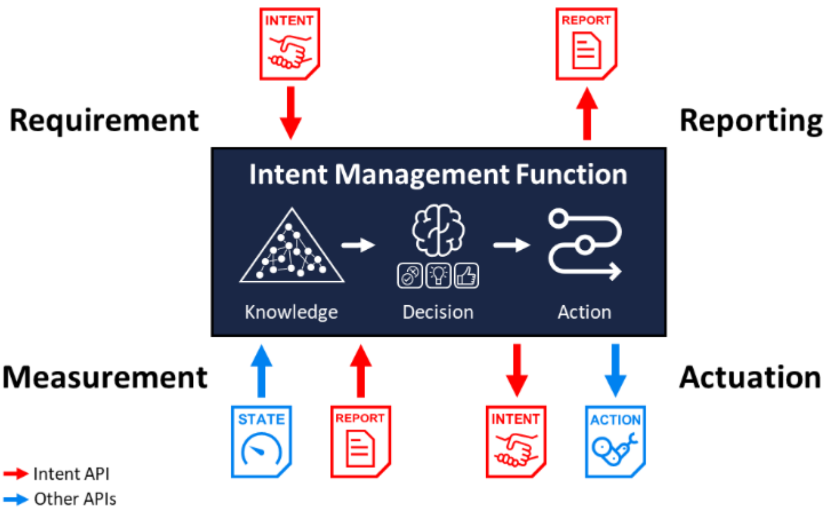


Figure 2-6 intent management function [13].

3 ARCHITECTURE OVERVIEW

This section presents an overall description of the UNITY-6G architecture. First, a high-level overview is provided, highlighting the key features that define the architecture. Next, the five architectural principles on which the architecture is based are described: the service-based approach, the AI-native approach, semantic communications, scalability and sustainability, and trust and security. Finally, the common cross-layer components are introduced, namely the Inter-Domain Management Orchestrator (IDMO), the Domain Management Orchestrator (DMO), and the Infrastructure Domain Manager (IDM).

3.1 HIGH LEVEL DESCRIPTION

UNITY-6G aims to deliver an AI-native, hierarchical, and distributed service-based architecture for fully automated (zero-touch) 6G network and service orchestration, extending beyond 5G's core-limited SBA to integrated, multi-domain environments (Open RAN, Core, Cloud/Edge, NTN, Transport, multi-RAT NPN). Its main objectives are to optimally distribute monitoring, analytics, and decision-making for scalability; embed fault tolerance, security, and energy efficiency by design; and unify lifecycle management of cloud-native applications and network functions (e.g., xApps/rApps, containerized 6G core) through open, standardized APIs across the edge-to-cloud continuum.

Leveraging advanced AI and DLT, it coordinates large numbers of parallel components and enables innovative multi-stakeholder business models via a programmable, open control plane, which accelerates agile service deployment. This approach is validated through three representative use cases: Sustainable networks for disaster handling, (ii) Immersive Experience with Real-time XR/holographic communications, (iii) Digital Twin for Integrated 6G Network Evaluation, (iv) Multi-RAT O-RAN enabled NPN for supporting time sensitive applications for Industry 4.0, described in detail in D2.1.

To answer the objectives described above, and, in the UNITY-6G proposal, we defined in this deliverable a global UNITY-6G architecture as illustrated in Figure 3-1. The proposed architecture is autonomous and enables more modularity and programmability thanks to the following aspects:

- **Multi-layered**: The proposed UNITY-6G architecture is composed of five layers (exposure, management and orchestration, infrastructure, data continuum and trust layer).

- **Multi-domain**: to enable the 6G continuum, we defined 7 domains (i.e., Core network, ORAN, NTN, Transport, Non-3GPP RAN, AI/ML subsystem and vertical APP domain). These domains will be described further in Section 4.2. These domains are represented in the unification layer and are connected and managed through a cross-domain service management bus in the management and orchestration layer.

- **AI native**: to manage the UNITY-6G AI agents we defined the AI/ML subsystem domain that exposes its capabilities to the other management domains (e.g., training, models, enabling AI agents for multiple tasks) so it can be consulted by the other domains through the service bus.

- **Distributed service-based architecture**: the main idea behind the UNITY-6G architecture is to propose a distributed service-based architecture with multiple domains. Each domain represents a service such as the Core domain or the NTN domain. Each domain has all the elements to enable its management and scalability.

- **Inter Domain Management and Orchestration (IDMO)**: the different domains are managed with IDMO in the management and orchestration layer.

- **Energy and security aspects consideration**: in UNITY-6G, we also consider the integration of security aspects and energy optimization 'by design' in the architecture. With the definition of a trust layer and the integration of the virtual power plant (VPP) for renewable energy consideration.

- **Co-existence with other architectures**: the coexistence with other architectures is enabled with the exposure layer. The exposure layer is detailed in the UNITY-6G architecture presented in Figure 3-1. The exposition of APIs using CAMARA enables an external exposition with other projects architectures such as AMAZING-6G and 6G-VERSUS.

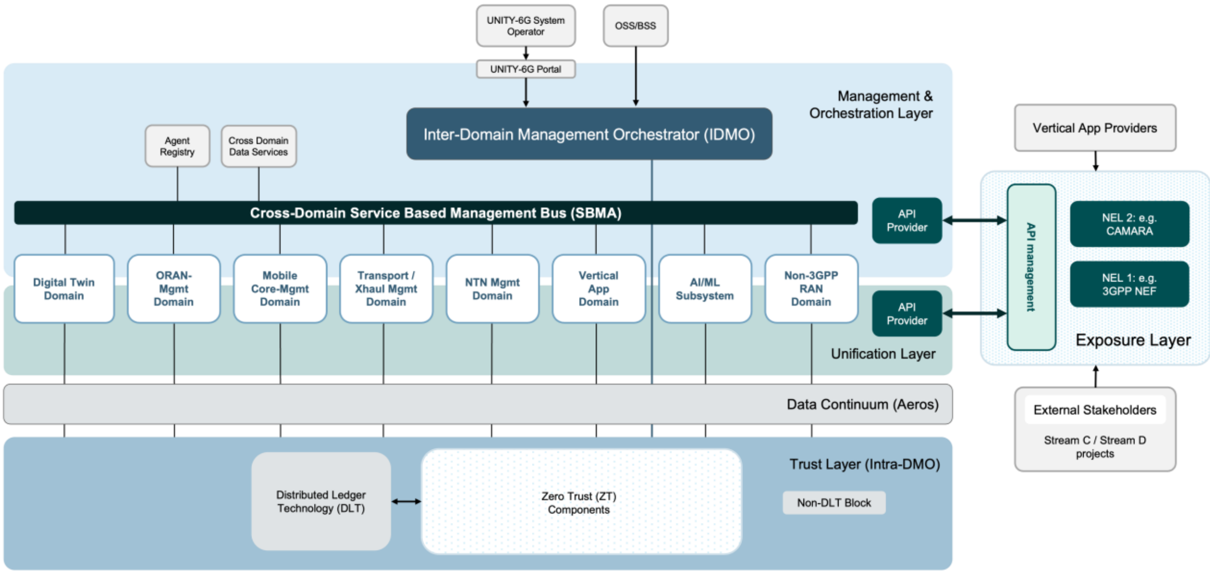


Figure 3-1. UNITY-6G General Architecture Overview

3.2 ARCHITECTURAL PRINCIPLES

3.2.1 Service-based approach

In the UNITY-6G architecture a service is defined as a capability delivered (offered) over the communication infrastructure that provides functionality to users or other services by any domain (O-RAN, NTN, etc.). It is characterized by having defined KPIs for monitoring, requirement (different services, latency, computational power, etc.) and can be optionally scaled to meet demand. A service acts as a sperate unit but can combine multiple other services; it can be to a single domain or span across multiple domains. Services are controlled (registered, discovered, created, updated, deleted, etc.) through service bus. Each service must adhere to strict Service Development Lifecycle (SDLC) enabling rules for activation, suspension or termination.

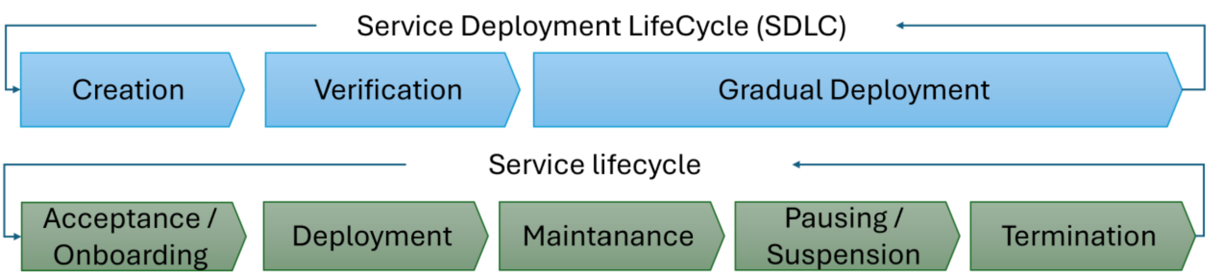


Figure 3-2. Service Development and Servie Lifecycles

Two lifecycles of a service overlaps and are important in the terms of unified network operation. In Figure 3-2 both lifecycles with their stages are visible. SDLC is similar to software development lifecycle, but focuses more on resource allocation, cross-domain integration and end-to-end service control (not only software artifacts that can be used individually). In UNITY-6G a proposed SDLC is divided into three stages:

a) Creation (Development team) – involves planning, analysis and implementation. At the end of this stage service should be in form of package with description, set of requirements and containerized application/service.
b) Verification – firstly the output of the creation stage should be verified in terms of requirements, dependencies, connections, size, etc. It is the place for quality gate check, test coverage analysis and security vulnerability scans. Secondly, when previous steps are completed a service can be tested in virtual environment (digital twin) using some sort of AI-based network selector (that determines which parts of networks, e.g. domains, are needed).
c) Gradual deployment – lastly, using, e.g. canary deployment, the service is deployed to limited number of nodes to monitor stability. If successful, then deployment expands gradually.

On the other hand, a service follows a service lifecycle. It is interlaced with SDLC. In this case a service lifecycle is divided into another five stages:

a) Acceptance / onboarding – to ensure if service is understandable and dependencies are defined. Service description should be correct (contains required fields, e.g. localization, available time, latency range), and fit registry pool.
b) Deployment – all dependent services must be deployed, running and healthy. Requirements for CPU, memory, power must be met at the moment of deployment. There must be no conflicts between services.
c) Maintenance (running) – the deployed service and its dependencies require constant health and KPIs monitoring.
d) Pausing / suspension – a service may be paused, e.g. if there are no users or requests for defined time, to save resources. A service can be suspended upon health check failures or critical dependencies fail.
e) Termination – a service can be terminated when requirements are no longer met (e.g. latency), is no longer needed (was created for a specific time period) or a service is replaced by an updated or alternative service.

3.2.2 AI-Native approach

In this architecture, AI is treated as a native capability rather than an add-on. Management and control functionality is realized through Agents that continuously observe parts of the system, analyse what they see, decide how to react, and, when authorized, carry out those reactions on the underlying infrastructure. To keep this uniform across all domains and implementations, each Agent is described using a common logical pattern based on the MS–AE–DE–ACT tuple, where MS refers to the Monitoring System, AE to the Analytics Engine, DE to the Decision Engine, and ACT to the Actuator.An Agent does not have to implement all four capabilities. Some Agents are purely observational (MS only), some perform analytics without taking direct actions (MS+AE), others only translate decisions from elsewhere into concrete changes (ACT only), and many combine multiple capabilities (for example, MS+AE+DE with actuation delegated to a shared domain controller). The tuple is therefore a descriptive model, not a rigid implementation requirement: it allows the architecture to describe what an Agent could do and where responsibility lies, even if some parts are implemented by other Agents or shared services.

An Agent is a logically autonomous control entity that can:

- Observes a subset of the system state (within one or multiple architectural domains).
- Produces analytics and predictions over that state.
- Takes decisions according to local and global policies.
- Triggers actions on the underlying infrastructure or services.

Agents are defined along two orthogonal axes:

- **Conceptual placement** – which domain(s) they belong to and are responsible for (e.g. RAN, transport/xHaul, core, AI/ML subsystem, time-sensitive networking, vertical application, trust/security, management/orchestration). This determines what information they are allowed to consume, and which resources they are allowed to influence.
- **Physical placement** – on which hardware tier they actually run (e.g., inside a base station, at a local edge rack, in a regional data center, or in a central cloud cluster, including distributed or federated deployments). Different capabilities of the same Agent may be placed on different tiers; for instance, monitoring and actuation close to devices, with analytics and decision logic in a regional or central node.

All inter-agent and inter-domain coordination (telemetry publication, analytics/insight sharing, intent exchange, conflict coordination, and feedback reporting) can be performed via the Service-based management bus (SMB) APIs/topics/events/intent interface. Actuation ultimately executes through domain-specific southbound interfaces (e.g., O1/NETCONF/gRPC), typically mediated by domain controllers; ACT exposes the resulting commands, status, and actuation feedback back onto the SMB. This inter-agent and inter-domain coordination applies both "north-south" (towards orchestrators, data management, digital twin, trust functions) and "east-west" (between Agents themselves). Direct, opaque coupling is avoided; an Agent is visible in the architecture by the services it exposes and consumes, and by its tuple descriptor.

The SMB is the architectural coordination plane through which Agents and domains expose and consume service-based interfaces (APIs, topics, events, and intent interfaces). It enables discovery, composition, and controlled interaction across heterogeneous domains (e.g., transport, RAN, NTN, digital twin), and supports standardized exposure patterns (e.g., capability exposure and API onboarding mechanisms).

In addition to the SMB, the architecture assumes a Data Continuum: a cross-domain, governed substrate for persistent data products spanning real-time streams and historical datasets. The Data Continuum supports materialization of selected SMB streams/events into reusable data products (time-series, logs, topology/config snapshots, DT snapshots, derived features, model artifacts), including cataloguing, lineage, access control, replay, and audit. This allows Agents to combine online coordination via SMB with long-horizon reuse and learning via the Data Continuum.

Arrows in Figure 3-3 depict logical information/control flows; depending on deployment, these may be realized as local calls (co-located tuple parts) or SMB-mediated services (distributed/federated tuple parts), while externally visible contracts remain SMB-exposed.

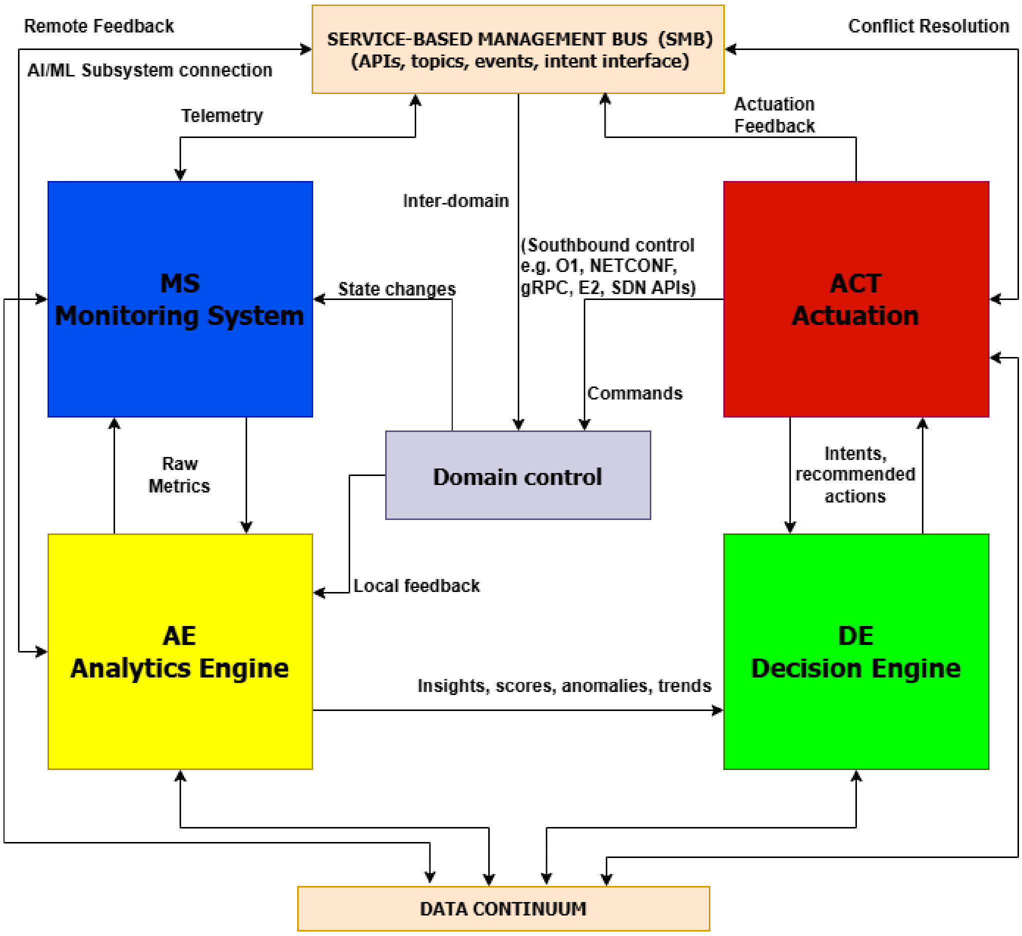


Figure 3-3. Tuple interconnectivity

3.2.2.1 Tuple definitions

3.2.2.1.1 Monitoring System

The **Monitoring System** capability provides the observation function of the AI-native loop. It acquires measurements, counters, events and state information from the infrastructure and service components that are relevant for the Agent's conceptual domain, normalizes and timestamps them, and enriches them with identifiers such as node, link, slice, service, user or application, as appropriate.

Conceptually, an MS belongs to the domain whose behaviour it describes (for example, a transport MS focused on xHaul links, or a RAN MS focused on radio and scheduler metrics). Physically, MS components may reside directly on the devices they observe, on nearby edge platforms, or in centralized collectors, depending on latency, overhead and resource considerations.

MS components may apply lightweight processing such as aggregation, filtering or semantic enrichment, but they do not perform complex analysis or control. Their role is to provide a consistent and trustworthy view of the system state to downstream analytics and decision functions, and to expose this view via the service-based management bus.

3.2.2.1.2 Analytics Engine

The **Analytics Engine** capability transforms observations into information that can guide decisions. It consumes telemetry from one or more MS components and, where needed, additional context such as topology, configuration, inventory, historical records, digital-twin outputs or external data sources. Based on this information, AE functions derive higher-level insights, including but not limited to anomaly indicators, correlations, classifications, forecasts, risk scores and performance indicators.

AE components may use a range of methods, from simple thresholding and rules, through statistical and optimization techniques, to advanced AI/ML and reinforcement learning models. The training, validation and lifecycle management of such models is coordinated with the AI/ML subsystem and data continuum, but from the architectural viewpoint an AE is described by the types of insights it produces, the scope of those insights (which nodes, links, slices, services, users they refer to), and their quality attributes (latency, confidence, expected accuracy).

Conceptually, an AE is associated with the domain whose behaviour it analyses (e.g., transport analytics, RAN analytics, core analytics, cross-domain analytics). Physically, AEs are typically deployed at edge, regional, or central tiers where sufficient computational capacity and data access exist. In some cases, a federated AE can combine partial analytics produced at different tiers.

3.2.2.1.3 Decision Engine

The **Decision Engine** capability maps analytics and policies into **intents** and **recommended actions**. It interprets insights generated by AEs in the context of objectives such as performance, latency, reliability, energy efficiency, cost, security, or regulatory compliance, and applies policies and constraints defined by the management and trust layers.

Each DE operates over a clearly defined **decision space**, specifying which parameters, resources, and abstractions it is allowed to influence (for example, routing and QoS parameters in a transport domain, radio configuration and scheduling policies in a RAN domain, placement and scaling of network functions in a core domain, or admission and configuration of application-level services in a vertical domain). DE functions may be implemented using rule-based logic, optimization algorithms, or learned policies; what is common is that they should produce machine-interpretable decisions that can be audited, traced, and, if necessary, reconciled with decisions taken by other Agents.

Conceptually, a DE is deployed in the domain in which it has authority to decide. Physically, DE components are deployed where the trade-off between reaction time and global visibility is appropriate: closer to the edge for fast, local loops, and at higher tiers for slower, cross-domain or strategic decisions. DE outputs are conveyed over the management bus to the corresponding actuators and, where applicable, to coordination entities responsible for conflict management and prioritization.

3.2.2.1.4 Actuator

The **Actuator** capability executes decisions on the managed system. It receives intents and recommended actions from DE functions and translates them into concrete operations on network and service elements, using the appropriate southbound management and control interfaces exposed by domain controllers, orchestrators, or directly by network functions and devices.

An ACT instance is conceptually linked to the domain it controls, such as RAN, transport, core, vertical application, or trust/security. In terms of physical placement, ACT components are typically located close to the controlled resources: as part of a domain controller, within a management function for a given technology, or on local control entities attached to the infrastructure.

The responsibilities of ACT functions include validating requested actions against applicable policies and safety constraints, enforcing rate limits and guardrails, executing the changes, verifying their effect, and reporting execution status back to MS, AE and DE components. This feedback closes the control loop and enables monitoring and learning of the impact of AI-driven decisions.

3.2.2.2 Stakeholders, ownership, and tuple description in the architecture

In a multi-stakeholder 6G environment, each Agent and its MS–AE–DE–ACT tuple is associated with specific stakeholder roles and ownership domains. These include, for example, infrastructure operators or network operators responsible for the underlying connectivity, service providers offering logical or virtualized services on top, vertical tenants consuming tailored network and application services, and end users or devices at the edge.

Ownership determines which part of the system a tuple may observe and control. Tuples associated with an infrastructure or network operator typically own monitoring and actuation over RAN, transport, core, and other infrastructure domains, and their decision and analytics capabilities operate within the corresponding resource and policy boundaries. Tuples associated with service providers or vertical tenants focus on service-level analytics and decisions, and request changes through the exposed service-based interfaces rather than directly modifying infrastructure elements. Lightweight tuples associated with users or devices may provide local monitoring or analytics and interact with operator or service-level Agents under well-defined access and policy constraints.

Within the architectural description, each Agent is represented by a tuple descriptor that identifies its conceptual domain(s), physical deployment options, functional role, stakeholder ownership, and permitted scope of observation and control, together with the MS, AE, DE, and ACT capabilities it implements or relies upon. These descriptors are used in the models and diagrams of the architecture to show how intelligence is distributed: which Agents exist in each domain and at each tier, which stakeholder they correspond to, and how they connect via the service-based management bus. This explicit description of tuples in terms of domain and stakeholder ownership enables the AI-native architecture to support shared infrastructure, multi-tenant operation, and end-to-end optimization while maintaining clear separation of responsibilities and control boundaries.

3.2.2.3 Tuple descriptors

Each Agent can be represented in the architecture by a tuple descriptor. The descriptor is the authoritative record that could include information of what the Agent is, where it sits conceptually and physically, what parts of the MS–AE–DE–ACT tuple it implements, and under which conditions it may observe or influence the system.

A tuple descriptor contains at least the following logical elements:

- **Identity and role**: a unique identifier, a human-readable name, and a short functional description (e.g., “Transport congestion optimization agent”). The descriptor specifies whether the Agent operates mainly as a monitoring, analytics, decision, actuation, or combined entity.
- **Conceptual domain**: the primary domain to which the Agent belongs (e.g. RAN, transport/xHaul, core, AI/ML subsystem, time-sensitive networking, vertical application, trust/security, management/orchestration). Secondary domains may be listed when the Agent has a cross-domain role.

- **Physical deployment profile**: the hardware tier(s) at which the Agent, or each of its capabilities, may be instantiated, such as device/base-station, edge/on-prem rack, regional data center, central cloud, or distributed/federated over multiple tiers. The descriptor indicates, for example, that monitoring and actuation may be local at each base station RU, while analytics and decisions may be centralized (e.g., cloud, clustered RICs, etc)

- **Tuple capabilities**: the MS, AE, DE, and ACT capabilities that are provided by the Agent itself, and those it depends on from other Agents or shared services. For each capability, the descriptor states whether it is fully implemented, partially implemented (e.g., only a subset of metrics in MS, or only advisory DE), or entirely delegated.

- **Observation and control scope**: the classes of objects over which the Agent may observe and/or act (nodes, links, slices, flows, functions, applications, users, devices), including the relevant identifiers or label spaces. For example, a transport DE may be scoped to specific domains or slice classes and never operate on RAN parameters.

- **Stakeholder ownership**: the stakeholder role on whose behalf the Agent operates (e.g., infrastructure operator, service provider, vertical tenant, end user/device) and the corresponding ownership domain. This determines the boundaries of permitted observation and control.

- **Operational profile**: the expected operating modes (continuously active, event-driven, scheduled, on-demand), temporal characteristics (e.g., near-real-time, non-real-time, offline), and high-level performance targets (e.g., typical decision latency class, data rates).

- **Interfaces**: the set of services and APIs the Agent exposes telemetry services from MS, insight services from AE, decision/intent services from DE, and actuation services from ACT. For each interface, the descriptor indicates the direction (provided/required), the interaction style (request–response, publish–subscribe, streaming), and the main data abstractions.

These descriptors are maintained in the system's architectural catalogue and could be referenced during IDMO/DMO orchestration, data management, and conflict management functions when deploying softwarised services to equip those services with appropriate Agents.

3.2.2.4 Interfaces and exposed APIs

All four tuple capabilities interact with each other using the close loop interconnectivity interfaces. And with the rest of the system through service-based interface exposed on the management bus or the data continuum substrate. The architecture distinguishes four broad interface classes:

1. **Observation interfaces**, used to publish and subscribe to telemetry and state (primarily MS, but also ACT feedback and AE-exported KPIs).
2. **Insight and decision interfaces,** used to exchange analytics results and decisions (primarily AE and DE).
3. **Actuation interfaces,** used to express intents and concrete actions (between DE and ACT, and between ACT and domain controllers or managed functions).
4. **Data-continuum interfaces,** used to read/write governed data products for cross-domain reuse over time (e.g., historical telemetry, logs, topology/config snapshots, DT snapshots), derived artifacts (features/embeddings), and lifecycle artifacts (trained models, evaluation reports). These interfaces include catalogue/lineage and access control properties consistent with data-space principles.

For each capability:

- The **MS** exposes one or more telemetry services. These services accept no control input beyond configuration and offer time-stamped measurements, events, and state snapshots, enriched with semantic identifiers. Consumers (typically AEs, sometimes other observability tools) subscribe using standard topics or API calls, with clear contracts on data rate, schema, and reliability. MS may additionally materialize selected telemetry/event streams into the Data Continuum for replay, training, benchmarking, and audit.

- The **AE** consumes MS telemetry and contextual data through observation interfaces and exposes one or more **insight services**. These services provide derived information such as anomaly flags, risk scores or forecasts. Downstream DE components invoke or subscribe to these AE APIs to obtain the most recent insights relevant to their decision space. AE may consume historical data metrics from the MS and model artifacts from the Data Continuum. ~~in addition to online SMB streams.~~

- The **DE** consumes AE insights and policy information through its input APIs. It exposes **decision and intent services** on the management bus. A DE advertises the types of decisions it can provide, their scope and expected frequency. ACT components subscribe to these services or are targeted via explicit intent APIs. DE may consult Data Continuum artifacts (historical outcomes, risk baselines, prior decision effectiveness) to enforce guardrails and stability.

- The **ACT** consumes decisions and intents via DE interfaces and exposes **actuation status and feedback services**. Downwards, it integrates with domain-specific management and control APIs (for example, through controllers or direct management interfaces), which are outside the management bus but referenced by ACT. Upwards and laterally, it publishes actuation results and any observed side-effects back as events or telemetry, closing the loop. ACT may log execution traces, outcome summaries, and rollback reasoning into the Data Continuum for audit and learning.

All these interfaces are described in the tuple descriptor by name, direction, and data abstraction, without constraining the concrete technology, which is specified at the implementation level in line with the general service-based approach.

3.2.2.5 Constraints

The AI-native architecture imposes a number of constraints on Agents and their tuples to ensure safe and predictable behaviour in a multi-stakeholder 6G environment.

First, **scope constraints** derive from stakeholder ownership and domain responsibility. An Agent's MS must not access data outside its declared observation scope; its AE must not infer or expose information beyond authorized abstractions; its DE must not issue decisions outside its decision space; and its ACT must not attempt to modify resources or parameters that are not within its control domain. These constraints are defined in the tuple descriptor and enforced by the underlying identity, authorization, and trust mechanisms.

Second, **performance constraints** apply to the temporal and reliability behaviour of the tuple. Each Agent is associated with one or more latency classes (for example, near-real-time loop for local optimization, non-real-time loop for planning or capacity management, offline loop for long-term learning). MS components must deliver telemetry within agreed bounds and loss rates; AE components must meet processing deadlines consistent with their consumers' needs; DE components must provide decisions before they become stale; ACT components must execute actions within acceptable time windows and report status reliably. These constraints can be expressed in the operational profile and interfaces and are taking into consideration when designing and orchestrating the close-loop.

Third, **safety and trust constraints** govern actuation. ACT components must adhere to guardrails such as maximum reconfiguration rates, permitted parameter ranges, rollback strategies, and blackout periods during which certain changes are disallowed. DE components must consider such constraints when generating decisions and must be able to signal when a requested optimization cannot be performed safely. Trust and security functions in the architecture provide the policies and enforcement mechanisms that ensure these constraints are respected.

Finally, **resource and scalability constraints** limit the resource consumption of Agents and their tuples. AE and DE components, particularly those relying on AI/ML, must operate within defined compute, memory, and storage envelopes at their target tiers, and must support scaling strategies (horizontal or vertical) consistent with the architecture's scalability and sustainability principles.

3.2.2.6 Conflict detection and resolution

Because multiple Agents may operate over overlapping resources or objectives, the architecture must support explicit conflict detection and resolution mechanisms.

Conflicts can arise when two or more DE components propose incompatible actions for the same resource (for example, two transport Agents attempting to set conflicting routing policies) or when an Agent's decisions violate higher-level policies or constraints. The tuple descriptors provide the information needed to detect potential conflicts: each DE should advertise its decision space, domain and stakeholder ownership; each ACT advertises the resources it can control.

Resolution may follow several patterns, which are not mutually exclusive:

- **Priority-based resolution**, where policies define an ordering between Agents based on stakeholder role, domain criticality, or function class. For example, operator-owned safety or trust Agents may override tenant-level optimization Agents when conflict is detected.
- **Hierarchical resolution**, where local Agents operate within bounds set by higher-level Agents or coordination functions. In this case, decisions from local DEs are either pre-filtered by a higher-level DE or subject to approval by a conflict management function before reaching ACT.

- **Negotiation-based resolution**, where DE components exchange proposals and constraints over the management bus and converge on mutually acceptable actions, typically under the supervision of a coordination function.

The architecture does not prescribe a single resolution algorithm but requires that: potential overlaps are detectable through descriptors; decisions and actions are traceable; and there is a clear, policy-driven mechanism for deciding which actions are allowed to reach ACT in case of conflict. Feedback from ACT and MS (in terms of execution status and resulting state) is used to refine both conflict management and future decision-making.

3.2.2.7 Deployment and operation modes

Agents may operate in different **deployment and operation modes**, depending on their scope, latency requirements, and available infrastructure. Depending on multiple considerations, we envisage the following modes.

In a **centralized mode**, most tuple capabilities (AE, DE, and sometimes MS) are placed in a central or regional data center, with minimal MS and ACT components at the edges. This mode offers global visibility and simplified coordination but is suitable primarily for non-real-time or slower loops, or for functions whose scope naturally spans large portions of the network.

In a **distributed mode**, multiple instances of the Agent (or of some of its capabilities) are deployed closer to the resources they manage. For example, each cluster of base stations or transport nodes may host its own MS and AE, and even its own DE, making decisions locally within its assigned scope. Distributed ACT components apply changes locally. Coordination between instances is achieved via the management bus, and conflict resolution is handled either by local rules or by a lightweight higher-level coordination function.

In a **hierarchical mode**, local Agents run near the infrastructure and handle fast, fine-grained loops, while one or more higher-level Agents operate at regional or central tiers and handle slower, coarse-grained or cross-domain optimization, similarly to what is currently proposed within O-RAN architecture. In this pattern, local MS/AE/DE/ACT tuples are constrained by policies and intents provided by higher-level DEs, and they provide aggregated insights upwards. Hierarchy is reflected both in conceptual placement (local vs global domain responsibility) and in physical deployment (edge vs central).

Hybrid arrangements are also possible, for example, centralized AE shared by several distributed DEs, or vice versa. The tuple descriptors explicitly state which deployment profiles are supported and which components are mandatory at each tier. Orchestration uses this information to place and scale Agents consistently with infrastructure capabilities and use-case needs.

3.2.2.8 End-to-end operation of the tuple

Putting these elements together, the operation of a tuple-based Agent can be summarized as follows. MS components, located conceptually within the Agent's domain and physically at one or more tiers, continuously or event-driven collect telemetry from infrastructure and services. AE components subscribe to the relevant telemetry and context, process it to derive insights, and expose these insights via their analytics APIs. DE components consume the insights and policies, reason about possible actions within their decision space, and publish decisions and intents. ACT components receive these decisions, check them against safety and policy constraints, map them onto concrete domain-specific commands, execute the commands using the appropriate southbound interfaces, and report back execution status and any observed changes onto the bus.

Throughout this process, descriptors and interfaces define what each component is allowed to see and do, constraints shape their performance and safety behaviour, conflict resolution ensures coherent control across overlapping Agents, and deployment modes determine where each part of the tuple resides. Together, these elements realize an AI-native, service-based management and control framework in which MS, AE, DE and ACT form a coherent pattern that can be reused across domains, stakeholders and use cases.

3.2.3 Semantic Communications

3.2.3.1 State of the Art

Recent 6G research shows that traditional data transmission alone is no longer sufficient for 6G networks, and that semantic-based communication, which focuses on transmitting only meaningful information, can reduce bandwidth usage while better supporting AI-driven services. A representative example is the SEM-NTN [14] framework, which integrates satellite and terrestrial networks through O-RAN and applies semantic feature extraction to transmit only the most important information. Building on this idea, the 6G-GOALS [15] project further introduces a semantic-aware and goal-based network architecture, where the network adapts its behaviour according to service objectives to enable efficient end-to-end control. These concepts have also been validated in practice, as real experiments demonstrate that AI-based xApps in O-RAN [16] can effectively reduce delay in applications such as video streaming. In addition, overview studies highlight how AI-native O-RAN architectures support challenging environments [17], including satellite, UAV, and HAPS networks, by providing flexible and intelligent control. Finally, recent URLLC-focused research [18] shows that deploying semantic xApps in the Near-RT RIC can reduce latency and improve reliability without relying on packet retransmissions, reinforcing semantic-aware control as a key foundation for future 6G networks.

3.2.3.2 Identified Gaps and Added Value of UNITY-6G:

Building upon these foundations, the UNITY-6G architecture Figure 3-4 introduces semantic communication as a system capability that facilitates the transfer of task-relevant meaning rather than transmitting all source bits equally. In practical terms, this means that instead of consistently forwarding entire data streams, the network can prioritize "what needs to be conveyed" (e.g., components of the scene for holographic/XR, state changes for digital twins). Due to limited spectrum, increased propagation delay, and intermittency limitations, this capability is especially important when NTN segments are included in integrated 6G deployments where network resources are limited. In the UNITY-6G architecture view, semantic communication is therefore treated as a cross-layer function spanning service endpoints, edge compute, and RAN control, with clearly defined placements for semantic processing and feedback. The architectural objective is not to redefine the radio or the transport stack, but to enable controlled reduction of data volume and adaptive quality control based on semantic relevance, while keeping compatibility with the existing O-RAN functional decomposition and UNITY-6G management principles.

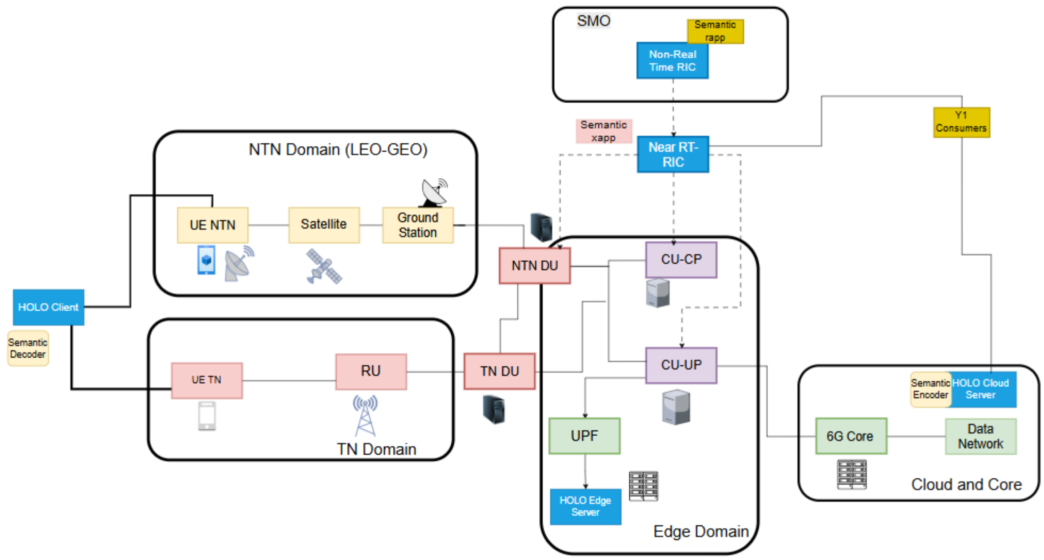


Figure 3-4. Semantic aware ORAN NTN-TN Architecture

The terrestrial semantic communication architecture is implemented by integrating semantic processing and control functions into an O-RAN-based TN domain. The semantic decoder is located at the HOLO client, which is connected via the UE to synchronize decoding with rendering and device context, while the semantic encoder is hosted at the HOLO cloud server and is connected via the Y1 interface to the semantic xApp. Semantic processing takes place at the service layer. This placement ensures that semantic data generation and interpretation remain independent of the radio implementation while still allowing adaptation based on network conditions. With the UE connected to the RU and DU, which serve as the execution points for radio access and scheduling, the TN connectivity adheres to the conventional O-RAN functional split. Instead of changing the radio stack itself, semantic awareness is introduced through interaction with the O-RAN control framework. The O-RAN RIC functions are used in the TN case to control and adapt semantic traffic. A semantic xApp hosted by the Near-RT RIC applies short-timescale control actions, such as modifying scheduling priorities or traffic handling rules for semantic flows, based on real-time measurements and feedback from the DU. Longer-timescale decisions, such as policy configuration, are handled by a semantic rApp located within the SMO's Non-RT RIC. Through the standard A1 and E2 interfaces, semantic control decisions are translated into executable RAN behaviours while preserving the O-RAN architecture and interface guidelines.

The O-RAN-based semantic control concepts are adapted to non-terrestrial connectivity by the semantic communication architecture in the NTN domain, where the radio access path consists of UE-NTN, satellite payload, and ground station components. Similar to the DU in the terrestrial scenario, the NTN domain feeds into an NTN-DU function. Semantic encoding is performed at the transmitting end (source side), while semantic decoding is performed at the receiving end (destination side), with optional split processing across both ends depending on the service implementation. The same RIC-based framework used in the TN case is used to implement semantic traffic control in the NTN domain. Short-timescale control actions are applied by a semantic xApp hosted in the Near-RT RIC depending on NTN-specific factors like congestion, delay, and link availability. Longer-timescale decisions, including semantic policy configuration and service parameter definition, are handled by a semantic rApp hosted in the Non-RT RIC within the SMO. Prior to traffic being sent over the satellite segment, the NTN-DU uses semantic-aware scheduling and prioritization to enforce these control decisions. By preserving the same control hierarchy and standard interfaces as in the terrestrial case, the UNITY-6G architecture supports unified semantic-aware operation across terrestrial and non-terrestrial domains.

In addition to the proposed semantic-aware architecture, contributions within UNITY-6G extend the framework toward intelligent optimization of traffic, timeliness, and energy efficiency. Distributed AI-based methods are introduced to jointly control semantic compression, traffic steering, and radio resource allocation through O-RAN RIC rApps and xApps, improving user QoE while reducing communication overhead across integrated TN-NTN deployments. From a user-centric perspective, semantic-aware scheduling is studied using perception-aware Age of Information (AoI) models that capture the trade-off between semantic accuracy and timeliness, including local versus remote processing. Furthermore, joint optimization of AoI, semantic data quality, and energy consumption is considered, with AI techniques such as deep reinforcement learning and feature extraction enabling adaptive operation across diverse scenarios and semantic tasks.

3.2.4 Scalability and sustainability considerations

The transition to 6G networks introduces a critical demand for massive scalability to seamlessly integrate heterogeneous environments, including terrestrial, non-terrestrial (NTN), and highly distributed cloud systems. To support this scale without overwhelming a single centralized controller, the UNITY-6G architecture introduces a hierarchical, vertical multi-domain orchestration model. Global intelligence is governed by an Inter-domain Management Object (IDMO), which maintains a broad overview of the entire network's capabilities and decomposes high-level service requests into necessary domain-specific compositions [19].

Operating beneath the IDMO, local intelligence is distributed across Domain Management Objects (DMOs). Each DMO acts within a distinct administrative or technological segment of the network, locally managing the lifecycles of resources and translating global commands into executable actions for underlying Infrastructure Managers (IDMs). To ensure these distributed components act cohesively, inter-layer communication is facilitated by a Service Based Management Bus (SBMA), which acts as middleware for continuous policy synchronization and service discovery across the massive 6G architecture.

Because traditional orchestration frameworks lack explicit considerations for environmental impact, the UNITY-6G architecture natively embeds a "Network-to-Energy" dimension to drive sustainability. This creates a system-of-systems that closely synchronizes network operations with the smart grid and local microgrids. At the foundation of this system, every distinct network domain is coupled with a localized Virtual Power Plant (VPP) as illustrated by Figure 3-5.

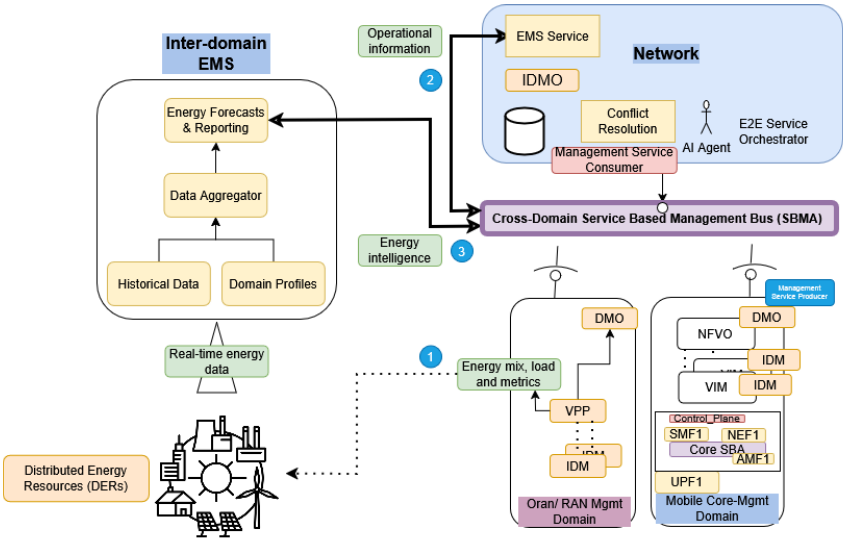


Figure 3-5. Energy-aware UNITY-6G Architecture

The VPP acts as an autonomous aggregation entity that treats decentralized power assets (e.g., renewable photovoltaics or wind generation and local storage capacities) as a unified power source. It continuously collects granular telemetry on the local network's energy usage and available green energy. These localized energy views are then propagated upward to a global Inter-domain Energy Management System (I-EMS). The I-EMS merges telemetry from all domains into a single, comprehensive view of the network's energy profile, utilizing historical data to forecast spatio-temporal variations in green energy availability.

To ensure orchestration remains fast and scalable, the I-EMS does not flood the main network orchestrator with raw power telemetry. Instead, it processes the data into simplified energy intelligence (e.g., signaling that Domain A possesses 50% green energy while Domain B has only 30%. The IDMO ingests these simplified metrics to fuel a sophisticated multi-criteria decision process, effectively evaluating candidate DMOs by balancing strict Quality of Service (QoS) parameters (like sub-20ms latency targets) alongside targeted sustainability goals.

This dynamic awareness allows the network to actively dictate where services should live based on environmental impact. When deploying network slices or heavy computational workloads, the IDMO actively prioritizes greener domains, such as explicitly instructing a DMO to instantiate 5G Core functions in a site currently running on 80% solar power. Supported by distributed AI-driven algorithms, this continuous, energy-aware placement of cloud-native network functions ensures that the network maximizes its use of renewables and minimizes its overall carbon footprint across the edge-cloud continuum.

3.2.5 Trust and security principles

The UNITY-6G architecture integrates trust and security as native, cross-layer properties of the system. Trust is not treated as a static prerequisite but as a dynamic and measurable attribute continuously evaluated throughout the service lifecycle. This subsection summarizes the core principles driving trust and security in UNITY-6G, which guide the design of functional blocks, interfaces, and management procedures across all domains. Those principles will guide the development of the Trust Layer and will be implemented through the components to serve the whole architecture.

3.2.5.1 Zero-Trust by design

UNITY-6G adopts a Zero Trust Architecture (ZTA) posture: no user, service, device, model, or domain is implicitly trusted. Every interaction is explicitly authenticated and authorized, and continuously re-evaluated based on evidence and context. Decisions are policy-driven, per-session, and revocable under changing risk conditions. This model eliminates static perimeter assumptions and supports multi-domain openness, where resources, domains, and service providers belong to different administrative entities.

3.2.5.2 Evidence-driven trust

Trust is treated as a measurable, evidence-based, dynamic property using both on-chain and off-chain sources. Level of Trust (LoT) is computed from device integrity and attestation proofs, operational telemetry, SLA and QoS monitoring metrics, AI-driven anomaly detection outputs, and behavioural analytics. The LoT is computed from MS and AE and the DE consume it as a constraint for admission, placement, scaling, and reconfiguration. ACT enforce decisions under guard-rails (parameter bounds, rate-limits). Evidence travels over the SMB, while the Data Continuum materializes selected streams and outcomes for audit, replay, re-training, and explainability, it provides a shared substrate for collecting evidences, while the Trust Layer transforms it into actionable trust scores and policies

3.2.5.3 DLT claims and auditability

Distributed Ledger Technology (DLT) anchors evidence, trust claims, and policy outcomes immutably, and support cross-domain verification via smart contracts and oracle/adapter functions. Trust claims and policy outcomes are anchored immutably and verified via smart contracts and oracle/adapter functions, keeping sensitive payloads off-chain (on-chain only hashes/indices/essential metadata). This provides verifiable, cross-domain evidence and non-repudiation without coupling to a single ledger technology.

3.2.5.4 Identity

The architecture supports federated/decentralized identities, Verifiable Credentials (VCs) and context-aware authorization (RBAC/ABAC), with least-privilege and need-to-know enforcement for entities across the TN-NTN continuum. This allows cryptographically secure onboarding, privacy-preserving authentication, cross-domain trust interoperability, fine-grained authorization based on verified claims.

3.2.5.5 AI-assisted & explainable trust

Analytics Engines (AE) compute LoT; in federated/multi-agent settings, trust weights contributions and reputation of participants. AI plays a fundamental role in evaluating trustworthiness it embeds behaviour-based trust evaluation, Explainable AI (XAI) to justify trust decisions and trust inference for distributed AI workflows such as Federated Learning. These capabilities ensure trust adapts to context, device mobility, and changes in operational conditions.

3.2.5.6 Conflict-aware security

When multiple DEs overlap, Conflict Management uses LoT-based priorities and constraints (policy-based, hierarchical, negotiation-based) to avoid oscillations and unsafe activations; intents, decisions, actions and feedbacks are traceable and linked to evidence/policies.

3.2.5.7 Secure Lifecycle for Multi-Vendor and Distributed Environments

Given UNITY-6G's emphasis on openness, the trust principles include: secure onboarding and provenance verification for third-party software (e.g., xApps/rApps), continuous attestation of components running in virtualized and cloud-native environments, domain agnostic enforcement through Trust Adapters and security aware orchestrators.

3.2.5.8 Privacy-Preserving Trust

To align with privacy requirements, the architecture uses selective anchoring of hashed evidence on DLTs, off-chain computation with on-chain proofs, privacy-preserving analytics for trust inference and minimal disclosure protocols for cross-domain trust.

3.3 GENERAL COMPONENTS

This subsection presents an initial description of the common components across domains, namely the Domain Management Orchestrator (DMO) and the Infrastructure Domain Manager (IDM), allowing the distributed service orchestration directed by the Inter-Domain Management Orchestrator (IDMO), as depicted in Figure 3-, included at the end of this section. This initial description serves as the foundation for the descriptions included in Section 4 for each architectural layer included in UNITY-6G architecture.

3.3.1 Inter-Domain Management Orchestrator

The Inter-Domain Management Orchestrator (IDMO) serves as the central intelligence of the UNITY-6G orchestration framework, enabling unified end-to-end orchestration and coordination across the heterogeneous domains distributed across the edge-cloud continuum. Its design is inspired in ETSI ZSM architectural principles and the Service Based Management Architecture (SBMA) described in 3GPP TS 28.533 [20], to enable the construction of complex management services and functions using the composition and interoperation of more basic components, i.e., the different domains.

Initially, we consider that the orchestration components of IDMO are structured into four primary elements: the Intent Manager, the Database Module, the Lifecycle Manager, and the Service Adapter[6]. The Intent Manager constitutes the entry point of the orchestration framework and is responsible for receiving, interpreting, validating, and decomposing user intents. It ensures that service requests, which could be expressed in structured or natural language, are converted into machine-actionable objectives. A more detailed description is found in Section 4.1.3.2. The Database (DB) module collects the information from the different domains in terms of available network service catalogue, computing resources and keeps track of available instances of running services across domains. After the intent is decomposed and the appropriate domains are selected, the Lifecycle Manager module at IDMO coordinates the orchestration sequence and manages the operational state of each sub-service throughout its lifecycle (instantiation, configuration, scaling, termination, etc.). It determines the correct order

[6] Additional components, like the Trust Adapter and the Conflict Management agent are other modules of the IDMO architecture enhancing and securing the management and orchestration processes beyond mentioned primary elements.

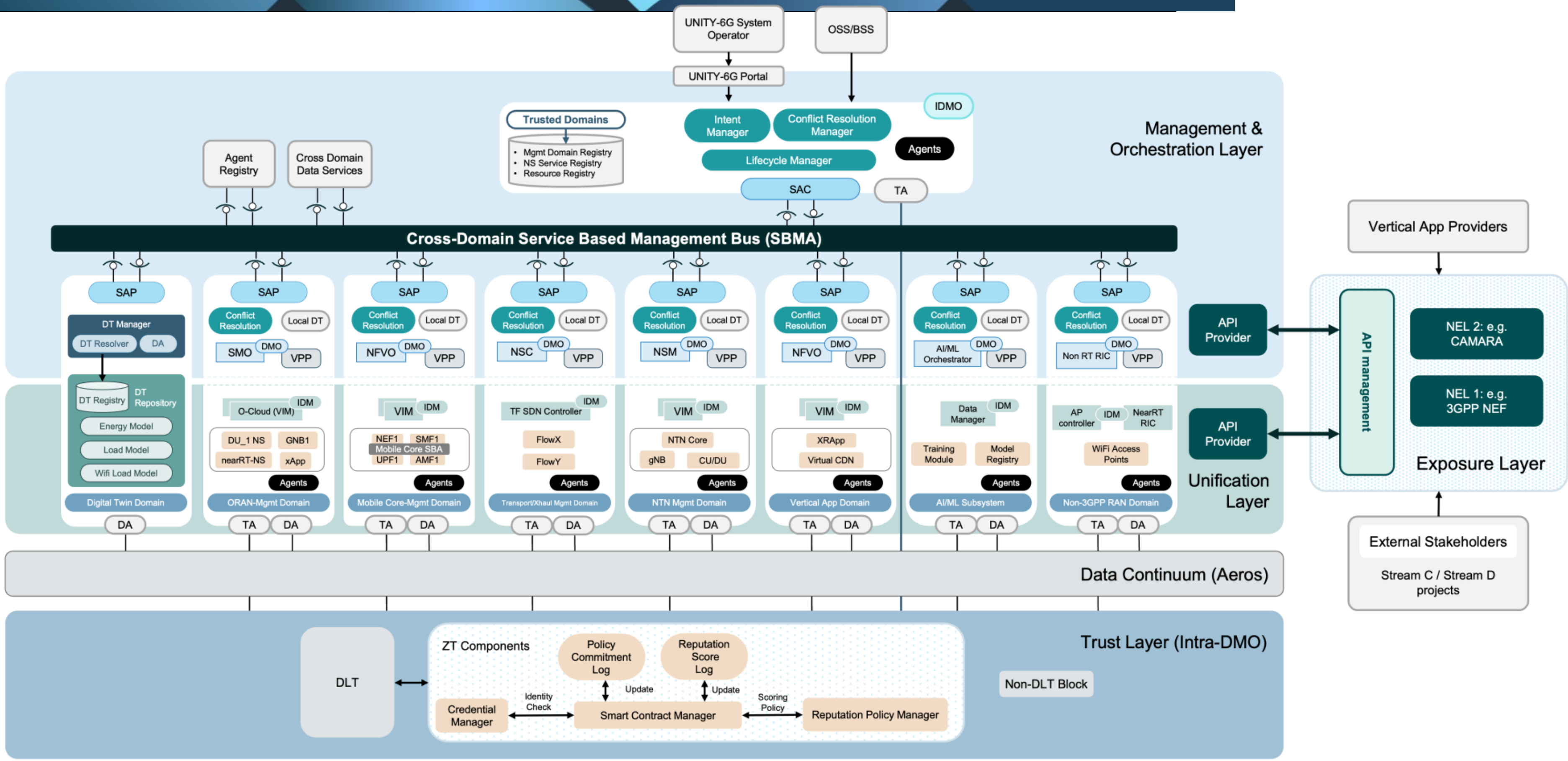
of execution ensuring dependencies between decomposed intents and produces corresponding messages to the underlying orchestration layers through the cross-domain SBMA bus. Requests are dispatched to the Service Adapter (SA) module, thus serving as the communication interface between the orchestration logic and the underlying DMOs placed at each domain. Note that multiple instances of domains of the same type (e.g. RAN or mobile core network domain) can and need to coexist within the UNITY-6G architecture to provide diversity in terms of functionality or geographical distribution. SA subscribes to and publishes messages on the cross-domain SBMA message bus, as explained in Section 4.1.3.4, mediating the exchange of operational data between the Lifecycle Manager, external systems, and the DB module.

3.3.2 Domain Management Orchestrator

The Domain Management Orchestrator (DMO) is the domain-level orchestrator responsible for the execution of domain-scoped service requests issued by the IDMO. Each DMO manages a single administrative and technological domain (e.g., RAN, Core Network, Transport), and based on the received requests, the DMO coordinates the execution of domain-internal orchestration workflows while enforcing domain-specific policies, constraints, and optimization objectives. More specifically, the DMO will receive translated requests coming from the Service adapter sub-module available within the domain connected to the cross-domain service-based management bus to perform the coordination of the lifecycle management of domain network services (e.g., a 5G core network instance), including instantiation, configuration, scaling, modification, and termination, while managing the dependencies and execution sequencing among domain-internal functions. In addition to this, the DMO will report to the service adapter module the collection and abstraction of domain-level information (e.g., service catalogues, resource availability) to update the domain-view included in the IDMO database module. That is, the DMO exposes its management and orchestration capabilities by interacting northbound with the IDMO through the cross-domain SBMA bus with the help of the service adapter module. Southbound, the DMO interfaces with one or more Infrastructure Domain Managers (IDMs) to request and control the infrastructure resources required for service realization. This separation enables domain autonomy while supporting end-to-end, cross-domain service management. The concrete realization of the DMO is domain dependent. For instance, those domains related with the creation of software-based network services (e.g., an instance of the 5G core), the DMO can be aligned with the NFV Orchestrator (NFVO) component of ETSI NFV specifications. Other example of DMO realization for an O-RAN type domain is the Service Management and Orchestration (SMO) component defined in O-RAN specifications. In the Transport domain, the DMO may be implemented by means of the Network Slice Controller, as developed in Section 4.2.3.4, which is responsible for the end-to-end lifecycle management of transport network slices.

3.3.3 Infrastructure Domain Manager

The Infrastructure Domain Manager (IDM) is the domain-internal management function component responsible for the control of underlying infrastructure resources (e.g., CPUs, GPUs, RAM memory, networking capabilities). It executes DMO requests, performing infrastructure-level lifecycle operations, including resource allocation, release and reconfiguration thanks to the exposure of resource inventory, capacity, availability and topology information. Like the DMO, the concrete realization of the IDM is domain dependent. In cloud-oriented domains, the IDM may correspond to a virtualization or cloud infrastructure management platform like Kubernetes or Openstack, thus aligned with the Virtual Infrastructure Manager (VIM) component of ETSI NFV. It is worth noting that within UNITY-6G project, there is a preference for the use of orchestration tools like Kubernetes, because there is a focus on the flexible lifecycle management of mobile network functions as applications benefitting from the properties of cloud-native principles. In the Transport domain, the IDM may be implemented as a Software-Defined Networking (SDN) Controller, responsible for the control and programming of transport network resources via standardized southbound interfaces. The IDM operates strictly within the domain boundary and does not perform cross-domain orchestration or decision-making.



Legend

AI/ML — Artificial Intelligence / Machine Learning
API — Application Programming Interface
BSS — Business Support System
CU/DU — Centralized Unit/Distributed Unit
DA — Data Adapter
DLT — Distributed Ledger Technology
DMO — Domain Management Orchestrator
DT — Digital Twin
DU_1 — Distributed Unit
GNB1/gNB — Next Generation Node B
IDM — Infrastructure Domain Manager
IDMO — Inter-Domain Management Orchestrator
NMS — Network Management System
NFVO — NFV Orchestrator
NEF1/SMF1/AMF1/UPF1 — Network Exposure/Session Management/Access and Mobility Management/User Plane Function
NEL — Network Exposure Layer
NS — Network Service
NSC — Network Slice Controller
NTN — Non-Terrestrial Networks
OSS — Operations Support System
PEP — Policy Enforcement Point
SAC — Service Adapter - Management Service Consumer
SAP — Service Adapter - Management Service Producer
SMO — Service Management and Orchestration
SDK — Software Development Kit
TA — Trust Adapter
TF SDN — Teraflow Software Defined Networking
VIM — Virtualized Infrastructure Manager
Virtual CDN — Virtual Content Delivery Network
XRApp — Extended Reality Application

Figure 3-6. UNITY-6G Detailed Architecture Overview

4 ARCHITECTURE LAYERS

This section provides a more detailed description of each UNITY-6G architecture layer. As Figure 3- shows, the architecture is divided into five main layers. Firstly, the Management and Orchestration Layer, described in Section 4.1, coordinates the end-to-end service lifecycle, intent handling, and cross-domain orchestration. Then, in Section 4.2, the Unification Layer harmonizes the capabilities of all technological domains and exposes them as services through a common management interface. The Data Continuum Layer, presented in Section 4.3, provides a federated, semantic, and governance-driven substrate for data produced and consumed across all UNITY-6G domains. Moreover, in Section 4.4, the Network Exposure Layer offers secure, standardized APIs that expose network and compute capabilities to internal and external consumers. Lastly, Section 4.5 presents the Trust Layer, which ensures secure, zero-trust operation through identity, attestation, trust scoring, and verifiable auditability.

4.1 MANAGEMENT AND ORCHESTRATION LAYER

The Management and Orchestration layer is the “brain” of UNITY-6G. It coordinates end-to-end service lifecycle management across all domains through a hierarchical structure composed of the IDMO, DMOs, and IDMs. This layer enables intent-driven orchestration, distributed AI-native closed loops, conflict management, service exposure, and energy-aware decision making. It ensures that services are instantiated, adapted, and optimized consistently across different 6G resources.

4.1.1 Objective

The objective of this layer is to provide a unified, intent-driven control plane that coordinates the end-to-end service lifecycle across heterogeneous domains (O-RAN, Core, Transport, NTN, Non-3GPP, Vertical Apps) using a hierarchical orchestration stack among the general components described previously (IDMO and the different domains including DMO and – IDM/s) and supported with the activation of AI-native closed loops (MS–AE–DE–ACT) for scalable, secure, and energy-aware automation.

4.1.2 Key Functions

The key functions listed for this layer are:

- Intent Management and Decomposition: Receive business/service intents, refine and translate them into actionable goals and orchestration workflows; align with policy constraints and negotiate conflicts prior to deployment.
- Conflict Detection and Resolution: Detect overlaps in decision spaces and resolve via priority-based, hierarchical, or negotiation-based strategies, ensuring coherent actions reach actuators.
- End-to-End Service Orchestration, Lifecycle Management, and Cross-Domain Coordination: orchestrate multi-domain services through the IDMO, which coordinates the different registered DMOs and their underlying IDMs through the Service-Based Management Bus to enable discovery, capability exposure, topic/event exchange, and orchestration messaging across all domains.
- Exposure Alignment: Coordinate with the Network Exposure Layer so orchestrated service instances and capabilities are consistently exposed via APIs.

4.1.3 Components

4.1.3.1 Service Orchestrator

The Service Orchestrator is a key component of the UNITY-6G management and orchestration layer. In the project, this functionality will be implemented using NearbyOne, which acts as the DMO for cloud-native service orchestration and automation. This platform is responsible for the automated deployment, coordination, and lifecycle management of services across the distributed infrastructure, multi-cloud, edge computing, and telecom environments.

NearbyOne solution provides mechanisms to automate and orchestrate the infrastructure and the deployment of components, both nodes (edge sites) and applications.

- **Infrastructure Management**: NearbyOne allows to fully manage the life cycle of the infrastructure resources as a service. It supports all kinds of heterogeneous physical and virtual infrastructure including edge devices, Hyperscalers, Public, Private and Hybrid clouds. In UNITY-6G, we leverage the capability of NearbyOne to manage and orchestrate multiple sites located in arbitrary and potentially separate locations, built upon a Kubernetes system to manage Cloud-native Network Functions (CNFs) and applications.

Application and Service Operations: Applications and services are the central component of any deployment, and they require the seamless integration of components provided by third-party software developers or vendors. Service management and orchestration enables workflow automation, cloud-native orchestration, and dynamic policy enforcement, ensuring that services are instantiated, monitored, and optimized across diverse infrastructure components.Figure Figure 4-1 shows the architecture and main interactions of the service orchestration platform in the UNITY-6G ecosystem.

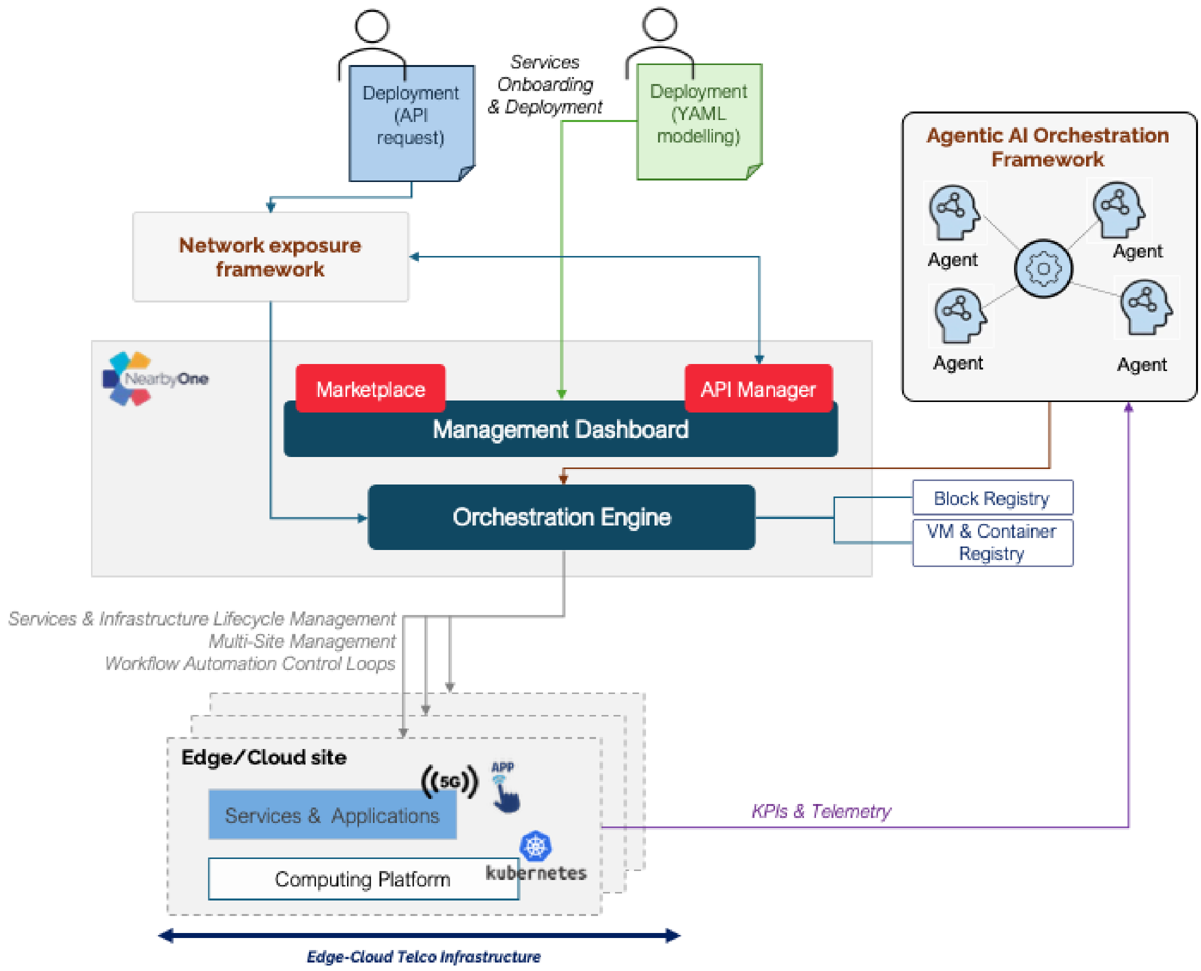


Figure 4-1. Service Orchestration based on NearbyOne

The service orchestration platform is composed of several key components and supporting elements:

- **Management Dashboard**: Provides an intuitive user interface that serves as a unified operational and governance surface. Through the dashboard, users such as operators, administrators, and developers can seamlessly navigate between infrastructure management, service orchestration, and API exposure tasks.
- **Marketplace***:* Offers access to a catalog of solutions, easily extendible, available for rapid service deployment through the NearbyOne Dashboard. It facilitates the choice and chained deployment of services and applications with only one click.

- **API management** module: Provides a unified platform to onboard, secure, monitor, and manage APIs. This component ensures controlled exposure and governance of service APIs across the platform.
- **Orchestration Engine**: Acts as the main operational component responsible for managing complex and distributed operations, enforcing eventual consistency rather than transactions. It follows the pattern of self-healing reconciliation loops. It also includes components that manage the policies and other dynamic behavior related to services and devices.
- **External Artifact Repositories**: Besides the NearbyOne components, the orchestration environment relies on external entities necessary for the operation. These are the Block, VM and container registries, which are the repositories used to store and access application or services artifacts and container and VM images.

The service orchestration supports seamless integration with the **Network exposure framework** (see Section 4.4) for external interoperability. The integration provides API resources that enable the exposure of pre-defined operations, such as the lifecycle management of applications as provided by the CAMARA Edge Application Management API.

The service orchestration platform also supports integration with the **Agentic AI Orchestration Framework**, which enhances the UNITY-6G management and orchestration layer by introducing intelligent, autonomous agents that support adaptive and data-driven service orchestrations. The framework enables continuous observation, reasoning, and action across applications, infrastructure, and network resources. The framework leverages a multi-agent architecture where the individual AI Agents analyse KPIs and telemetry collected from edge and cloud sites to detect anomalies and optimize service behaviour. The agents operate in a collaborative manner towards achieving a common goal. The agents requisite to realize the end-to-end autonomous service orchestration include:

- The intent Agent, responsible for converting natural-language requests into structured goals,
- The observability Agent, responsible for turning raw telemetry into alerts and recommended actions including identification of intent violations and system state changes,
- The planning Agent, responsible for generating multi-step orchestration plans, and
- The Action agent, responsible for executing validated plans via relevant APIs.

These agents interact with the NearbyOne platform through exposed APIs, triggering orchestration actions via the Orchestration Engine in compliance with service orchestration objectives while respecting existing policies. Through integration with the Network Exposure Framework, the Agentic AI framework can leverage network-level capabilities and expose AI-assisted operations in a secure and interoperable manner. This results in self-optimizing and self-healing services, complementing NearbyOne's deterministic orchestration with proactive intelligent automation suited for dynamic 6G edge–cloud environments targeted by the UNITY-6G project. Next section highlights the relevance of proactive orchestration.

4.1.3.1.1 CNF Proactive Orchestration using ML

Within the UNITY-6G architecture, Artificial Intelligence is not conceived as a standalone optimization layer operating on top of traditional management and orchestration functions. Instead, AI is integrated as a **native** control primitive, enabling proactive, intelligent, and autonomous network operation. In this context, ML techniques play a key role in enabling proactive CNF orchestration, shifting the paradigm from reactive management to predictive and anticipatory control.

One of the main pillars of proactive CNF orchestration is **proactive CNF placement and scheduling**. The core idea is to continuously analyze multi-source telemetry data, such as network traffic patterns, service KPIs, and container-level metrics. By leveraging ML models capable of capturing both temporal and spatial correlations in this data, the orchestrator can predict future resource demand and workload evolution across the infrastructure. These predictions allow the orchestration layer to anticipate changes in service requirements and take informed decisions ahead of time. Based on the predicted demand, the orchestrator can proactively allocate, scale, or relocate CNFs before resource shortages or performance degradation occur. This prevents cold-start delays and reactive scaling actions, which are known to negatively impact service latency and SLA compliance. Conversely, when a decrease in resource demand is predicted, the orchestrator can downscale CNFs or consolidate workloads, freeing up computational and network resources for future needs. This predictive resource management enables more efficient utilization of the infrastructure while maintaining service quality.

Proactive CNF orchestration is particularly relevant in **resource-constrained and heterogeneous environments**, such as the NTN domain, where energy and connectivity resources are limited and highly variable. In this scenario, the ability to anticipate demand and optimize CNF placement across edge, cloud, and NTN segments is essential to reduce unnecessary resource usage and improve energy efficiency. Overall, ML-driven proactive orchestration in UNITY-6G contributes to reduced energy consumption, improved SLA assurance, and increased network resilience.

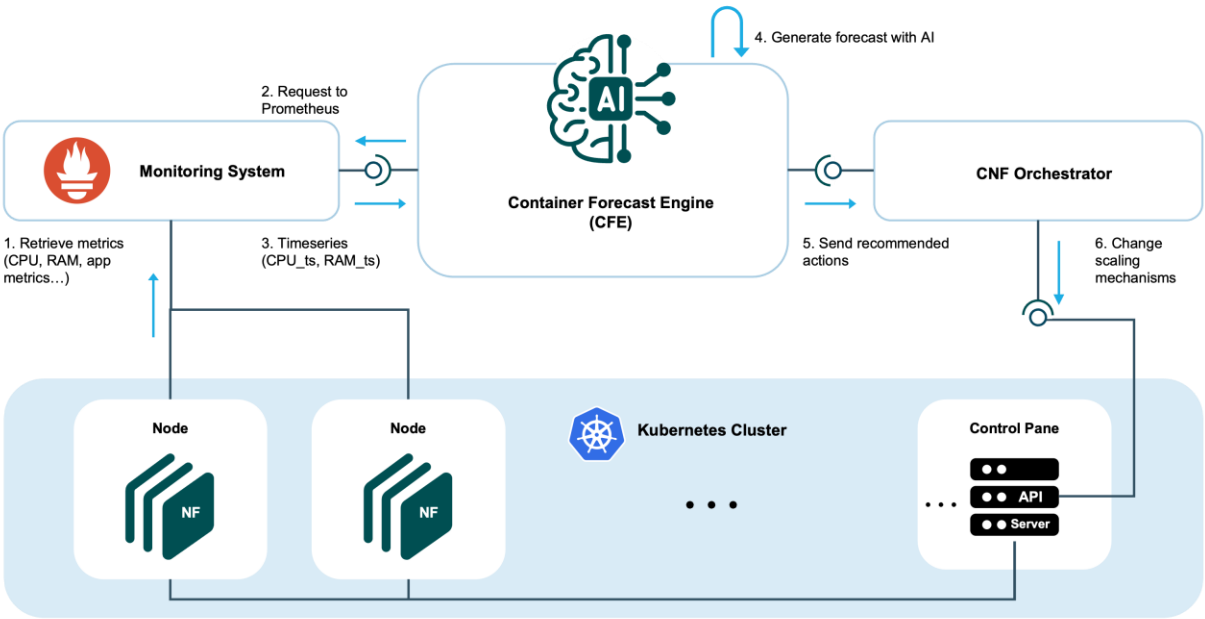


Figure 4-2. ML to enable proactive orchestration of CNFs

4.1.3.2 Intent Manager

The intent manager is in the orchestration and management layer of the UNITY-6G architecture and is part of the IDMO. Its role is to manage "intents." Intents have multiple definitions according to standards; for example, in 3GPP, an intent specifies expectations, including requirements, goals, and constraints for a specific service or network management workflow [13].

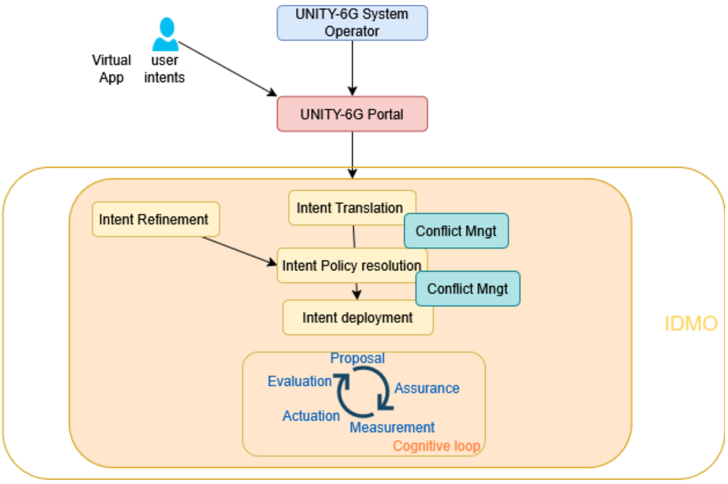


Figure 4-3. UNITY-6G Intent Manager

An intent is translated from the business level into a set of SLAs, which are then converted into configurations for deployment. A business intent represents the needs described in natural language by UNITY-6G clients for 5G connectivity. Once the intent is received through the

UNITY-6G portal, it is translated into deployment configurations. However, prior to this, an intent refinement step occurs, during which the intent is negotiated and clarified with the client to gather all necessary information before deployment.

The intent manager also incorporates intent policy resolution and interacts with the conflict manager before deploying the intent. Here is a description of the role of each component presented in Figure 4-3:

- **Intent Refinement:** clarifies customer needs and expectations, prioritizes intents based on business impact, and uses feedback to improve understanding.
- **Intent Translation:** converts customer language into technical specifications, maps intents to relevant services or actions, and ensures consistency across different systems.
- **Policy Resolution:** identifies appropriate solutions for customer issues, automates responses where possible, escalates complex cases to human agents, and confirms resolution with the customer.
- **Intent Deployment:** monitors service performance and quality, detects and addresses anomalies proactively, and validates that solutions meet customer expectations.

Once the intent is deployed, the cognitive loop collects measurements from the network using UNITY-6G tuples (i.e., MS, AE, and DE) to ensure SLAs are respected during deployment. In the UNITY-6G architecture, this cognitive loop is based on TMForum definitions. It consists of the following elements:

- **Proposal:** establishing viable solutions
- **Evaluation:** determining how closely the proposed solutions meet the intent requirements
- **Actuation:** executing the devised solution to maximize the system's utility
- **Measurement:** continuously observing and measuring the system's state for each requirement in the intent
- **Assurance:** verifying whether the system meets the stated requirements

4.1.3.3 Virtual Power Plant

The Virtual Power Plant, or VPP, functions at the local domain level while taking into account centralized orchestration. It continuously monitors and actively manages the combined energy status of all Infrastructure Domain Managers, known as IDMs, present within the domain. Through this role, the VPP brings together and orchestrates a diverse set of assets, which include distributed energy resources such as photovoltaic panels, small wind turbines and other local generation units, various forms of energy storage like batteries, as well as flexible and controllable loads that can adjust consumption patterns. In its core operation the VPP gathers real-time telemetry data related to energy flows, production, consumption and status, directly from the domain infrastructure through connected IDMs. It then fuses all this incoming information to produce a single, coherent picture of the domain's overall energy situation. That unified view encompasses net load profiles, available local generation capacity, the current proportion of renewable or green energy, and other essential abstractions of the underlying resources. To the outside world the VPP presents clean, virtualized interfaces that allow energy providers, market participants and even embedded local sources to interact with what appears to be one unified, controllable entity rather than many separate devices. This approach supports higher-level network coordination and system-wide optimization while leaving the operational autonomy and direct grid control responsibilities of the individual Domains and local sub-grids completely untouched.

When participating in electricity markets, the VPP serves as the sole gateway for the entire local domain. It handles communication with the Transmission System Operator or other counterparties by employing widely accepted energy market protocols. Through these standardized channels the VPP manages nomination processes for day-ahead and intraday schedules, receives and executes real-time dispatch instructions, follows scheduling commitments, carries out operational setpoints and continuously returns accurate telemetry feedback consisting of measured values and status updates.

The three principal categories of signals it must process reliably are therefore scheduling information, operational commands that include immediate control actions and dispatch orders, and telemetry streams that provide ongoing measurement and reporting back to market actors and system operators:

- **Scheduling Signal (VPP → TSO)**: The inter-domain Energy Management System (IEMS) performs forecast, sends to the VPP, and the VPP submits it to the TSO.

- **Operational Signal (TSO → VPP):** TSO sends a command to the VPP. The VPP accepts it and the routes the requirement internally, via SBMA BUS, to the IEMS which in turn translate and conveys the message to IDMO.
- **Telemetry (VPP → TSO)**: The VPP aggregates real-time data from the IDM's and reports the net status to the TSO.

The above signaling will try to cover the attributes listed below for the Energy Market participation:

Attribute	**Value (typical)**
Preparation period	Defined in terms and conditions for FSPs
Ramping period	TBD
Full activation time	15 minutes
Minimum quantity	1 MW
Maximum quantity	TBD
Deactivation period	TBD
Granularity	0.01 MW or 0.001 MW
Minimum duration of delivery period	TBD
Maximum duration of delivery period	TBD

Co-funded by the European Union

Validity period	15 minutes
Mode of activation	Manual
Activation price	Yes
Aggregation allowed	Yes

Table 4. Value for Energy Market Participation

Architecturally the VPP maintains both physical and logical connections to the underlying network while fully preserving the independence of local grids and Domains. At appropriate intervals it forwards comprehensive, aggregated status reports at the VPP level upward to any superordinate inter-domain Energy Management System or higher coordination layer. In essence the VPP transforms a scattered group of heterogeneous, geographically distributed energy resources into one cohesive, fully observable and market-compatible virtual power plant. This enables effective, efficient and economically attractive participation in energy and flexibility markets without compromising local control authority or overall grid stability.

4.1.3.4 Cross-Domain Service Based Management Bus

This component provides the logical connection between the different domains and the IDMO. Through this bus, the domains register the IDMO and exchange information objects (e.g., JSON-formatted messages) to populate IDMO database modules by announcing thedomain capabilities. This information includes service catalogues, computing resource availability, potential geographical footprint, and transport network interconnection points. It is worth noting that this information is continuously updated. Additionally, lifecycle management operations (e.g., instantiation, termination) are also exchanged between IDMO and the different domains through the cross-domain SBMA bus.

Depending on the scenario and expected features (i.e., scale, target domain, latency, etc.) and focus (messaging, cache and memory, event streaming), various solutions can be considered, including Apache Kafka, gRPC, Apache Pulsar, RabbitMQ, Zero Message Queueing (ZMQ), Redis, MQTT. However, the choice of communication protocol and messaging topic organization can be left to implementation. Some principles that can be adopted for the organization of the communication between IDMO and the different domains are: a) a single well-known registration topic to ensure a single entry point to join the system; b) per-domain topics after domain registration for bidirectional communication and c) per-request/service topic to coordinate the orchestration of different domains and exchange of information among domains involved in a given service request.

4.1.3.5 Conflict Management

Each network domain, such as RAN, Core Network (CN), or transport, can be considered an autonomous operational entity equipped with a domain-specific management and orchestration entity referred to as the DMO [21]. Examples include the Service Management and Orchestration (SMO) component in Open RAN and the Network Functions Virtualization Orchestrator (NFVO) in the 5G Core, both responsible for managing domain resources. In such a unified architecture, a supervisory cross-domain component called in UNITY-6G as the Inter-Domain Management and Orchestrator (IDMO), enables integrated delivery of network services across domains. However, DMOs are not limited to managing intra-domain resources; they must also address operational challenges such as conflicts between concurrently running applications. In Open RAN environments, conflict management (CM) becomes particularly challenging due to independently developed intelligent applications (xApps and rApps) that may produce conflicting control actions, as reported in [22] [23] [24] [25]. Consequently, multi-domain resource management can naturally be interpreted from a multi-agent perspective, where CM mechanisms deployed in individual domains operate autonomously. As illustrated in Figure 4-4, the IDMO may operate as a supervisory layer coordinating underlying domains to support end-to-end (E2E) service delivery. Communication between the IDMO and domains can be realized through a service management bus (as described in the section 4.1.3.4), while direct inter-DMO communication may further improve coordination efficiency.

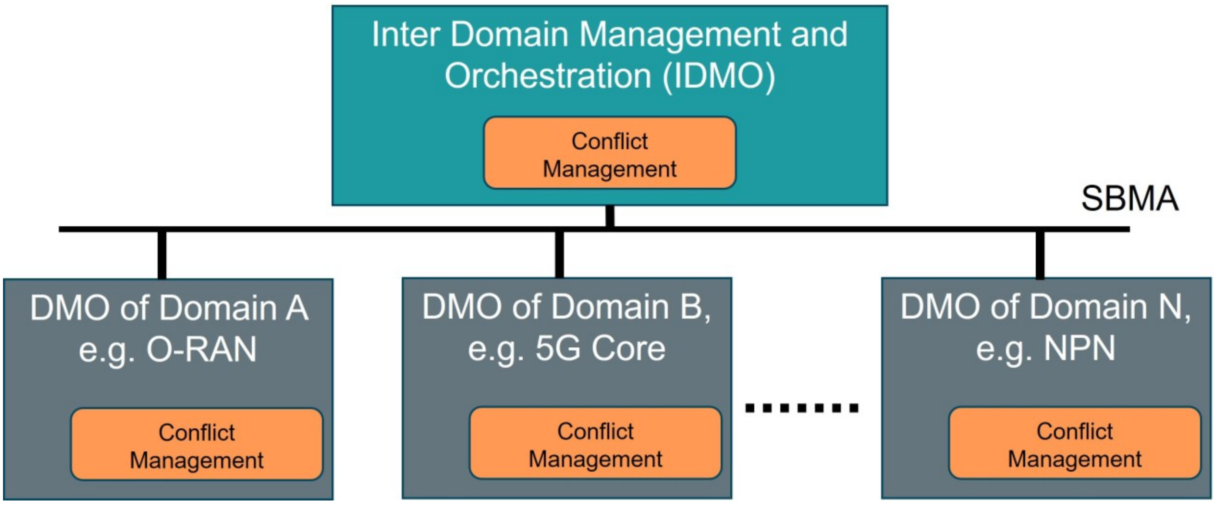


Figure 4-4. UNITY-6G Conflict Management Architecture

Independent operation of domain-level CM modules may therefore lead to globally suboptimal outcomes, since decisions taken in one domain can impact neighbouring domains. Furthermore, when IDMO-based coordination is introduced, asynchronous E2E resource allocation decisions must be considered. For instance, IDMO may enforce cross-domain resource allocation to enable deployment of a new service; however, if such decisions are not synchronized with domain-level controllers, conflicts may arise.

Introduction of IDMO with its CM opportunities enables effective yet advanced and complicated approaches for cross-domain resource optimization and effective creation and deployment of E2E services. An illustrative, yet simplified example of the prospective conflict that arises between the particular domain and the IDMO CM module can be described as follows. The DMO (implemented as SMO in Open RAN) resolves conflicts between independently operating xApps, such as the Energy Saving xApp (ES-xApp, that minimizes energy consumption by redistributing users among base stations and disabling underutilized cells) and the Quality-of-Service xApp (QoS-xApp, that aims to maximize per-user resource allocation and distribute load across available stations). These objectives naturally create conflicting control actions resolved by the domain-CM module. Simultaneously, an upper-layer IDMO translates incoming intents [26] into deployment actions into specific Open RAN domain. Since service requests may arrive asynchronously with respect to domain-level control policies, performance variations may occur depending on the applied CM mechanisms. Such simple example illustrates the prospective cross-domain conflict; it will be further investigated in the remaining project lifetime.

4.1.4 Interfaces

The Management and Orchestration layer exposes two main interfaces. First, the northbound interface with the Intent Portal acts as the entry point through which high-level intents and service requests are delivered to the IDMO. Through this interface, users or external systems express their requirements, while the IDMO validates, refines, and translates them into actionable orchestration workflows. The interface also provides feedback to the requester, including deployment status, lifecycle events, and assurance information generated during service execution.

Secondly, the layer relies on the Cross-Domain Service-Based Management Bus as the core mechanism enabling coordination between the IDMO and the various domain-level orchestrators. This bus supports domain registration, capability exposure, and continuous synchronization of resource and service catalogues. It also carries all orchestration-related exchanges, such as lifecycle operations, per-domain and per-request topics, event notifications, and state updates, ensuring consistent, scalable, and loosely coupled communication across all UNITY-6G domains.

4.2 UNIFICATION LAYER

The Unification Layer aggregates and harmonizes the capabilities of the various UNITY-6G technological domains, providing a normalized abstraction for orchestration. UNITY-6G architecture considers seven different domains: ORAN, Core Network, NTN, Transport, Non-3GPP RAN, AI/ML, Vertical Application and Digital Twin. Each domain exposes its capabilities through a common Service-Based Management Bus, enabling cross-domain coordination while retaining domain autonomy. The unification layer effectively acts as the "execution fabric" where domain-specific functions, controllers, and resources are orchestrated into coherent end-to-end services. All domains share the same basic structure: a service adapter that translate the domain request to specific instructions for the orchestrator to handle, the DMO, defined in section 3.3.2, which then trigger the allocation of resources at its underlying IDM/s, defined in section 3.3.3, and the adaptors to interact with the Trust and Data Continuum layer.

4.2.1 Objective

The objective of the Unification Layer is to ensure consistent, service-based exposure and operational alignment of all UNITY-6G domains, so that orchestration decisions issued by the IDMO can be executed uniformly across different technological and administrative domains. It provides the architectural environment in which domain-specific controllers, functions, and resources can be seamlessly combined to form coherent end-to-end services without requiring the IDMO to deal with domain-internal complexity.

4.2.2 Key Functions

The key functions listed for this layer are:

- Expose domain capabilities in a unified manner, ensuring that each domain presents its services, catalogue, telemetry, and operational state through standardized service interfaces rather than domain-specific mechanisms.
- Translate orchestration requests into domain-specific actions, allowing the IDMO to issue technology-agnostic commands while each domain executes them using its own DMO, IDM/s, and internal workflows.

- Enable cross-domain coordination, ensuring that dependencies between domains are resolved and that domain actions are synchronized to support consistent end-to-end behaviour.
- Maintain domain autonomy, ensuring that each technological domain retains its own optimization logic, resource management policies, and internal execution paths while still participating in unified orchestration.

4.2.3 Components

4.2.3.1 ORAN domain

4.2.3.1.1 Overview

The Open Radio Access Network (O-RAN) domain provides the capabilities required in UNITY-6G for the deployment and operation of (pre-)6G radio access network (RAN). In more detail, the O-RAN paradigm has provided 5G systems with a significant advance from the traditional RAN approach, by promoting an open, disaggregated, and intelligent architecture.

4.2.3.1.2 Architecture

Moving away from a monolithic implementation, O-RAN facilitates the distribution of the RAN functions by dividing the gNB into three main components, namely the Central Unit (CU), Distributed Unit (DU), and Radio Unit (RU) [27], and by defining standard (open) interfaces and different functional splits, such as the 7.2 which relies on the open fronthaul (OFH) specifications [28]. Disaggregation greatly enhances scalability, flexibility, and vendor interoperability, while facilitating RAN function virtualization (vRAN). Moreover, RAN softwarization and intelligence are at the heart of O-RAN, which has also introduced the RAN Intelligent Controller (RIC) to support the integration of third-party AI-based applications for dynamic network control and optimization. The E2 interface connects the CU and DU to the near-real-time (RT) RIC to deploy xApps and implement control loops, with latencies between 10 milliseconds and 1 second [29]. Similarly, the O1 interface connects the gNB to the non-RT RIC to deploy rApps and implement non-time-sensitive tasks aimed at Service and Management Orchestration (SMO) automation [30].

The O-CU/O-DU solution used in UNITY-6G will be based on a software-defined radio (SDR)-based gNB implementation, which is part of OCUDU [31], an open-source commercial-grade 5G CU/DU from SRS. Previously, it was srsRAN Project [32], which included advanced features only available from its commercial counterpart srsRAN Enteprise 5G [33]).. OCUDU provides a complete RAN solution compliant with the 3GPP and the O-RAN Alliance specifications and includes the full L1/2/3 stack with minimal external dependencies. The software is portable across processor architectures and scalable from low-power embedded systems to cloudRAN, providing a powerful platform for mobile wireless research and development. Figure 4-5 shows the main elements of the O-CU/O-DU solution provided by OCUDU, which will conform the initial blueprint of the O-RAN-based RAN architecture of UNITY-6G.

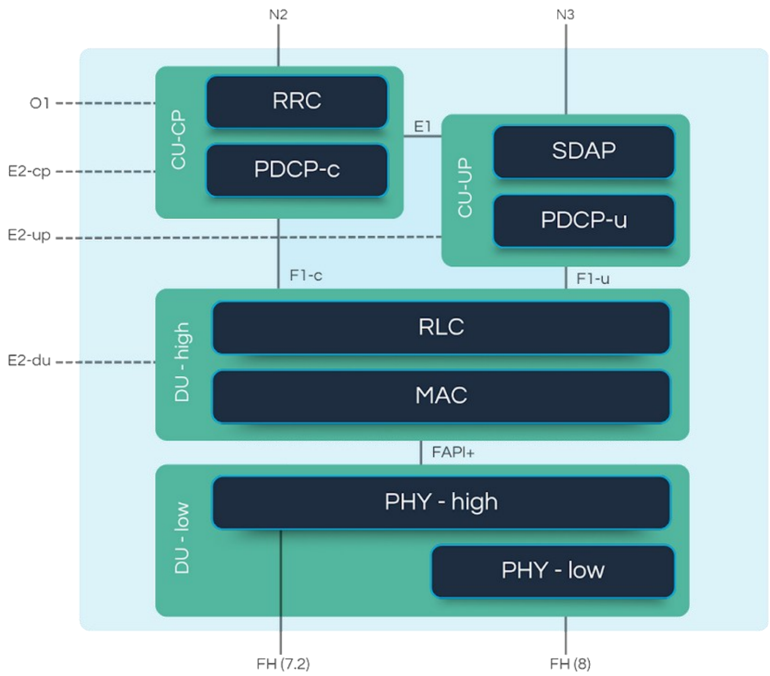


Figure 4-5.Overview of the O-CU / O-DU solution provided by OCUDU.

In terms of features, amongst others, the O-CU/O-DU solution at provides support for all FR1 bands (TDD), with up to 100 MHz signal bandwidth (15 and 30 kHZ SCS), all PHY channels, RLC and MAC procedures, up to 4-layer MIMO, split CU (CU-CP and CU-UP) and DU (can be deployed as independent components), and initial NR-NTN features, which will be used in the NTN domain (which is introducted in Section 4.2.3). The latter is also being actively extended in UNITY-6G, with the aim to allow regenerative payload architectures by leveraging the CU/DU split features of the O-RAN gNB.

It must be underlined that the O-CU/O-DU implementation follows the current standard specifications (for both components and interfaces), and that the features described above are to be considered the baseline upon which UNITY-6G will build upon to add the necessary modifications and extensions (e.g., add latency-bounding features towards a TSN-enabled RAN). In this regard, strong adherence and alignment to current 3GPP and O-RAN standards is a key aspect when defining the enhancements and extensions required by UNITY-6G.

Amongst others, this facilitates backwards compatibility and adoption of the solutions proposed by the project. Consequently, the enhancements envisioned for the O-CU/O-DU will be designed with that principle in mind.

4.2.3.1.3 Interfaces

The O-CU/O-DU implemented by OCUDU includes the required 3GPP and O-RAN interfaces to enable disaggregated deployments and interact with the other (third-party) O-RAN components of the UNITY-6G architecture. In more detail, the following included interfaces are particularly relevant to the project:

- F1-c interface connecting the O-DU with the O-CU-CP.
- F1-u interface connecting the O-DU with the O-CU-UP.
- NG interface connecting the gNB with the Core Network.
- O-RAN Open Fronthaul (OFH) interface connecting the O-DU with the O-RU (based on an in-house implementation of the OFH).
- E2 interface connecting the O-CU/O-DU with the near-RT RIC, including support for the following services:
    - E2SM-KPI for RAN monitoring purposes, with support for >15 O-RAN-defined metrics
    - E2SM-RC for RAN control purposes, with support for slice-level PRB quota allocation and (basic) handover control.
    - E2SM-CCC for cell configuration and control purposes, which is being developed as part of the UNITY-6G activities.
- O1 interface connecting the O-CU/O-DU with the non-RT RIC (SMO), which is being extended as part of the UNITY-6G activities.

4.2.3.2 Core Network domain

4.2.3.2.1 Overview

A Core network domain instance provides the management and orchestration capabilities required for the deployment and operation of 5G/Beyond 5G Core Network Functions (NFs) instances, which are developed based on the 3GPP Service-Based Architecture (SBA).

4.2.3.2.2 Architecture

In the core network domain, the Service Adapter (SA) receives the requests from the cross-domain service management bus, processes and enriches these requests to send commands

Co-funded by the European Union

to the DMO, which is responsible for orchestrating the lifecycle of 5G Core NFs in the underlying available IDM(s) registered at DMO. Examples of DMO implementations can be an ETSI NFVO component, like ETSI Open Source MANO (OSM) or the Service Orchestrator described in Section 4.1.3.1. With regards or IDM implementation, the preferred option is the use of Kubernetes cluster instances, as mentioned in Section 3.3.3. This schema is represented in Figure 4-6, focusing on general components of the UNITY-6G architecture included in the core network domain.

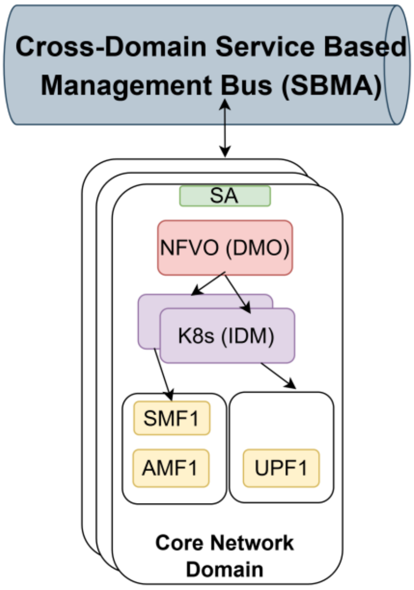


Figure 4-6. Core Network Domain Architecture

In terms of mobile core network function artefacts, a fundamental characteristic of the (5G) Core domain allowed by its SBA is the support for explicit separation between Control Plane (CP) and User Plane (UP) functions, as defined by 3GPP. Control Plane NFs, including AMF, SMF, AUSF, UDM, PCF, and NRF, are realized as service-based functions exposing standardized interfaces and can be instantiated, scaled, and updated independently of User Plane Functions (UPFs). This separation enables flexible deployment strategies, where CP functions may be centralized or regionally distributed, while UPFs can be dynamically deployed closer to the network edge to meet latency and throughput requirements in single or multi-slice deployments. This approach is aligned with 5G/Beyond-5G use cases and it represents a challenge for the continuous development of UNITY-6G architecture where core network and O-RAN domains may have co-existing/overlapping computing resources at the same location managed by the shared IDM instances or different ones with available network connectivity. Moreover, the mobile core network artefact can be equipped with monitoring capabilities or connected to the Network exposure framework, developed in task T3.2, thus allowing the exposure of metrics of interest or the modification of its capabilities, as explained in Section 4.4.

4.2.3.2.3 Interfaces

Within the core network domain, the following interfaces are considered for the lifecycle management and the exposure of capabilities:

- The service adapter, after processing and enriching the received request from the cross-domain service management bus contacts the northbound interface offered by the DMO in the form of REST-API based endpoints.
- Once a core mobile network instance is deployed, it can use the Network Exposure Framework to securely expose network events, analytics, QoS impact and traffic management functionalities.

4.2.3.3 NTN domain

4.2.3.3.1 Overview

In the UNITY-6G architecture, the NTN infrastructure provides connectivity to user terminals by means of satellite links to extend the coverage (complementing the terrestrial network) and increase the resiliency of the network, e.g., in disaster scenarios. In the UNITY-6G architecture, NTN is integrated with cellular networks. In particular, referring to Figure 3-, the NTN domain comprises the IDM, which controls underlying infrastructure resources, i.e., the NTN core and the gNB, which can be split into CU, DU, and RU, and the IDM fulfils the VIM function. As in the other domains, the IDM interacts with DMO.

4.2.3.3.2 Architecture

NTN platforms include: i) spaceborne satellites, such as very low Earth orbit (LEO) and LEO (between 300 km and 2000 km of altitude), medium Earth orbit (MEO) with an altitude between 8.000-20.000km, and geostationary Earth orbit (GEO) at 35.786 km above the Earth's equator, and ii) airborne vehicles operating at altitudes typically between 8 and 50 km, including high altitude platforms (HAPs).

NTN platforms carry a payload that can be either transparent (bent-pipe payload) or regenerative. A transparent payload only performs changes to the frequency carrier of the received uplink radio frequency (RF) signal, filtering and amplification of the signal before transmitting it on the downlink, i.e., the platform corresponds to an analogue RF repeater. A regenerative payload has in addition to RF filtering, frequency conversion, and amplification also on-board processing capabilities (for demodulation/decoding, switching and/or routing, coding/modulation).

Starting from Rel. 17, three main satellite network solutions have been integrated in the 5G ecosystem: i) an NR-based satellite access network, in which the non-terrestrial nodes are connected to the 5G core (5GC) and support the NR access technology; ii) a long term evolution (LTE) based satellite access network, in which the non-terrestrial nodes are connected to the 4G evolved packet core network (EPC) through NB-IoT and eMTC radio access; and iii) satellite backhaul, with a transport network over satellite providing connectivity between the 5GC and the gNB(s), which can be based on 3GPP or non-3GPP radio protocols.

Focusing on the NR-based satellite access, the system architecture can be based on a transparent payload and an NTN gateway (GW) or a regenerative payload. According to 3GPP releases 17 and 18, the satellite implements a transparent payload and the gNB, logically placed at the gateway, is connected to the user plane function (UPF) and access and mobility function (AMF) in the 5GC for the user plane (UP) and control plane (CP), respectively, through a terrestrial link implementing the next generation (NG) Air Interface. The user equipment (UE) is directly connected to the satellite by means of the user link, while the satellite is connected to the gNB on ground through the feeder link.

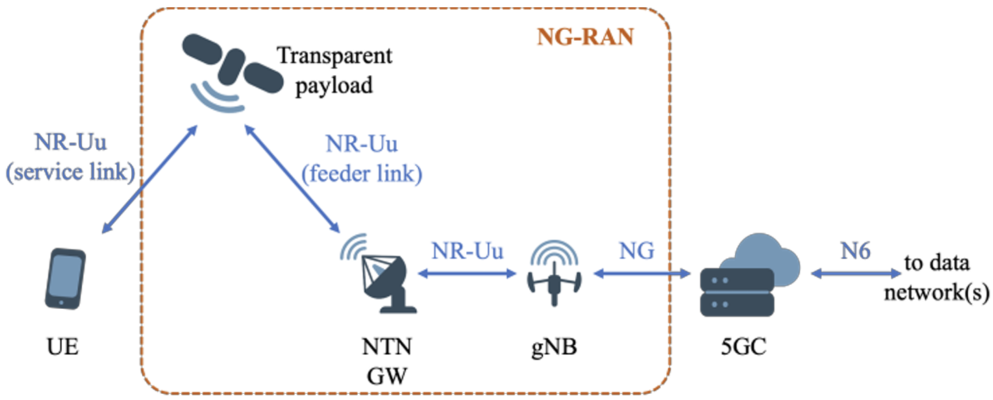


Figure 4-7: NTN architecture with transparent payload and direct access [34]

While a single gNB is shown in Figure 4-7, it is worthwhile mentioning that: i) multiple NTN payloads can be connected to a single gNB; ii) multiple gNBs can be connected to a single NTN payload. In essence, the NTN transparent payload acts as a RF repeater, thus forwarding the radio protocols received from the UE to the gNB and vice versa. The service link can be implemented based on: i) Earth-fixed beams, with on-ground beams continuously covering the same geographical area for the entire time, as with GEO satellites; ii) Quasi Earth-fixed beams, in which the beams cover the same area for a period limited to the NTN node visibility, as for LEO satellites with steerable beams; or iii) Earth-moving beams, in which the beams are non-steerable and continuously cover the area below the NTN payload, moving on the Earth's surface along the NTN node movement on its orbit.

In terms of functional split, since this architecture is based on a transparent payload, the on-ground gNB implements both the centralized unit (CU) and the distributed unit (DU).

3GPP release 18 did not introduce any modifications to the kind of payload within the NTN node, while a significant enhancement has been achieved in release 19, where the regenerative payload is considered.

As mentioned above, for regenerative payloads, radio frequency filtering, frequency conversion and amplification as well as demodulation/decoding, switch and/or routing, coding/modulation are implemented on-board. This is effectively equivalent to having all or part of the gNB protocol stack on the NTN platform.

Figure 4-8: NTN architecture with regenerative payload and direct access, full gNB on-board.

Figure 4-8 shows the architecture with a regenerative payload and the entire gNB implemented on-board. In this case, the NR-Uu protocols are entirely terminated on-board, consequently, the GW basically acts as a transport network layer node, terminating all transport protocols and connecting to the 5GC and the on-board gNB.

The exploitation of regenerative payloads also allows the introduction of Inter-Satellite Links (ISLs), operating in RF or optical frequency bands, which can be either 3GPP or non-3GPP defined. In addition, regenerative payloads also allow to lower the required bandwidth for in-space routing [35]. Other two important applications (introduced in Release 19) for regenerative payloads are: i) the support of UE-Satellite-UE communications, which allows UEs to communicate using the satellite access directly without the need of a ground network (at least for the UP part), thus avoiding large latencies and limited data rates, in addition to reducing the consumption of backhaul resources; and ii) Store and Forward operations, which supports delay-tolerant communications by enabling the 5GS to provide a specific service, i.e., storing and forwarding data, during periods of intermittent or temporarily unavailable satellite connectivity with the ground network.

The architecture with the regenerative payload option allows to significantly reduce the over-the-air latency, since all NR-Uu protocols are dealt with by the regenerative payload on-board; however, it is also more complex and the cost of the satellite, and, thus, of the overall NTN system, is increased when compared to a transparent payload architecture.

The regenerative payload opens the door to the implementation of different functional splits [36]. It shall be noted that the functional split is out of the scope of release 19 and might be be considered in future releases of 3GPP in the process of the 6G technical specification.

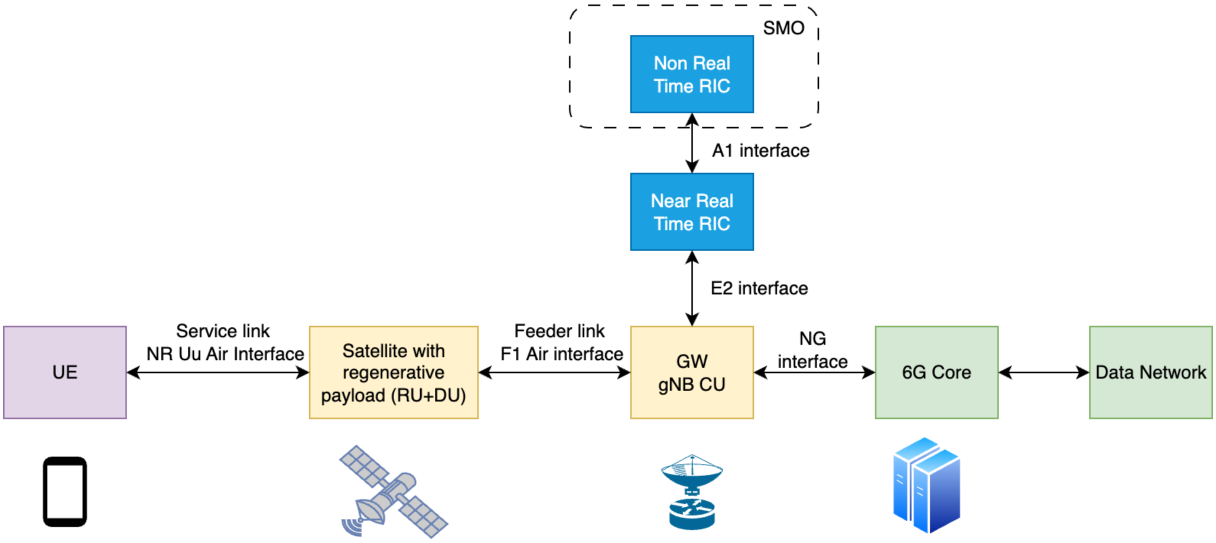


Figure 4-9: Example of NTN domain

Figure 4-9 shows an example NTN domain, where the DU and the RU are implemented on board the satellite and the CU on-ground at the gateway.

- ***Interaction with the Management and Orchestration layer***

An important element of the unification layer in the NTN domain is the Domain Management Orchestrator (DMO), responsible for the management and orchestration of satellite based network resources and as such acting as bridge between the unification and the management and orchestration layers. Such functionality is instrumental to achieving effective operations in the satellite domain such as beamforming, resource allocation, and general management of feeder (satellite to ground gateway communication) and service links (satellite to user equipment's communication), just to cite some of the key ones. Moreover, it can also control onboard data routing mechanisms within the satellite payload to efficiently forward traffic between ground segments and space segments, depending on the specific functional split considered (e.g., full gNB in space) as formerly commented in the previous section

In more detail, the NTN domain supports the management and lifecycle control of network functions, including gNB and 5G core network functions that operate in conjunction with satellite communication infrastructure as described in the previous section. These functions enable seamless integration between terrestrial and non-terrestrial network segments and as such play a very important role in the overall framework of the UNITY-6G architecture

Finally, the NTN domain is operated by using a local closed loop control mechanisms to ensure efficient resource utilization and network performance optimization within the domain. This closed loop control is implemented through components such as Monitoring service (MS) for monitoring and control, DE for policy execution and decision making, and an AE for data analysis and performance evaluation. These components jointly guarantee reliable and efficient operations of the NTN domain within the overall orchestration framework defined within the management and orchestration layer. In this respect, the KPI metrics generated within the NTN domain are continuously monitored and then reported to the IDMO through the cross-domain service-based management bus (SBMA). These KPIs include performance indicators such as link latency, throughput, signal quality, resource utilization and service reliability, which provides a real-time view of the operational state of the NTN domain.

The IDMO employs these KPIs to perform cross domain orchestration and service composition across multiple network domains. It enables the creation of the end-to-end services by coordinating resources from different domains such as AI/ML-driven management systems, transport networks, Xhaul (fronthaul, midhaul and backhaul), Non-public networks (NPN) and Wi-fi access networks. By analysing the collected KPIs and network conditions, IDMO can dynamically determine the most suitable combination of domains required to deliver specific service. Based on use case requirements and service level agreements (SLAs), IDMO decides when and how NTN domain should be integrated into the overall service chain. For example, NTN resources may be included to provide extended coverage in remote areas, service continuity in mobility scenarios or backup connectivity when terrestrial infrastructure is unavailable. Through this process, IDMO ensures that end-to-end service performance meets the required quality, reliability and latency constraints across heterogeneous network environments.

From a satellite operator perspective, the description of the NTN domain may be complemented with some additional operational considerations. While the UNITY-6G architecture already assigns to the NTN Domain Management Orchestrator (DMO) the management of satellite-related resources and the interaction with the Inter-Domain Management and Orchestration (IDMO) layer, it may be useful to note that the operation of the NTN domain is subject to a number of domain-specific conditions. These may include, depending on the implementation scenario, aspects such as beam availability, gateway visibility, feeder link constraints, orbital characteristics, spectrum coordination requirements, and service prioritization policies.

In this context, the interaction between the NTN domain and the upper orchestration layer can be understood as relying on an abstracted view of resources and service capabilities, so that cross-domain service coordination can be supported without requiring full exposure of domain-internal mechanisms. This interpretation is consistent with the role of the DMO as the entity responsible for translating higher level service requests into domain-specific actions and configurations.

From this viewpoint, the satellite domain may therefore be regarded not only as a connectivity enabler, but also as an operational domain with its own control logic, monitoring processes, and resource management constraints. This is particularly relevant in scenarios where NTN resources are incorporated into end-to-end service delivery together with terrestrial segments, since the contribution of the satellite domain may depend on operational factors that are not fully visible at cross-domain level. Clarifying this aspect may help better reflect the practical role of satellite systems within the overall architecture.

4.2.3.3.3 Interfaces

- ***The interface between IDMO and NTN-DMO***
    - The interface between IDMO and DMO-NTN enables coordinated service provisioning across multiple network domains. Through this interface, the IDMO sends a technology agnostic service template to the NTN DMO. This template describes the service level requirements that must be satisfied by the NTN segment without specifying the underlying satellite technology or configuration details.
    - The service template typically includes parameters such as minimum throughput, maximum coverage area, latency constraints, reliability

requirements and service availability targets. These parameters represent the SLA or the operational requirements of the requested service.

- Upon receiving the service template, the NTN DMO translates these high-level requirements into domain specific configurations and resource allocations within satellite networks. This process includes selecting appropriate satellite beams, allocating spectrum resources, configuring feeder and service links and optimizing routing and scheduling policies to meet the specified service objectives. By converting technology agnostic service requirements into concrete NTN resource configurations, the NTN DMO ensures that the satellite network segment can support the requested service while maintaining efficient utilization of available satellite resources.

- ***The interface between NTN-DMO and NTN service***
  - For O-RAN based NTN deployments, the DMO of the NTN domain interacts with several key management and control components to coordinate network functions and infrastructure resources. In particular, the NTN DMO interfaces with the SMO, RIC platforms, SDN controllers and NFVO responsible for managing virtualized functions within the NTN domain.
  - Through these interactions, the NTN DMO enables the deployment and lifecycle management of VNFs such as gNB and core network functions including UPF, AMF etc. These VNFs may be instantiated either in edge cloud infrastructure, ground gateways or satellite supported computing platforms depending on the service requirements. In addition, the NTN DMO coordinates satellite resource management tasks including beam allocation, dynamic beam steering, traffic routing and backhaul path configuration between satellite gateways and terrestrial network segments. The SDN controllers assist in dynamically configuring network paths and transport resources while the SMO and RIC platforms provide policy control, optimization, and AI- driven RAN management capabilities.
  - All these interactions are typically performed through standardized northbound APIs, allowing the NTN DMO to communicate with orchestration and control entities in a technology agnostic and interoperable manner. These interfaces enable efficient coordination between satellite and terrestrial network components and support automated service deployments and optimization

across the NTN domain. The overall interaction architecture and functional relationships among these components are illustrated in figure 4.9.

- ***Interfaces on the access link***
  - When the satellite payload is transparent or regenerative the user links are implemented via the NR-Uu Air Interface.
- ***Interfaces on the feeder link***
  - In the case of a transparent payload, the feeder link, i.e., the link between the satellite and the gateway, transports the NR-Uu air interface
  - In the case of a regenerative payload and a full gNB on board, the feeder link transports the NG interface. The air interface between the gNB (on the satellite) and the gateway is logical, i.e., it can be implemented by means of any Satellite Radio Interface (SRI), as, for instance, the DVB-S2, DVB-S2X, or DVB-RCS2.
  - In the case of a regenerative payload and a distributed gNB, the air interface on the feeder link depends on gNB split. For example, referring to the split represented in Figure 4-9, i.e., RU and DU on the satellite and CU on ground, the feeder link carries the F2 air interface.

4.2.3.4 Transport Domain

4.2.3.4.1 Overview

The transport domain manages all wired and wireless infrastructure that enables inter-domain connectivity. This includes transport segments connecting the RAN, CN, NTN, and NPN. Management responsibilities within this domain are divided into two primary roles: the Network Slice Controller (NSC) acts as the DMO, handling slice lifecycle and connectivity orchestration, while the SDN controller assumes the IDM role, managing the underlying transport infrastructure. As an extension, the same approach could be applied for intra-domain connectivity, but this is currently out of the scope of this document.

4.2.3.4.2 Architecture

The NSC, serving as the DMO, is responsible for the end-to-end lifecycle management of transport network slices. It processes slice requests, enriches them with contextual information, determines the appropriate transport technology (e.g., L2VPN, L3VPN), and oversees their realization and operational control. In this context, the NSC serves as the decision engine for the transport domain. Its internal structure is composed of four key components:

- Northbound Interface Component, which receives transport-related slice requests, such as IETF Network Slice Service defined in RFC 9543 [37].
- Mapper, which augments each request with contextual details needed for successful slice provisioning.
- Realizer, which selects the implementation technology and translates the slice request into SDN-level instructions.
- Database, which stores state information for all deployed slices.

The SDN controller, acting as the IDM, manages device-level configuration and oversight of the transport network, for example through a TeraflowSDN [38] controller. Within this architecture, the SDN controller effectively serves as the actuator, executing decisions made by the NSC.

It includes:

- A translation module that converts NSC instructions into specific device configurations.
- A telemetry module functioning as the monitoring system, collecting link and node status.
- A topology database that maintains information about transport elements and their configurations.

In addition, the transport domain is expected to be supported by digital twin technologies. These can assume the role of an analytics engine by predicting optimal connectivity paths while ensuring that network slice requirements are met.

Figure 4-10 depicts the whole architecture of the domain.

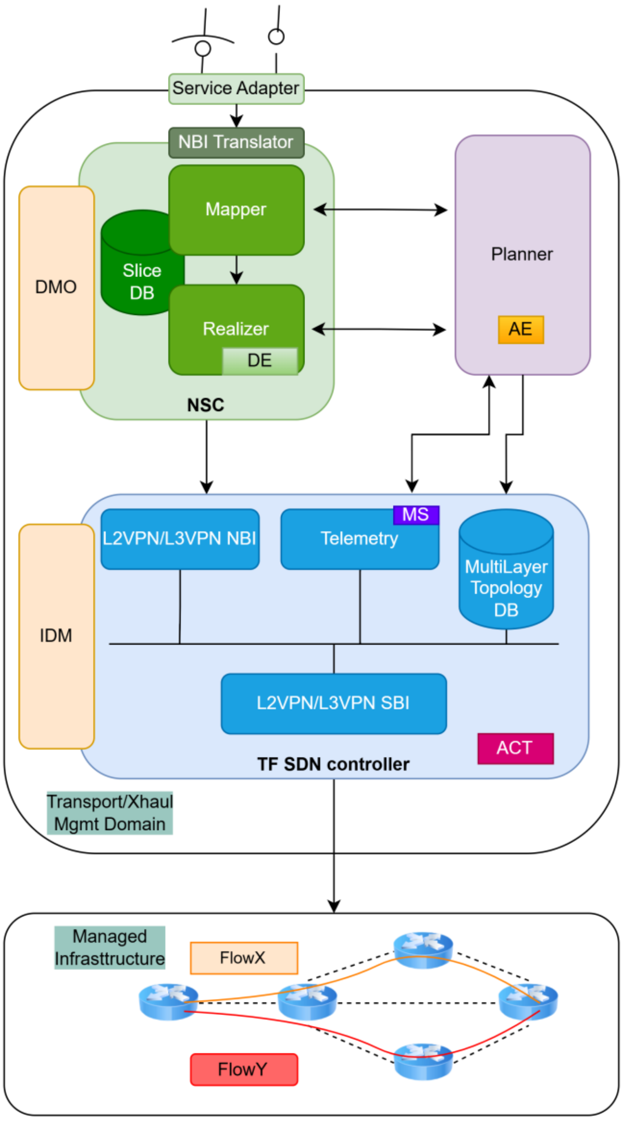


Figure 4-10. Transport Management Domain Architecture

4.2.3.4.3 Interfaces

The transport domain architecture relies on well-defined interfaces between its components to ensure consistent end-to-end orchestration:

- The northbound interface of the NSC receives high-level slice requests and adapts them into standardized formats such as the IETF Network Slice Service (RFC 9543).
- The NSC–SDN controller interface serves as the primary control link through which the NSC conveys implementation instructions to the SDN controller's configuration module.
- The telemetry interface between transport devices and the SDN controller provides real-time monitoring data, enabling the controller to maintain accurate topology knowledge and support both assurance and closed-loop operations.

- The digital twin interface allows DT components to ingest topology, telemetry, and slice requirement information, enabling predictive analytics and assisting the NSC in validating slice feasibility.

Together, these interfaces ensure seamless coordination between orchestration, infrastructure management, monitoring, and analytics within the transport domain.

4.2.3.5 Vertical App Domain

4.2.3.5.1 Overview

A Vertical Application Domain provides management and orchestration capabilities for the deployment and operation of vertical applications, e.g., extended Reality (XR) instances that consume and interact with the mobile network instances composed by the interconnected components deployed across domains instances of the types explained previously, e.g., RAN, mobile core, transport. The vertical application instances are tightly coupled with the 5G/B5G network, typically exchanging user-plane traffic via the UPF and relying on network-provided capabilities such as connectivity, latency guarantees, and traffic steering.

4.2.3.5.2 Architecture

Similarly to the other domains, a vertical app domain exposes its capabilities through the cross-domain service-based management bus, thus being part of end-to-end orchestration coordinated by the IDMO across the edge-cloud continuum to perform flexible, scalable, and performance-aware deployment of vertical services and enable advance vertical use cases in 5G/B5G environments. The Vertical Application Domain is structured around a Service adapter, a DMO instance connected to the Service adapter sub-layer, interacting with the registered IDMs instances under its control. The concrete realization of the Vertical Application Domain entities (i.e., DMO, IDM) may vary depending on the deployment environment. For example, the DMO may be realized using an ETSI NFV Network Functions Virtualisation Orchestrator (NFVO) or an application-level service orchestrator, such as the one described in Section 4.1.3.1. With regards or IDM implementation, the preferred option is the use of Kubernetes cluster instances, as mentioned in Section 3.3.3. as depicted in Figure 4-11. These realizations are considered implementation choices and do not constrain the logical architecture.

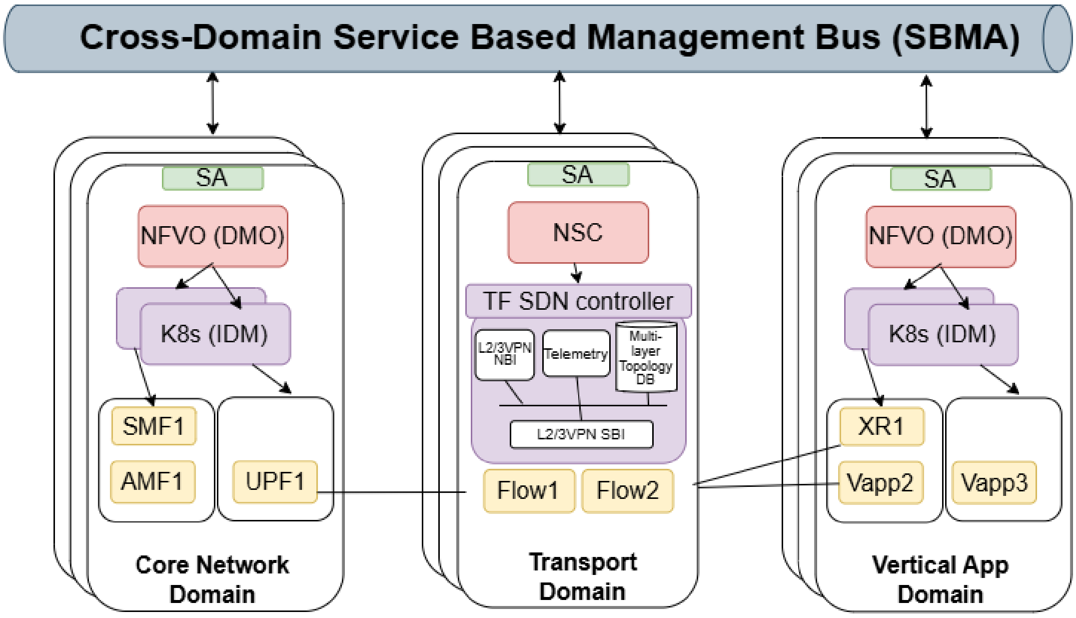


Figure 4-11 Vertical App Domain Architecture

4.2.3.5.3 Interfaces

The service adapter, after processing and enriching the received request from the cross-domain service management bus contacts the northbound interface offered by the DMO in the form of REST-API based endpoints.

It is worth mentioning that the deployment of Vertical application may enable additional interfaces to interact with the deployment, like the use of a Graphical User interface. This will depend on the characteristics of the artefact, and it will be returned from the DMO to the IDMO through service adapter and cross-domain bus to be exposed to the vertical application requester.

4.2.3.6 AI/ML Domain

4.2.3.6.1 Overview

The AI/ML subsystem is considered a separate domain that targets to provide intelligent services to the multiple UNITY-6G domains, aligning with the concept of AI-native architecture. The following functional design principles are identified for the overall system: (i) The subsystem shall provide AI/ML as a service (AIaaS) to the 6G network architecture in the form of interconnected tuples, i.e., training, inference and evaluation pipelines; (ii) The intelligent services shall be available in all UNITY-6G domains and at multiple locations, for instance in the RAN, Core and Management domains; (iii) The ML pipelines shall be either deployed centrally or in a distributed manner (e.g., ML models can be deployed at the location of the data), while supporting model management lifecycle, including data gathering and analysis, model training, testing, validation and performance monitoring; (iv) The framework shall be able to gather data for model training from multiple domains of the 6G system.

4.2.3.6.2 Architecture

In this context, the internal AI/ML subsystem designed for the UNITY-6G architecture is shown in Figure 4-12. Its components include, among others, an AI/ML Orchestrator that is used to interface with the rest of the domains, also coordinating the lifecycle management of intelligent services. In addition, a Data Management (DM) module is responsible for gathering data from the relevant network functions, as well as chaining the tuple components for building pipelines in the AI/ML Pipeline Subsystem. Other components of the system include typical MLOps functionalities (data processing, training services, model database, model serving, several libraries that can be utilized for the model training, etc.) that can be provided by several open-source platforms[7,8,9] and an MLOps orchestrator to dynamically allocate machine learning infrastructure resources. In addition, a Sandbox simulation test environment is included in the architecture that can used to evaluate the performance of the ML models, acting like a playground environment either for trained models (e.g.., testing the performance of different model versions) or simulated environments that can be used to train reinforcement learning (RL) algorithms. Finally, a Conflict Resolution module is also incorporated to the system, targeting to resolve any intra-domain conflicts (e.g., related to utilization of resources for MLOps), as well as link to the hierarchically higher Conflict Manager component that resides in the management level. Finally, the architecture includes a Trust Adapter (connecting to the trust layer) and a Data Adapter (linking the framework to the data continuum).

[7] https://www.kubeflow.org/

[8] https://mlflow.org/

[9] https://metaflow.org/

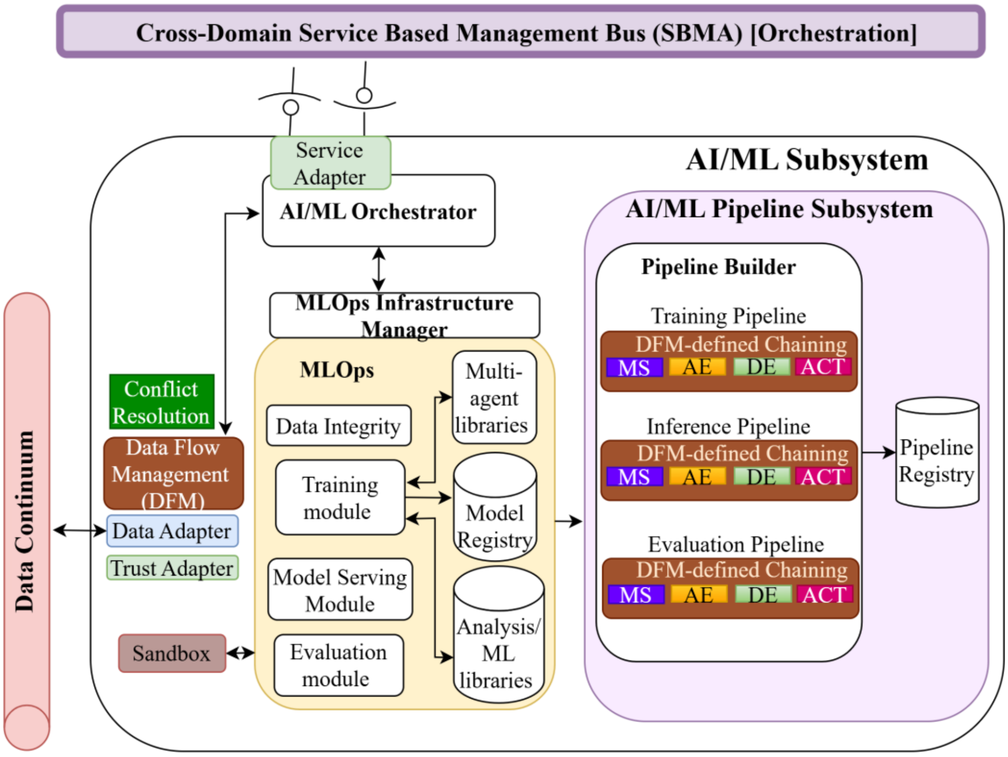


Figure 4-12. AI/ML Domain Architecture

The AI/ML subsystem is used to develop AI/ML pipelines in terms of connected tuples (MS, AE, DE, ACT) for three main purposes: (i) training pipeline, linking the functional tuple building blocks to achieve all the required operations for the training of a ML model. This pipeline is extended to also incorporate distributed learning mechanisms, such as federated learning and multi-agent reinforcement learning; (ii) inference pipeline, serving an already trained model to an inference host to be used for online operation; (iii) evaluation pipeline, monitoring the online inference of a model and assessing its performance. This pipeline may trigger a re-train process in case of model degradation detection.

The training pipeline for an O-RAN control loop provided by the AI/ML subsystem is shown in Figure 415, illustrating the functional tuple components. The process is initialized by an ML intent, i.e., a request that includes all the relevant information for performing the training of an ML model (optimization objective, required training data and algorithm, etc.). The AI/ML Orchestrator acknowledges this request and invokes the Data Management component to gather the raw data (corresponding to the MS component, for instance a radio unit) and the requested AI/ML algorithm (ML code and base model) that will be used to perform the model training process. The raw data are then forwarded to the AE block (can be hosted in the Near-RT RIC) that includes fundamental data integrity and pre-processing operations, along with data generative methods provided by the Sandbox to enhance the training dataset. In addition, the AE block prepares and initiates the training procedure (data loading, pre-processing, training, testing/validation). Then, the DE component (also hosted in the Near-RT RIC) is responsible for exploring the training hyper-parameters and deciding upon the completion of the process. The trained model is stored centrally in a model registry database but can be also stored locally (at the location of the inference host as an xApp).

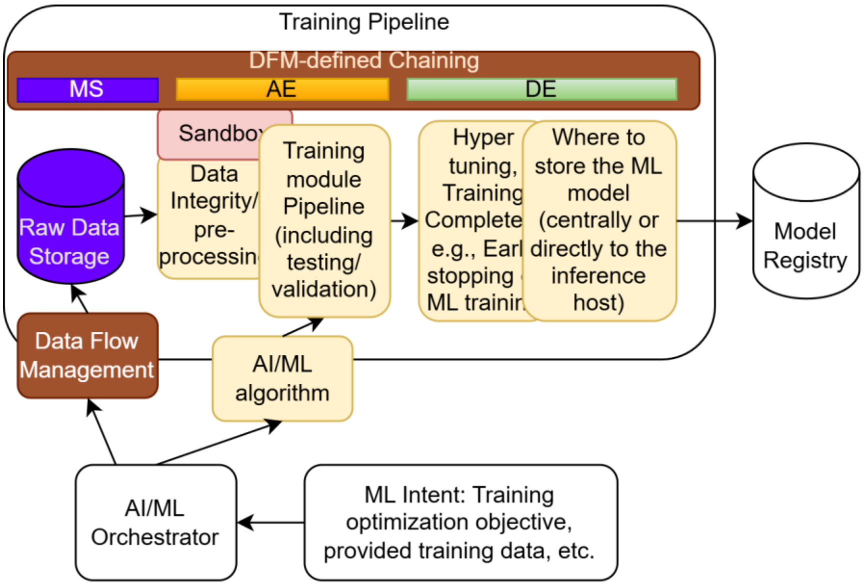


Figure 4-13. Functional view of the training pipeline

In the inference pipeline, the DM is responsible for: (i) pointing out where the online data can be found for inferring the model; (ii) properly configuring the tuples and the chaining of the subcomponents (MS, AE, DE, ACT) of the inference pipeline.

The Intent Manager in the inference pipeline is responsible for: (i) defining the optimization objective (e.g., delay minimization, optimization of network resources, energy efficiency maximization, forecasting of energy usage, etc.); (ii) providing the details for selecting the pre-trained model for deployment among different models in the ML catalogue; (iii) specifications of the inference, e.g., the required online data, etc.

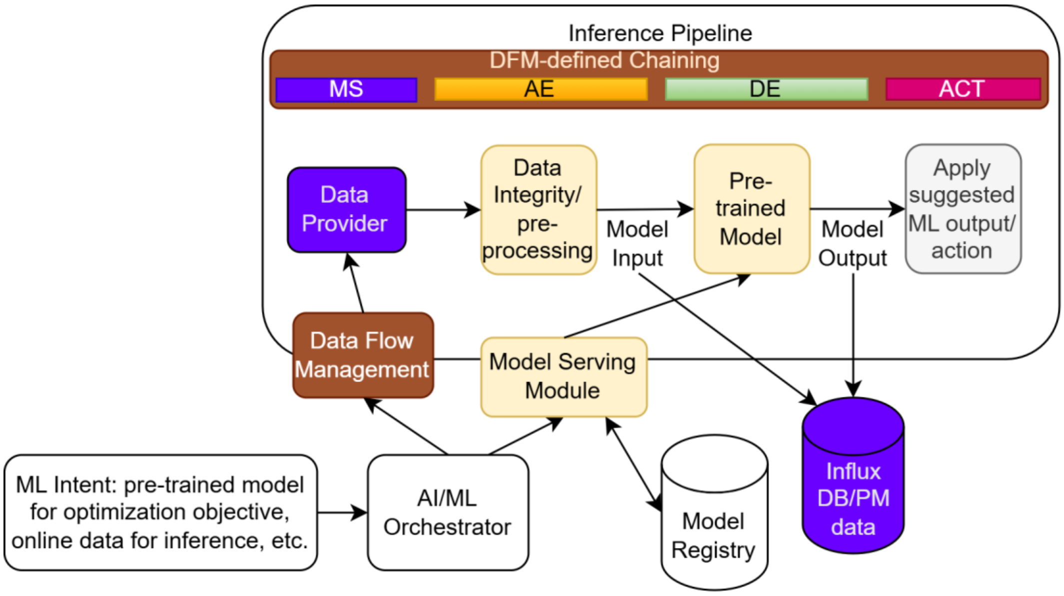


Figure 4-14. Functional view of the inference pipeline

Regarding the evaluation pipeline, the performance monitoring is initialized based on the defined intent (here we assume that the PM data have been gathered for accuracy of the ML model during the online inference phase). The DM in the evaluation process is responsible for: (i) pointing out where the PM data can be found and gathered; (ii) properly configuring the tuples and the chaining of the subcomponents (MS, AE, DE, ACT) of the evaluation pipeline. The Intent Manager is responsible for: (i) providing the specifications of the performance monitoring of a running model (e.g., accuracy of the model, latency or defined KPIs).

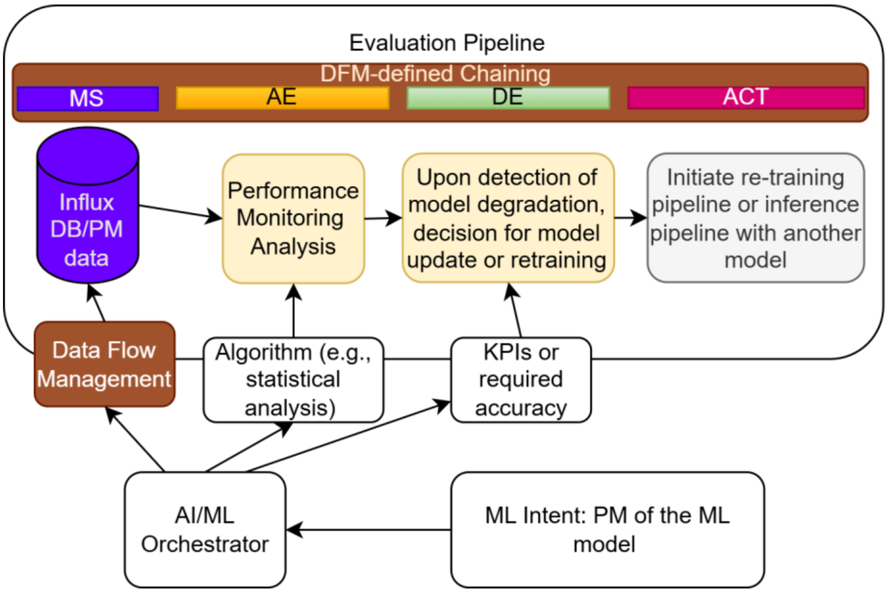


Figure 4-15. Functional view of the evaluation pipeline

4.2.3.6.3 Interfaces

The AI/ML subsystem includes three main interaction adapters to communicate with the different domains and layers of UNITY-6G. The first is the link to the cross-domain service-based management bus through the Service Adapter, which ensures that the requests for the use of the AI/ML services (e.g., in the form of ML intents that sent through a message broker using pub/sub mechanism or REST API) are conveyed to the AI/ML Orchestrator. Within the AI/ML subsystem, the Orchestrator interacts with the MLOps platform and the infrastructure orchestrator for initiating the training/testing/evaluation sequences, as well as with the Data Flow Manager to construct the described training, inference and evaluation pipelines, linking the relevant components for pipeline deployment. In addition, the subsystem includes a Data adapter that will be used to transfer data (mainly training data in case of centralized training or evaluation data) from other domains through the data continuum. This adapter can be also implemented via pub/sub mechanisms to enable the initiation of data streams towards the AI/ML subsystem. Finally, the Trust adapter is included to exchange information via the Trust layer with the other domains, sending and receiving identity data and attestation results to verify the authenticity of the communicating entities.

4.2.3.7 Non-3GPP RAN domain

4.2.3.7.1 Overview

This domain will deal with orchestration of non-3GPP RAN domain for the multi-radio access technology (RAT) purpose. In non-public deployment (NPN) deployment IEEE 802.11 is used to serve users in different verticals as another RAT to cellular. Additionally, to support end-to-end time sensitive applications the new introduced RAT (in this case IEEE 802.11) needs to provide features that ensure accurate time synchronization, traffic differentiation and traffic scheduling. All these features need to be managed and orchestrated amongst IEEE 802.11 APs and the rest of networks.

4.2.3.7.2 Architecture

Figure 4.18 shows the building blocks of the non-3GPP domain architecture. At its core, the non-3GPP domain consists of the DMO, which interfaces northbound with the IDMO while orchestrating the operation of the IEEE 802.11 AP controller, as well as the Central Network Controller (CNC) and Central User Controller (CUC) needed for the TSN features. The DMO hosts also the application requirements data base as well as telemetry database. The application requirements database will host the time sensitive traffic flow profiles and requirements that network need to fulfil, while the telemetry database will host the monitoring information. A domain digital twin (DDT) entity is expected to perform automatic scheduling based on application requirements from the CUC, real-time telemetry data from the CNC telemetry database and real-time topology monitoring database. Scheduler in the IDM will take decisions on scheduling for IEEE 802.11 APs, as well as will apply such decision to the underlying infrastructure. Architecture functional specifications are assigned as follows: DDT is responsible for decision and analytic engine functions; CNC and the AP controller are responsible for analytics, action, and monitoring engine functions; and network nodes are responsible for action engine functions.

4.2.3.7.3 Interfaces

Regarding the interfaces a Domain Data Fabric (DDF) is foreseen to enable the access to the databases in the DMO. Both CUC and CNC will write data to application requirements and telemetry databases, respectively. AP controller will use those data from the databases for decision purposes, in its DE. DDF can be a broker-less pub/sub interface, REST API or event driven interface. The non-3GPP DMO, in its northbound interface, will interact with the SMO of the O-RAN DMO via cross-domain SBMA.

In the southbound direction, DMO interacts with IDM. Further, IDM interacts with stations and IEEE 802.11 APs using a unified approach that follows O-RAN principles. The interface between IDM and underlying infrastructure will utilize E2 interface of O-RAN framework. All the IEEE 802.11 APs will become O-RAN compliant, terminating the E2 O-RAN interface for exposing telemetry data as well as for receiving control instructions from IDM.

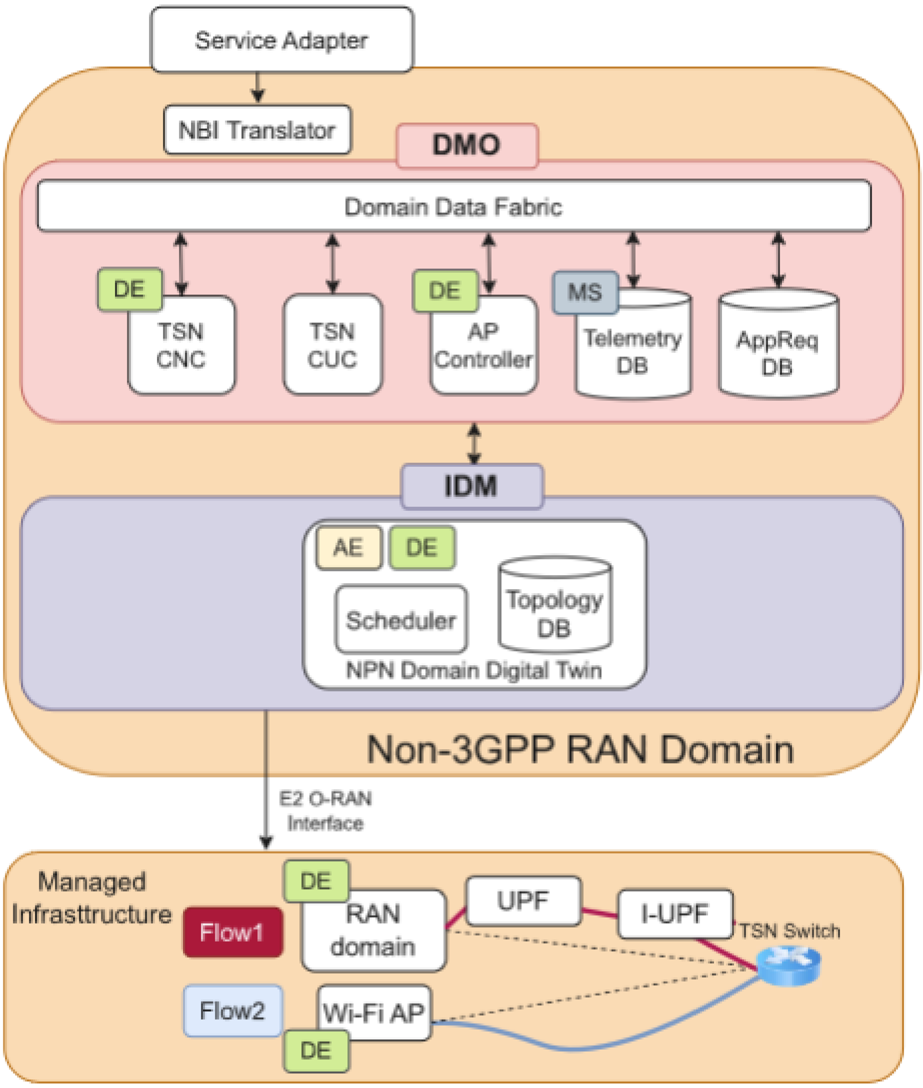


Figure 4-16. Non-3GPP RAN Domain

4.2.3.8 Digital Twin

4.2.3.8.1 Overview

The Digital Twin (DT) framework is considered as a subsystem that targets to resolve the fragmentation and static nature of current network orchestration, providing a unified platform for cross-domain intelligence. As a primary functional capability, the subsystem acts as a real-time virtual replica of the integrated 6G infrastructure to abstract domain heterogeneity. The DT addresses the fundamental requirement of network sustainability by enabling energy-aware decisions and environmentally optimized service management. To support this, the framework incorporates specific analytical models depending on the domains and services to evaluate system energy efficiency. Finally, the platform allows operators utilizing the UNITY-6G architecture to execute simulative "what-if" scenarios, evaluating real-time resource allocation strategies and energy consumption impacts, such as assessing the activation of a new transport network path or changes to the set of active base stations.

4.2.3.8.2 Architecture

Rather than functioning as a single, monolithic entity, the Digital Twin (DT) implementation relies on a "Composed Digital Twin" architecture managed by a DT Models Manager. This manager oversees multiple, specific DT models tailored to different functions (e.g. load modeling for 5G and Wi-Fi, or power consumption prediction) which act as precisely defined mathematical functions. These specific models can take various forms, including file-based look-up tables, Machine Learning models (like Graph Neural Networks), or network simulators and emulators (such as NS-3 or srsRAN).

Figure 4-17 outlines the internal components and external interfaces of the Digital Twin Service. At its core, the architecture is designed to bridge external management applications with the real-time operational state of the physical network. This is achieved through a structured environment that ingests live data, manages analytical models, and processes complex simulative requests, ultimately supporting the continuous optimization and evaluation of modern telecommunications infrastructure.

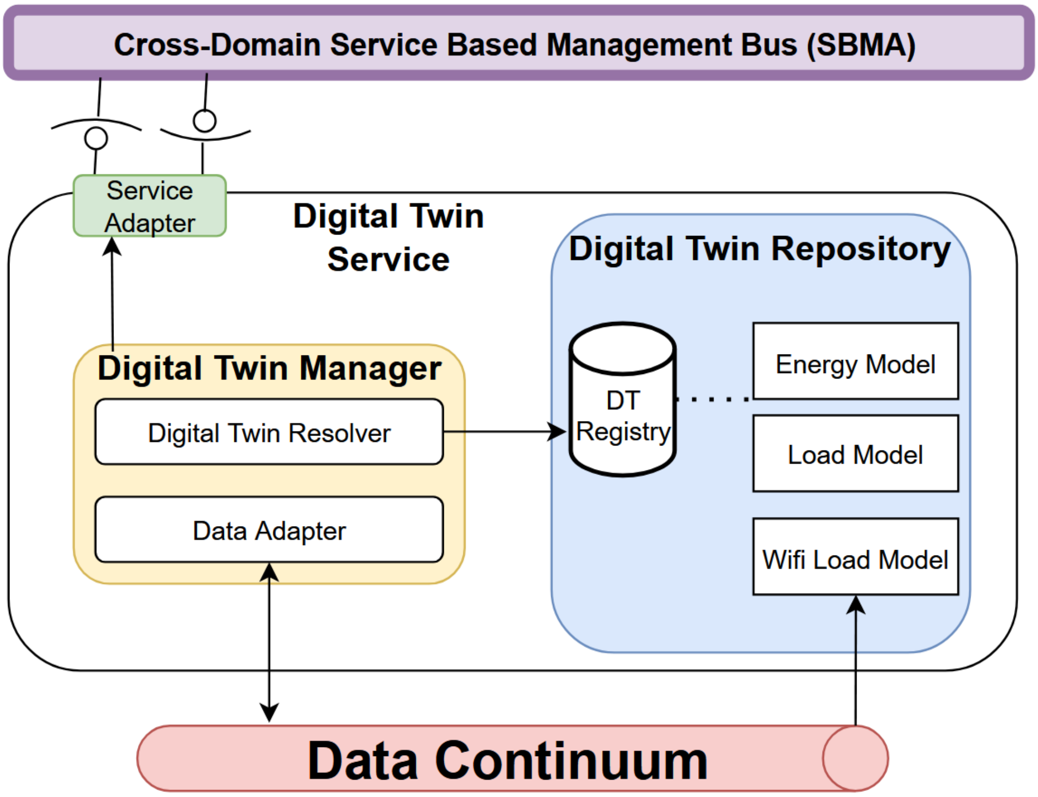


Figure 4-17. Digital Twin Domain Architecture

The system interacts with external environments through two primary boundaries: the overarching management layer and the underlying data source. At the top, the Cross-Domain Service Based Management Bus (SBMA) allows external network applications to request services, execute "what-if" scenarios, and extract cross-domain insights. To facilitate these interactions, a Service Adapter acts as an API gateway. It receives requests via standard interfaces and translates them into actionable internal commands for the DT Service. At the foundation lies the Data Continuum, which represents the continuous, real-time stream of telemetry, historical metrics, and network events extracted directly from the live physical infrastructure, serving as the lifeblood for network synchronization.

Within the domain, the Digital Twin Manager functions as the orchestration engine responsible for coordinating all internal operations. It contains two primary sub-components: the Digital Twin Resolver and the Data Adapter. When an external request is processed by the Service Adapter, the Digital Twin Resolver determines the operational requirements and queries the internal registry to locate the appropriate analytical models needed to execute the requested simulation. Simultaneously, the Data Adapter manages the data ingestion pipeline. It acts as the bridge to the Data Continuum, pulling in raw telemetry and formatting it to ensure the virtual models remain accurately synchronized with the real-world network state. The development of this architecture will be reported in future deliverables.

4.2.3.8.3 Interfaces

The Digital Twin (DT) subsystem interfaces with the UNITY-6G architecture through defined integration points that connect its virtual environment to the live network and management layers. Cross-Domain Service Based Management Bus (SBMA) is facilitated by the Service Adapter, which ensures that external management applications can seamlessly invoke DT services, such as executing simulative "what-if" scenarios or energy optimization evaluations. These interactions are conveyed to the internal Digital Twin Manager using standard communication protocols, such as REST APIs or publish/subscribe mechanisms via a message broker, prompting the Digital Twin Resolver to interact with the Digital Twin Repository to locate and instantiate the required analytical models (e.g., Energy Model, Load Model) from the DT Registry. Southbound data ingestion and synchronization are achieved by interfacing directly with the Data Continuum. This connection is primarily managed by the Data Adapter, which extracts real-time operational state, telemetry, and historical data from the physical network domains to ensure the virtual models remain continuously and accurately synchronized with the physical infrastructure. Furthermore, the DT architecture establishes specialized data interfaces where specific computational models, such as the Wifi Load Model, can connect straight to the Data Continuum, bypassing the central Data Adapter to establish low-latency data streams necessary for rapid telemetry ingestion, real-time forecasting, and continuous model learning.

4.2.4 Interfaces

The Unification Layer interacts with the rest of the UNITY-6G architecture through three main adapters, each ensuring consistent communication across layers. First, every domain has a Service Adapter, which provides the service-based interface towards the IDMO and enables the exchange of orchestration messages over the cross-domain service based management bus. Through this adapter, each domain announces its capabilities, publishes its service catalogue, reports resource availability, and receives orchestration actions such as instantiation, configuration or termination. Additionally, each domain integrates a Data Adapter, which links domain-level functions to the Data Continuum. Finally, each domain includes a Trust Adapter, which enforces the UNITY-6G trust model within the Unification Layer. The adapter exchanges identity information, attestation results, trust-related evidence, and LoT constraints with the Trust Layer, ensuring that domain-level operations comply with global trust policies.

4.3 DATA CONTINUUM

The Data Continuum in UNITY-6G, powered by the aeriOS framework, provides the architectural foundation for a decentralized Meta-OS that spans resource-constrained IoT devices, edge nodes, and cloud datacenters. It is able to act as the "connective tissue" that enables intelligent service deployment and lifecycle management across the heterogeneous 6G domains defined in the unification layer.

4.3.1 Objective

The primary objective of the Data Continuum is to provide a production-grade, 6G-aware runtime environment that abstracts underlying infrastructure heterogeneity. It serves as a transparent execution substrate that bridges high-level architectural intents with physical infrastructure, facilitating the operationalization of the Management and Orchestration Layer's commands. The architecture aims to enable intent-driven orchestration by providing the real-time, trustworthy continuum awareness required for 6G network slicing and service placement. By creating a common resource model for all Infrastructure Elements (IEs), regardless of whether they are Kubernetes nodes, bare-metal servers, or IoT gateways, it unifies resource abstraction across the entire network. Furthermore, it potentially facilitates the placement of candidate AI/ML components and services (e.g., MS–AE–DE–ACT tuples) close to the data source to meet strict 6G latency requirements. Ultimately, it provides infrastructure transparency through a normalized view of heterogeneous partner resources, allowing the IDMO/DMO to perceive the distributed continuum as a single, contiguous resource pool while feeding granular telemetry to enable AI-native closed loops.

4.3.2 Key Functions

The Data Continuum acts as the "connective tissue" that enables intelligent service deployment and lifecycle management across heterogeneous 6G domains. Key functions include federated multi-domain orchestration, which allows independent partner testbeds like NCSRD, IMEC, and CTTC to operate autonomous local domains within a unified UNITY-6G continuum. It utilizes Next Generation Service Interface – Linked Data (NGSI-LD) and context brokers to provide continuum-wide visibility of resources, AI agents, and 6G telemetry, such as O-RAN RIC xApp performance. Trustworthy data provenance is achieved by integrating with the UNITY-6G Trust Layer to provide immutable audit trails and trust evidence. The continuum also supports sustainability optimization by minimizing data movement and exploiting local edge resources. Additionally, it acts as the local enforcer of orchestration policies, ensuring that containerized services are maintained and scaled according to end-to-end service lifecycle requirements. Real-time event propagation and self-orchestration enable autonomous adaptation; when an IE detects overload, it emits an NGSI-LD event that prompts the High-Level Orchestrator (HLO) to recompute optimal placement and trigger live migrations without service interruption.

- Federated Multi-Domain Orchestration: Allowing independent partner testbeds (e.g., NCSRD, IMEC, CTTC) to operate autonomous local domains while participating in a unified UNITY-6G continuum.

- Semantic Discoverability: Utilizing NGSI-LD and context brokers to provide continuum-wide visibility of resources, AI agents, and 6G telemetry (e.g., O-RAN RIC xApp performance).

- Trustworthy Data Provenance: Integrating with the UNITY-6G Trust Layer to provide immutable audit trails and trust evidence.

- Sustainability Optimization: Supporting energy-aware placement policies by minimizing data movement and exploiting local edge resources.

4.3.3 Components

The aeriOS Data Continuum is a multifaceted architecture integrating three orthogonal fabrics, Network & Compute, Data, and Service, managed by a hierarchical two-level orchestration model. The fundamental compute unit is the Infrastructure Element (IE), which encompasses any container-capable device, such as Kubernetes nodes, Raspberry Pis, VMs, or Single Board Computers (SBCs), that exposes standardized monitoring/control APIs and semantic self-description. At the physical layer, IEs expose standardized telemetry including flavor, resource utilization, geolocation, container runtime status, and trust levels. Orchestration is led by the High-Level Orchestrator (HLO), a Python-based component acting as the continuum "brain" responsible for global resource decisions, policy enforcement, and translating Topology and Orchestration Specification for Cloud Applications (TOSCA) blueprints into executable implementation plans. The HLO employs reinforcement learning pipelines that ingest continuum state (IE telemetry, service SLAs, network topology) to generate optimal placement strategies for complex 6G scenarios like network slicing co-placement with O-RAN RIC xApps/rApps. Execution is carried out by the Low-Level Orchestrator (LLO) Kubernetes operators, which reside on IEs to handle blueprint deployment, local optimization, health monitoring, and telemetry streaming via adapters that ensure portability across Docker, KubeEdge, and native container systems.

In Figure 4-18, aeriOS domain can be presented inclding both data, network and computing fabric.

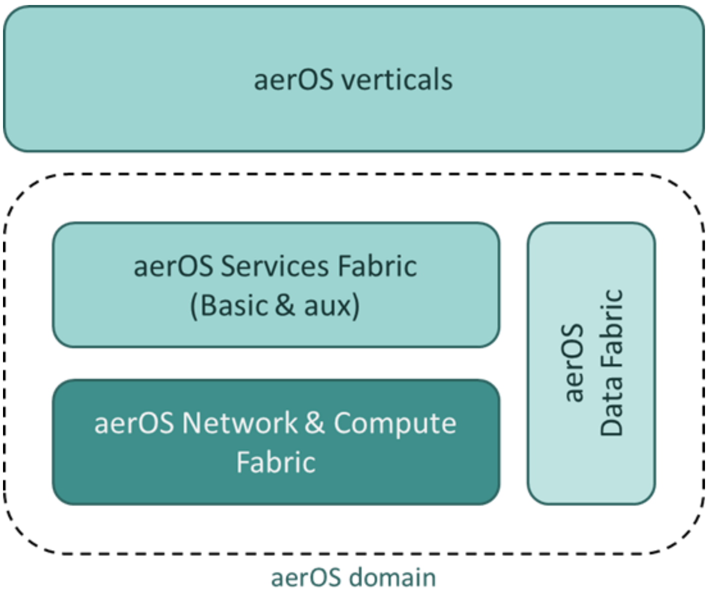


Figure 4-18. aeriOS Domain basic structure

The Data Continuum's core data management is built around the Distributed State Network of Brokers (DSNB), a federation of NGSI-LD Orion-LD context brokers that eliminates data silos while preserving domain sovereignty and privacy. These brokers federate through a multi-hop subscription model that propagates context updates without centralized storage, creating a navigable knowledge graph for resource discovery. The Network & Compute Fabric provides the physical/virtual infrastructure abstraction layer and supports cross-runtime workload migration and hardware acceleration (GPU/TPU). The Data Fabric establishes semantic federation via context sharing, semantic annotation, and ontology mapping, utilizing a three-tier semantic hierarchy that maps physical infrastructure to orchestration abstractions. This fabric includes a Semantic Annotation Engine for enriching raw telemetry and Semantic Translators to mediate ontology mismatches. The Service Fabric manages microservice lifecycles through a five-phase intent-driven workflow, Intent Submission, Global Planning, Blueprint Generation, Distributed Execution, and Closed-Loop Monitoring, enabling autonomous self-healing and reallocation on SLA violations.

Operational management is handled by the Management and Federation Layer, featuring an entrypoint-only Management Portal for blueprint design and visualization, and the aeriOS Federator for domain bootstrapping, peer discovery via Friend of a Friend (FOAF) triples, and liveness monitoring. To ensure production-grade performance, architectural decisions like the HLO/LLO split enable global optimality with local efficiency, while scalability targets include thousands of IEs and 10 million events per second through asynchronous processing and ML model caching. All orchestration decisions are logged as tamper-proof transactions within the Distributed Ledger Technology (DLT) framework to provide verifiable evidence for Level of Trust (LoT) computation. Finally, aerOS is packaged as modular Helm charts, comprising the Runtime Layer (LLO), Basic Services (Data Fabric and Federator), and Auxiliary Services (HLO and Management Portal), with automated bootstrap scripts for seamless domain registration and peer discovery.

4.3.4 Interfaces

The Data Continuum utilizes four primary interfaces to integrate with the UNITY-6G architecture. The Northbound Interface, Southbound, East-West and Data Fabric. Specifically:

- Northbound Interface (Management Plane Integration): This interface connects the aeriOS High-Level Orchestrator (HLO) with the UNITY-6G Intent-Driven Management and Orchestration (IDMO). It allows the IDMO to treat the entire NCSRD testbed (or any aeriOS-enabled site) as a single, programmable continuum. Through this interface, aeriOS receives service descriptors and provides "Continuum Awareness" telemetry, enabling the global orchestrator to make informed placement decisions.

- Southbound Interface (Infrastructure Abstraction): This is the operational interface between aeriOS and the physical/virtual resources. It leverages the Low-Level Orchestrator (LLO) to manage the lifecycle of 6G workloads. By abstracting the differences between a cloud server and an edge gateway, this interface ensures that UNITY-6G services can be deployed consistently across the continuum.

- East-West Interface (Domain Federation): Crucial for the multi-domain nature of UNITY-6G, this interface allows different aeriOS instances to "peer" with each other. Using a Distributed State Network of Brokers (DSNB), it enables the discovery of resources in a partner domain. For example, if NCSRD requires extra compute capacity for a UNITY-6G use case, it can discover and utilize IMEC or CTTC resources through this secure, NGSI-LD based federation.

- Data Fabric Interface (Cross-Domain Unification): This interface links the aeriOS control logic with the UNITY-6G Unification Layer. It specifically interacts with the project's metadata repositories and context brokers to ensure that data flows, such as those required for AI model training or O-RAN telemetry, are routed through the most efficient path in the continuum.

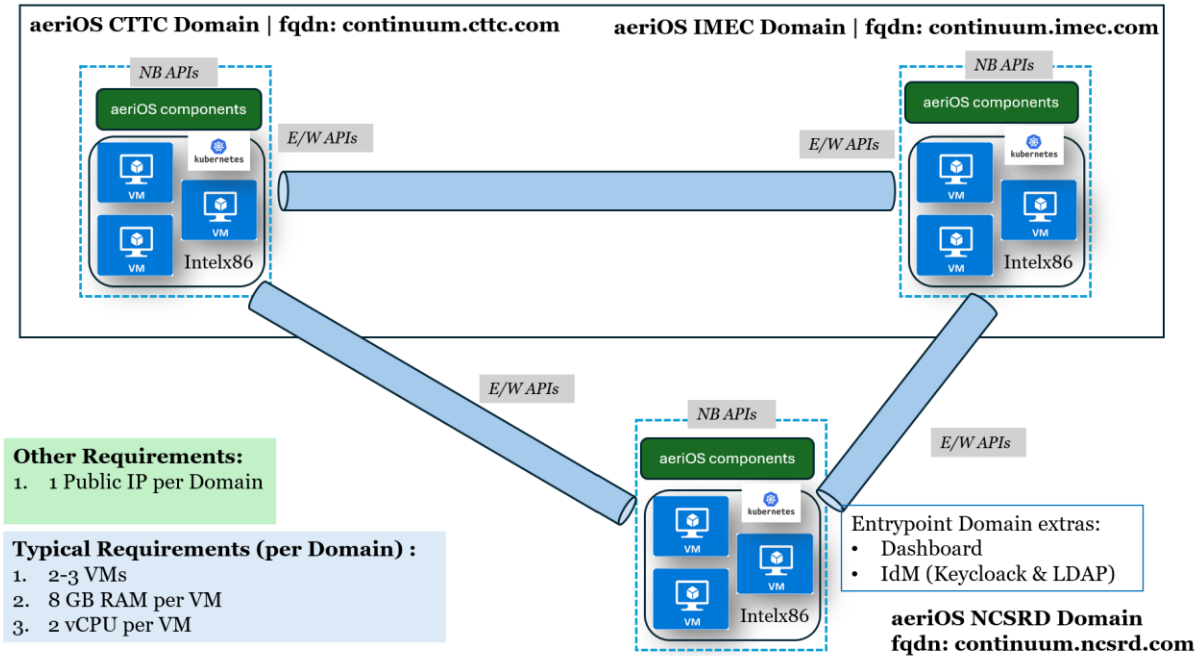


Figure 4-19. aeriOS-enabled multi-domain UNITY-6G continuum

In this diagram (Figure 4-19) IMEC and CTTC deploy independent aeros domains, interconnected via aerOS (edge orchestration platform) using E/W APIs for cross-domain collaboration. NCSRD contributes dashboard, admin tools, Keycloak, and LDAP services via fgdn:continuum.ncsrd.gr and aerosNCSRD LDAP, enabling a data continuum for discoverability and loading across all UNITY-6G nodes and services. Requirements include 1 public IP/domain, 2 VMs per domain, and 3 VNFs, with entrypoints exposing domains for seamless interoperability.

4.4 NETWORK EXPOSURE FRAMEWORK

4.4.1 Objective

The UNITY-6G architecture includes a Network Exposure Framework designed to provide a unified, secure and interoperable interface between the integrated 6G network infrastructure and a diverse set of application functions, including vertical applications, service orchestrators, network controllers and digital twin platforms.

Acting as a centralized exposure layer, the network exposure framework enables secure and standardized access to monitoring data and control mechanisms across multiple computing and network domains (RAN, transport, core). By exposing network information and fine-grained control mechanisms, it facilitates data-driven, AI-native optimization for service placement, resource management, and end-to-end QoS/QoE satisfaction.

The framework therefore serves as a key enabler for the UNITY-6G proof-of-concepts (PoCs), including XR/holographic streaming, Industry 4.0 NPNs, DT-based evaluation, and disaster-resilient connectivity. To achieve interoperability and alignment with industry practices, the framework adopts standardized technologies and APIs compliant with initiatives such as 3GPP NEF/CAPIF [39], CAMARA [12], and O-RAN.

The following subsections briefly describe the key functions, components and interfaces of the UNITY-6G Network Exposure Framework. Further details on the design, implementation, and operational aspects will be provided in the upcoming deliverable D3.1, where the framework architecture, supported APIs, integration mechanisms, and deployment considerations will be described in greater depth.

4.4.2 Key Functions

The Network Exposure Framework provides several key functions to enable cross-domain visibility and control of network resources:

- Secure exposure of network information and capabilities, enabling users to access monitoring data and control mechanisms across different network domains.
- Support for AI-driven network optimization, allowing orchestration platforms and vertical applications to leverage exposed metrics and analytics for intelligent service placement and resource management.
- Standardized API access to network services, ensuring interoperability with external systems and alignment with industry-standard exposure mechanisms.
- Facilitation of advanced 6G services by exposing standardized network capabilities through developer-friendly APIs, enabling vertical applications and service providers to seamless access to network functionalities in a simplified manner.

4.4.3 Components

The Network Exposure Framework implements exposure mechanisms defining two complementary layers, as depicted in Figure 4-20:

- **Network exposure Layer-1** (NEL-1), provides telco-specific APIs facilitating direct access to domain functionalities such as xApps/rApps/dApps in the RAN, SDN controllers in the transport, and NEF/CAPIF in the core; and
- **Network exposure Layer-2** (NEL-2), for app-developer-oriented APIs founded on CAMARA initiatives, simplifying complexity for external developers and vertical service providers.

Moreover, an **API management** module provides the governance and operational capabilities required to securely publish, manage, and monitor exposed APIs. This component supports functions such as API onboarding, authentication and authorization, access control, usage monitoring, and lifecycle management, ensuring secure and controlled exposure of network capabilities to internal and external consumers.

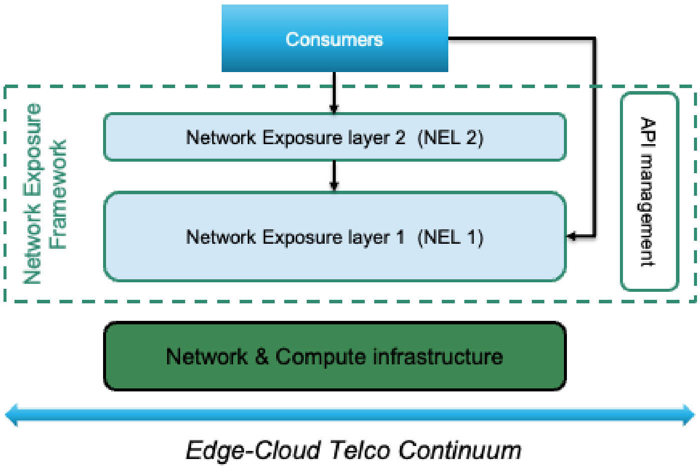


Figure 4-20. High level overview of the Network exposure framework.

4.4.4 Interfaces

The Network Exposure Framework implements domain-specific exposure mechanisms across the different network segments interfacing as follows:

1. **Core Network Exposure:** utilizes 3GPP CAPIF, NEF, and NWDAF to securely expose network events, analytics, QoS impact, and traffic management functionalities to application functions.
2. **Transport/xHaul Exposure**: leverages SDN-based interfaces to expose link- and path-level information, including topology status, latency, bandwidth availability, congestion indicators and flow-level metrics. The APIs allow external and internal UNITY-6G consumers to obtain a real-time view of the transport domain and to influence flow management decisions.
3. **RAN Exposure (Multi-RAT NPN/NTN):** encompasses both monitoring and control capabilities for multi-RAT terrestrial and non-terrestrial deployments. Leveraging O-RAN interfaces and in-band network telemetry, it delivers enhanced per-flow and per-device KPIs such as signal quality indicators, e.g., Channel Quality Indicator (CQI), resource utilization, and UE throughput, among others. Control mechanisms include parameters and scheduling policies selection, and thresholds tuning.

Through these interfaces, network exposure consumers, ranging from internal UNITY-6G architecture components such as orchestrators, to external vertical application, can interact with the framework through direct API invocations. The exposed information is jointly exploited by UNITY-6G vertical applications and internal network components to enable intelligent, cross-domain optimization decisions, ultimately supporting flexible, resilient, and performance-aware 6G service.

4.5 TRUST LAYER

The layer implements the “triangle of trust”, authentication, authorization, encryption, as a layered defense, with pervasive encryption of proprietary data and strict access control to interfaces and artifacts. Zero-Trust policies and LoT thresholds define safety envelopes (reconfiguration rates, permitted parameter ranges, blackout windows, rollback strategies). Together, these controls minimize the blast radius of misconfigurations or malicious activity and keep orchestration decisions explainable, auditable, and reversible.

4.5.1 Objective

The Trust layer is the architectural capability that turns trust from an implicit, one-off prerequisite into a continuous, measurable, and enforceable property of UNITY-6G. It addresses a fundamental challenge of 6G systems: in heterogeneous, multi-domain, and federated environments (TN/NTN, O-RAN, core, transport, edge, verticals), stakeholders cannot rely on implicit trust or static perimeter controls; instead, every interaction must be authenticated, authorized, and backed by evidence that can be verified and audited across administrative boundaries. Concretely, the layer collects and reasons over heterogeneous evidence to compute a Level of Trust (LoT), enforces Zero-Trust policies during admission, placement, scaling, and reconfiguration, and anchors trust claims and outcomes on a Distributed Ledger Technology (DLT) to provide cross-domain verifiability and non-repudiation, without exposing sensitive payloads on-chain. In doing so, it enables AI-native, zero-touch operation while preserving accountability and privacy by design.

4.5.2 Key Functions

4.5.2.1 Identity & Credential Management

The layer issues, rotates, and revokes cryptographic identities and short-lived tokens, and supports decentralized identifiers (DID) and Verifiable Credentials (VCs) for entities (users, services, models, agents) operating across domains. Authorization follows least-privilege and need-to-know principles, ensuring that each entity can only observe or influence what its role permits.

4.5.2.2 LoT Evaluation

Through the Data Continuum and the Service-Based Management Architecture (SBMA) bus, the Trust layer ingests attestations, integrity proofs, telemetry/KPIs, anomaly scores and provenance metadata generated by MS–AE–DE–ACT tuples. Analytics Engines (AE) compute LoT/DTL (with explainability), while Decision Engines (DE) consume LoT as a guard-rail for orchestration actions; ACT apply decisions under explicit safety constraints (rate limits, parameter ranges, rollback).

4.5.2.3 Zero-Trust Policy Enforcement

Access and control decisions are policy-driven and per-session, continuously re-evaluated as the operating context evolves. Trust Adapters translate and enforce these policies inside each domain (O-RAN/SMO, Core/NFVO, Transport/SDN, NTN, etc.), ensuring a consistent posture without coupling domains to a single trust technology.

4.5.2.4 Verifiability & Audit via DLT

The Smart Contract Manager (SCM), Blockchain Oracle, and Blockchain Adaptor transform selected evidence and policy outcomes into immutable, verifiable claims on the DLT: the ledger stores only hashes/indices/metadata, while full content remains off-chain. This provides independent verifiability across domains and non-repudiation of critical trust events.

4.5.2.5 Privacy-Preserving Trust

Trust computation and exposure follow data minimization: sensitive artefacts remain off-chain and are shared via the Data-Continuum; only minimal references are anchored on-chain. Privacy-preserving analytics and minimal-disclosure flows support inter-domain trust without unnecessary data exposure.

4.5.3 Components and Modules

The Figure 4-21 shows the architecture of the Trust Layer in UNITY-6G. The Trust Layer exposes its functionalities through the Data Continuum, every Domain in the Infrastructure Layer has its own Trust Adapter. Trust Adapter is also present in the IDMO.

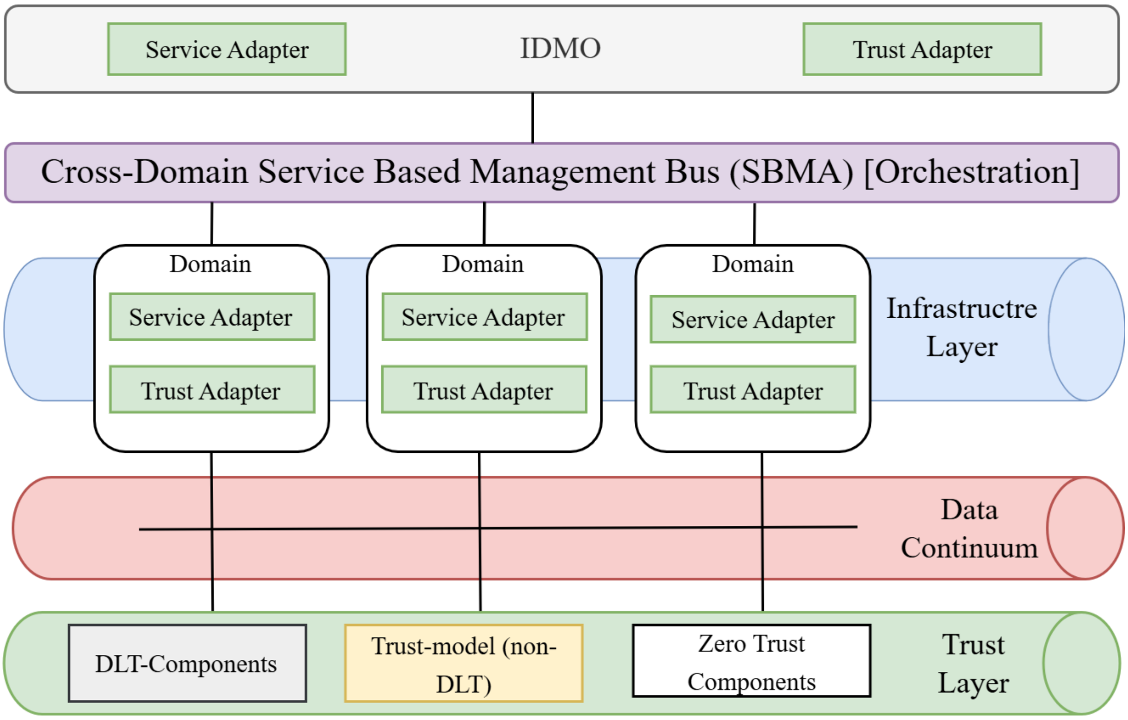


Figure 4-21: High-level reference trust domain architecture

4.5.3.1 Underlying Trust Layer

This is the foundational substrate that integrates DLT-based trust (e.g., IOTA for identity and verifiable credentials), non-DLT trust models, and Zero-Trust functions. It ensures authenticity, integrity, and privacy across the architecture and provides the auditable system of record for trust-critical events.

4.5.3.2 DLT Components

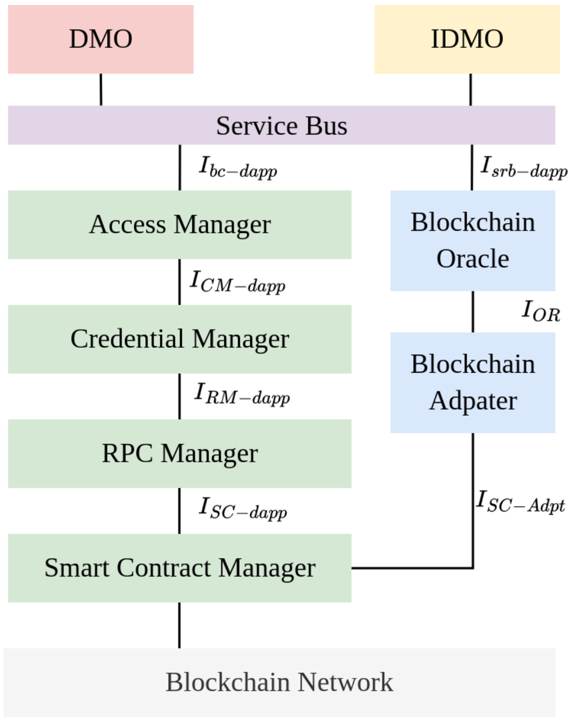


Figure 4-22. DLT components ii the UNITY-6G trust layer

- **Access Manager:** The policy enforcement point of the DLT stack: it authenticates and authorizes requests to ledger-backed services and to smart-contract actions, ensuring only entitled actors can trigger on-chain operations.
- **Credential Manager:** Manages the lifecycle of identities, keys, and credentials (including DIDs/VCs): issuance, update, rotation, and revocation, as well as propagation of revocation status to dependent components.
- **RPC Manager:** Abstracts and secures connectivity to DLT nodes (transaction submission, state queries), providing reliable access to on-chain state and receipt handling.
- **Smart Contract Manager:** Formats and submits smart-contract calls (EVM-compatible), monitors state, events and governs the contract lifecycle following a test-then-deploy discipline to minimize operational risk.
- **Blockchain Oracle:** The off-chain to on-chain gateway; it acquires telemetry/KPIs and other evidence, performs sanity/trust checks and normalization, prepares signed updates for the ledger, and listens to contract events to feed actionable signals back to orchestrators and agents**.**
- **Blockchain Adaptor:** A bi-directional adapter between Oracle and SCM: validates semantics, maps inputs to contract-specific data types, attaches domain and trust

metadata, and provides batching, rate limiting, retries, logging, and monitoring for end-to-end traceability.

4.5.3.3 Trust Adapters

Trust Adapter are lightweight adapters that translate trust services, LoT constraints, and guard-rails into the ontologies and control surfaces of each domain (e.g., SMO, NFVO, SDN controllers), aligning local control loops with system-wide trust decisions.

4.5.4 Interfaces

4.5.4.1 With Management & Orchestration (IDMO/DMO/IDM)

The Trust layer acts as a tight control partner of orchestration: it exposes identity, LoT, and Zero-Trust policies that gate admission, placement, scaling, migration, and reconfiguration. It ingests lifecycle events and monitoring outcomes, materializing significant states and decisions into governed Data-Continuum artefacts for audit, explainability, and re-training. This closed loop ensures that policy intent and operational evidence remain consistent over time.

4.5.4.2 With the Unification Layer

Through Trust Adapters, the layer brings LoT and guard-rails into domain-specific control planes like SMO in O-RAN, NFVO in Core, SDN in Transport, NTN gateways, so that local MS–AE–DE–ACT loops operate within explicit trust boundaries and export fresh evidence back to the Trust layer.

4.5.4.3 With the Data Continuum

The Data Continuum is the transversal substrate for evidence and results: telemetry, anomalies, KPIs, and provenance flow into LoT evaluation; outcomes (scores, policy decisions, events, and explanations) are materialized as data products with cataloguing, lineage, and access control. This enables replay, audit, cross-domain reuse, and model lifecycle management.

4.5.4.4 DLT service-stack interfaces

As Figure 4-22 shows, the DLT service-stack interfaces are:

- I_bc-dapp: expose operations such as “deploy contract”, “submit transaction”, “query ledger state”, so orchestrators can use ledger capabilities without binding to a specific client.
- I_srb-dapp: requests for off-chain data collection/verification that will be forwarded on-chain via the Blockchain Oracle.
- I_CM-dapp: key/credential issuance, update, revocation, and associated authorization policies.
- I_RM-dapp: transport of validated credentials and connection parameters to establish authenticated RPC channels to DLT nodes.
- I_SC-dapp: invocation of smart-contract methods, submission of state-changing transactions, read-only calls, and return of hashes/receipts/events.
- I_OR: push of normalized data items and events, acknowledgements and retransmissions, freshness checks, and admissible-source validation.
- I_SC-Adpt: hand-over of validated/formatted payloads and target contract/method metadata, closing the loop from off-chain evidence to on-chain state updates.

Co-funded by the European Union

5 ARCHITECTURAL SUPPORT FOR USE CASES

5.1 UC1 - DISASTER RECOVERY AND HANDLING FOR SUSTAINABLE 6G NETWORKS AND FUNCTIONAL FLOW SPECIFICATION

5.1.1 Use case definition and scope

UC1 defines an AI-native operational flow for resilient, sustainable 6G networks under disruptive events (e.g., power failure, fiber cut, cyber breach, extreme environmental conditions). The core objective is to keep critical services available and recover rapidly while optimizing multi-objective trade-offs (latency, energy, cost, connectivity and capacity, etc).

The approach is explicitly Digital-Twin (DT)–guided and policy-driven, implemented via a cross-domain mesh of services/agents that cooperate through Monitoring, Analytics, Decision, Act (MS–AE–DE–ACT) tuples, with execution via orchestration and control functions (e.g., SMO/RIC).

5.1.2 Operational lifecycle

UC1 is organized as a closed-loop lifecycle with explicit stage transitions. The network runs a readiness loop during normal operation, escalates when detection thresholds are met, executes disaster operations until functionality is restored, and then returns to steady-state while learning from outcomes.

UC1 lifecycle diagram:

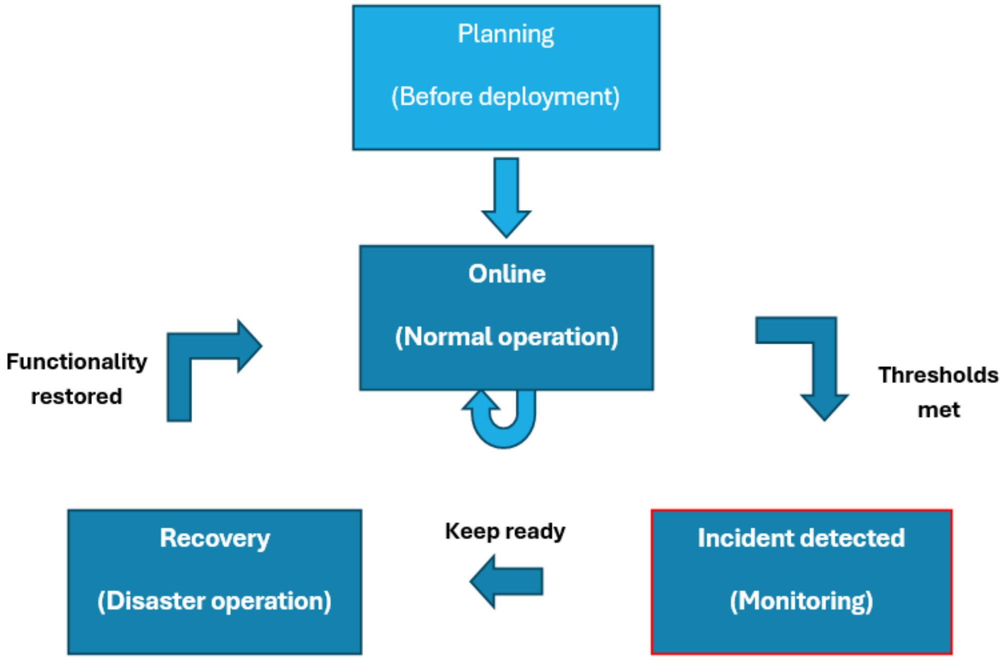


Figure 5-1. UC1 Lifecycle Diagram

5.1.3 Functional flow by operational stage

5.1.3.1 Stage 1 - Planning (Before network deployment)

Goal: Be ready. Know the baseline, the weak spots, and the fallback playbooks.

Core processes and steps:

- **Optimal baseline setup search (DT-driven optimization).**
  - Run topology and resource-allocation simulations in the Digital Twin to find a "best possible" baseline configuration.
  - Solve multi-objective trade-offs (e.g., allocations, latency, energy, cost, connectivity, etc.) to define the target baseline functionality to regain after disruptions.
- **Failure simulations (scenario library + policies).**
  - Simulate representative disaster scenarios (or approximations) and derive an optimal policy/playbook per scenario (e.g., resources, management actions, traffic priorities).

- Identify single points of failure and weak spots; define mitigation patches and priorities according to resilience vs. cost/complexity trade-offs.

- **Predetermine resilience locations to host control elements and agents.**
  - Rank geographic/physical locations by resiliency; position critical control/compute elements and agents in the most resilient locations.
  - Define redundancy/backup locations and required additional hardware to enable takeover if a primary control site is impaired.
- **Offline training of AI/ML models for survival modes and forecasting.**
  - Train models/policies for "surviving" topologies and degraded-service modes (e.g., which services/slices are throttled first under resource scarcity).
  - Train prediction models for environmental changes (weather/geological/etc.) to estimate disaster probability and pre-trigger proactive actions.

5.1.3.2 Stage 2 - Online (Normal operation)

Goal: Stay ready. Continuously predict, update, and keep agents in sync.

Operational flow (steady-state preparedness loop):

- **Run cross-domain disaster-preparedness loops during calm periods.**
  - Each domain continuously monitors KPIs/alerts and maintains readiness in a lightweight steady-state loop.
  - A mesh of autonomous agents and shared services cooperates through MS–AE–DE–ACT tuples deployed at different layers of the network.
- **Maintain online service/agent pillars.**
  - Observability & Telemetry: keep KPI/alert pipelines real-time and loss-free.
  - Digital Twin & Forecasting: continuously simulate the "best possible" baseline and forecast load/temperature/failure risk.
  - Anomaly & Drift Detection: detect subtle KPI drifts before thresholds break.

    - Trust, Security & Configuration Integrity: ensure binaries, models and tokens are authentic.
    - Readiness Governance: quantify and track how prepared the network is at any point in time.
- **Define and maintain escalation triggers.**
    - Continuously evaluate detection thresholds and escalation policies to hand off from Online to Incident/Disaster operations when needed.
- **Keep the Digital Twin synchronized with real-world conditions.**
    - Monitor real-time network components/state and feed observations back into baseline calculations to update current “best possible” setups.
    - Update resiliency points and issue “transfer of responsibility” to alternative control/compute locations when needed to maintain readiness.
    - Keep all running agents updated with global/neighbor states to enable coordinated decisions.
- **Continue disaster prediction and strategy updates.**
    - Refresh prediction models periodically and generate early warnings for impending disaster conditions.
    - Update optimal strategy models (playbooks/policies) conditioned on the most likely disaster scenarios.
- **Maintain readiness via proactive, distributed actions.**
    - Continuously assess risk factors; when risk is high, take evasive actions before the disaster (move critical components, allocate extra resources, spin up agents/services).
    - Ensure distributed control can take over if core/control components are cut off; elect a new leader where a predefined mechanism exists.
    - Maintain local autonomy: each device/component updates local models/agents to take localized actions if no external instructions are available.

5.1.3.3 Stage 3 - Incident detected (Monitoring)

This stage is entered when monitoring thresholds are met (e.g., KPI anomalies, correlated alarms, or forecasted hazard conditions). The primary purpose is rapid detection, initial classification, and controlled transition into disaster operations with minimal instability.

- **Detect and declare the incident.**
  - Monitor key indicators across the network; once thresholds are crossed, declare a disaster/incident and perform an initial classification.
- **Classify and select the recovery path (DT-guided).**
  - Use the Digital Twin scenario library to match observed symptoms to a scenario class (e.g., power failure, fiber cut, cyber breach).
  - Instantiate the corresponding first-response template and objectives for the next stage.
- **Enter a controlled degraded mode (“graceful degradation”).**
  - Reset KPI targets/threshold interpretations against the new reality to avoid unstable control loops and alert storms.
  - Stabilize admission/control policies for critical services prior to large-scale reconfiguration.

5.1.3.4 Stage 4 - Recovery (Disaster operation)

Goal: Detect, Isolate, Contain, Stabilize, Restore, Learn.

- Apply the scenario-aware first-aid template.
  - Execute Detect, Isolate, Contain, Stabilize actions as guided by the DT policy selected in the Incident stage.
- Divert resources to maintain maximal capability for life-critical traffic.
  - Re-allocate spectrum, power, compute and slice capacity to prioritized services/traffic.

    - Steer tunnels from central to edge-UPF or satellites as dictated by the DT capacity map.
- Run the closed-loop decision and execution cycle.
    - Feed live KPIs into the Digital Twin; the Decision Engine selects the next intent bundle; SMO/RIC executes actions.
    - Store each action's regret score to support online learning and post-incident improvement.
- Maintain trust and resolve conflicts as the topology and users change.
    - Require remote attestation for newly activated cells/edge DCs; use PKI to issue short-lived certificates.
    - Arbitrate conflicting slice intents across operators/public-safety/MVNOs via a conflict solver.
    - Accelerate UE re-authentication to prevent attach storms.
- Prioritize physical repair actions for field/technical teams.
    - Use DT ranking to prioritize fixes (e.g., backbone paths, public-safety RUs, edge-DC power/cooling, mass-market sectors).
    - Auto-dispatch work orders; recompute priorities whenever KPI thresholds are crossed.
    - Generate concise situation reports for regulators, customers, and partners.
- Restore, validate, and learn.
    - Progressively restore services while validating stability and KPI recovery against the updated baseline.
    - Update the DT scenario library, policies, and readiness governance metrics using the outcomes and recorded regret scores, then transition back to Online once functionality is restored.

5.1.4 Operational assumptions and policy framework

Disaster taxonomy and trigger logic

UC1 requires a repeatable and auditable mechanism to recognize disruptive conditions, classify them into actionable disaster types, and trigger a controlled operational posture that preserves critical services while minimizing collateral impact. The intent is not to prescribe fixed KPI numbers, but to define a policy-driven framework that can be instantiated per operator, per stakeholder, and per service class. The framework is designed to work across domains (RAN, transport, core, edge/cloud, NTN) and to remain valid under partial observability during disasters.

From a satellite operator perspective, NTN in UC1 should be regarded as a resilience domain, not merely as a backup connectivity layer. Its contribution to service continuity under terrestrial degradation depends on operational constraints such as beam availability, gateway accessibility, orbital conditions, and prioritization policies. For this reason, coordination with the UNITY-6G framework should be based on abstracted status and capacity indicators, allowing cross-domain orchestration while preserving operator control over internal resource management.

At the core of the approach is a disaster taxonomy that groups incidents into a small set of categories, each corresponding to a response family and actuation envelope. A typical taxonomy includes: (i) precursor/early warning conditions that justify increased readiness without declaring a disaster; (ii) localized infrastructure disruption that is likely containable within a limited region, domain, or dependency cluster; (iii) regional multi-domain disruption where correlated symptoms span multiple adjacent sites and multiple layers, suggesting cascading behavior or a common external cause; (iv) wide-area disruption/partitioning in which reachability is lost or parts of the network operate in "island mode," requiring distributed autonomy; (v) energy-constrained operation where power scarcity drives explicit energy-aware prioritization; and (vi) security-driven disruption where integrity/trust anomalies require isolation and restricted actuation. In addition to the category, each incident is assigned a severity level defined conceptually by the extent of service impact, exposure of critical services, and the geographic/topological spread of symptoms, rather than by fixed numeric thresholds.

To declare and classify incidents reliably, UC1 uses a multi-gate trigger logic rather than single-metric triggers. The proposed logic includes four gates. First, a detection gate recognizes abnormal signals ("symptoms") from the monitoring plane such as sustained KPI drift, unusual alarm patterns, reachability anomalies, resource stress, or trust/integrity events. Second, a correlation gate validates that symptoms cohere in time and scope and are consistent with known dependencies. Correlation considers: persistence beyond transient fluctuations; clustering by geography/topology/service chain; corroboration across relevant domains when applicable; and alignment with a dependency graph (service paths, slice mappings, control-plane relationships). Third, a classification gate maps the correlated evidence to the taxonomy category to select the appropriate response family. Fourth, a declaration gate determines whether to declare an incident and chooses an initial operational posture, normal, heightened readiness, controlled degradation, or partition-tolerant mode, based on the classified event and its conceptual severity/confidence. This structure supports multiple implementation strategies (rules, statistical detection, ML, DT-assisted inference, or hybrids) while maintaining consistent semantics at the UC level.

Because UC1 must remain adaptable across deployments, thresholds are treated as policy-controlled criteria rather than hard-coded numbers. Thresholds are defined conceptually as configuration artifacts describing: which metric/event is relevant; the scope (per slice/service, per region/site, per path/service chain, per domain); the condition type (absolute bound, deviation from baseline, rate-of-change, persistence); and the confirmation semantics (must persist, must co-occur, must match dependency expectations). This enables different stakeholders (e.g., public safety, industrial tenants, consumer services) to apply differentiated policies while preserving a common incident-declaration process. All threshold policies are versioned and auditable, and can be adapted over time without changing the UC1 flow.

UC1 also defines a timing model using conceptual correlation windows to separate transient disturbances from sustained incidents and to avoid oscillations. A detection window identifies that symptoms have emerged; a confirmation window validates persistence; a correlation window links multiple symptoms into a coherent hypothesis; and a stabilization/cooldown window prevents immediate re-triggering after an action is applied. The durations are not fixed in the specification; they are governed by operational policy and informed by measurement granularity, domain reaction times, and service criticality.

Once an incident is declared, UC1 transitions into controlled degradation rather than unconstrained reactive changes. Controlled degradation is governed by guardrails that define what must be preserved, what actions are allowed, and how rapidly the system may change. Guardrails include: a priority ladder for services/slices (stakeholder-defined, emphasizing critical communications first); an action envelope limiting the types and scope of actuation (e.g., rerouting, slice reprioritization, resource reallocation, service migration, admission control) to the affected region/domain where feasible; anti-oscillation principles such as hysteresis and stabilization periods; safety constraints that must never be violated (e.g., isolation boundaries, industrial safety traffic requirements, security containment rules); and exit criteria that define when the system may transition from degradation into recovery optimization (conceptually: sustained stability and acceptable service state).

Finally, the trigger-and-classification pipeline produces explicit output artifacts that integrate directly with the UNITY-6G architecture's closed loops. These artifacts include an incident descriptor (scope, taxonomy category, severity/confidence), evidence bundle (symptoms and correlation rationale traceable to telemetry/events), an operational posture profile (controlled degradation policy and permitted actuation envelope), and an audit record (policy version and rationale).

5.1.5 Alignment with the UNITY-6G proposed architecture

5.1.5.1 Architectural elements exercised

- Service-based orchestrator and lifecycle management capabilities to instantiate, scale and migrate functions and supporting AI services across heterogeneous domains under stress.
- Network exposure layer with standardized APIs to query abstracted RAN/transport/core state (including per-tenant resource usage) and trigger reconfiguration actions without vendor-specific bindings.
- Distributed MS–AE–DE–ACT tuples placed across the edge–cloud continuum (centralized, distributed or federated as needed) to maintain operation when connectivity to higher tiers is degraded.
- Open RAN control surfaces (SMO / non-RT RIC and near-RT RIC xApps) for rapid RAN reconfiguration, complemented by transport/core controllers for multi-domain recovery actions.
- Trust, security and governance functions (attestation, identity and policy enforcement) to keep control-plane decisions and activations safe during dynamic topology changes.

5.1.5.2 Stage-to-architecture mapping

5.1.5.2.1 Planning (before deployment)

Executed primarily in the DT and slower-loop analytics: build a scenario library, baseline configurations, and playbooks. The orchestrator pre-positions critical control elements and defines fallback placements and policies; models and policies are trained offline for survival modes and disaster forecasting.

5.1.5.2.2 Online (normal operation)

Runs continuous readiness loops via distributed tuples: MS streams KPIs/alarms; AE maintains forecasts and drift detection; DE updates readiness scores and pre-emptive intents. The exposure layer provides consistent observability across domains; orchestration keeps agents synchronized and can proactively move or scale key services when risk rises.

5.1.5.2.3 Incident detected (monitoring thresholds met)

Transitions to a fast reactive loop: correlated symptoms and forecasts trigger incident declaration and classification. AE/DT matches the observation to a scenario class; DE selects a first-response intent bundle and enforces controlled degradation guardrails to prevent instability and alert storms.

5.1.5.2.4 Recovery (disaster operation)

Operates as repeated observe, decide, execute cycles. DE generates prioritized intents (traffic/slice prioritization, routing and resource reallocation, edge/NTN anchoring shifts), and ACT executes via SMO/RIC and domain controllers. Execution feedback (MS) closes the loop; actions and outcomes are logged for post-incident learning and DT/playbook updates before returning to Online.

5.1.5.3 Non-functional considerations and governance

UC1's recovery objectives include energy-aware operation under scarcity. The proposed architecture supports this by exposing energy telemetry via dedicated data APIs, using AI-based forecasting to guide workload shifts and prevent blackouts, and enabling coordination with local microgrids and renewable sources. These capabilities are used both proactively (risk-driven preparedness) and during recovery (energy-aware resource and service reconfiguration).

5.1.5.4 Cross-use-case interactions

UC1 is implemented as an AI-native, service-based closed-loop that spans RAN, transport (X-haul), core and edge/NTN domains. The proposed UNITY-6G architecture enables this by combining (i) a distributed intelligence sublayer based on MS–AE–DE–ACT tuples, (ii) a Digital Twin (DT) for scenario-driven optimization and policy selection, and (iii) orchestration/control functions (e.g., SMO / non-RT RIC, near-RT RIC and domain controllers) that can reconfigure services, priorities, routing and energy profiles during disruptive events.

5.1.5.4.1 Multi-tenant governance and conflict resolution

Disaster conditions amplify contention between tenants (e.g., public safety, MNOs, ISPs, satellite operators) and between concurrent control loops. The proposed architecture addresses this through explicit conflict detection and resolution mechanisms (policy-based arbitration and hierarchical coordination) and trust mechanisms such as distributed ledgers / smart contracts for transparent, enforceable resource sharing and decision traceability. UC1 leverages these functions to arbitrate slice/resource intents, prevent oscillations, and maintain isolation and fairness while prioritizing critical services.

5.2 UC2 - REAL-TIME XR HOLOGRAPHY IN 6G NETWORKS AND FUNCTIONAL FLOW SPECIFICATION

5.2.1 Use case definition and scope

UC2 targets real-time, multi-user XR (holographic/immersive) collaboration over 6G, where high-fidelity 3D content is rendered on edge/cloud GPUs and streamed to XR clients with stringent latency and Quality-of-Experience (QoE) constraints. The flow coordinates orchestration, network resource allocation (including transport backhaul choices), multi-user session management, and closed-loop monitoring/adaptation to maintain SLA targets (e.g., latency, bandwidth, frame rate/FPS).

5.2.2 Operational lifecycle

UC2 operates as a continuous loop: a session is initialized, deployed on suitable compute/network resources, runs in real time with multiple users, and is continuously monitored to adapt resources and policies. Monitoring insights go through analytics engine and the decision engine to improve future allocations and enable runtime adaptation.

UC2 lifecycle diagram:

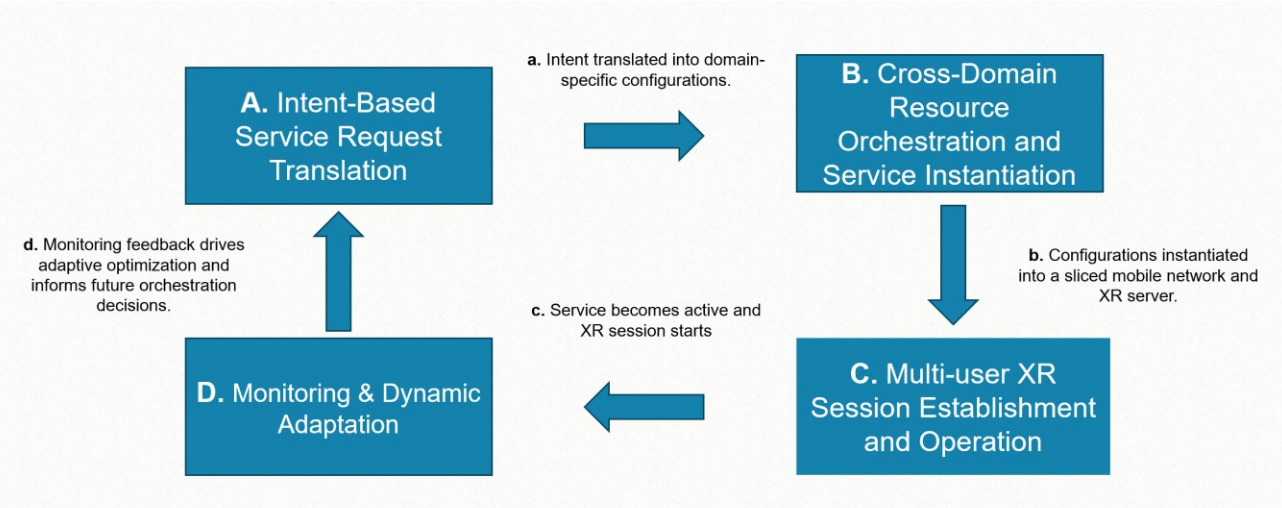


Figure 5-2. UC2 Lifecycle Diagram

5.2.3 Functional flow by operational stage

5.2.3.1 Stage 1 - Intent-Based Service Request and Translation

Goal: To translate a high-level XR service request with defined Service Level Agreements (SLAs) into domain-specific configurations and identify suitable interconnected domains.

Processes and steps:

Processes:

- Intent interpretation and SLA extraction.
- Decomposition into Resource Facing Services (RFS).
- Domain capability matching.
- Control tuple generation.

Steps:

1. The XR service intent (including SLA and compute requirements) is submitted to UNITY-6G.
2. The intent is decomposed into RFS configurations for:
    a. RAN (radio behaviour and QoS),

   b. Transport (latency-aware routing),

   c. Core (session management),

   d. XR vertical App compute resources (GPU/VM resources).

3. A matching mechanism selects interconnected domains capable of jointly satisfying the SLA.

4. Domain-specific tuples are prepared to control instantiated components derived from the initial request decomposition.

5.2.3.2 Stage 2 - Cross-Domain Resource Orchestration and Service Instantiation

Goal:. To provision and interconnect compute and network resources in compliance with SLA-defined constraints.

Processes and steps:

Processes:

- Sliced mobile network creation.
- Cross-domain interconnection.
- Compute resource instantiation.
- Vertical App deployment.

Steps:

1. A sliced mobile network is instantiated across:

   a. RAN (radio configuration),

   b. Transport (low-latency connectivity),

   c. Core (UPF placement and traffic steering).

2. Transport configuration ensures proper interconnection between all selected domains.

3. The XR server application (Hololight Space) is started by the UNITY-6G framework on the selected compute infrastructure.

4. Confirmation is provided at the UNITY-6G interface that all components associated with the intent have been instantiated.

5.2.3.3 Stage 3 - Multi-User XR Session Establishment and Operation

Goal: To establish and sustain a synchronized, immersive, multi-user XR collaboration session under SLA constraints over the already-instantiated mobile network and XR server instance.

Processes and steps:

Processes:

- XR client launch.
- XR Session establishment.
- Real-time rendering and XR streaming.
- Runtime service enforcement.

Steps:

1. Upon service readiness confirmation, the XR headset client is launched.
2. The client app on the headset connects to the running XR server via the configured sliced mobile network.
3. A live XR session is established; additional users repeat the same process for multi-user collaboration.
4. The XR session operates over the configured RAN, transport, and core segments.

5.2.3.4 Stage 4 - Monitoring & Dynamic Adaptation

Goal: To ensure continuous SLA compliance and optimised resource utilisation through real-time telemetry analysis and dynamic adaptation.

Processes and steps:

Processes:

- Cross-domain telemetry collection.
- SLA evaluation and analysis.
- Decision-making.

- Actuation via control tuples.

Steps:

1. Telemetry is collected from:
    a. Application layer: latency, bitrate, FPS.
    b. Compute layer: GPU utilization, memory usage.
    c. Network layer: latency, jitter, bandwidth, packet loss.
2. Decision engines evaluate SLA compliance and identify necessary adjustments.
3. Actuators modify domain components using predefined control tuples:
    a. Adjust radio scheduling,
    b. Modify transport routing,
    c. Scale compute resources,
    d. Tune XR application parameters.
4. The system stabilizes and continues monitoring, forming a continuous optimization loop.

5.2.4 Operational assumptions and policy framework

5.2.4.1 Session profiles and adaptation policy ladder

UC2 (Real-time XR holography) benefits from explicitly defining two complementary constructs: session profiles (classes) and an adaptation policy ladder. Together, they provide a systematic way to (i) translate an XR service request into concrete network/compute intents aligned with the proposed architecture, and (ii) ensure runtime adaptations are applied in a stable, non-oscillatory manner when conditions change. The goal is to remain technology-agnostic and avoid hard-coded numeric thresholds while still being implementable across heterogeneous deployments.

Architectural and Operational Assumptions

The following assumptions are made prior to runtime execution:

- Intent-Based Service Invocation - XR services are requested through a high-level service intent submitted to the UNITY-6G IDMO. The intent includes SLA descriptors (e.g., latency bounds, throughput requirements, reliability targets) and compute characteristics (e.g., GPU acceleration requirements). The intent does not directly configure infrastructure components; instead, it is translated into domain-specific configurations.
- Cross-Domain Orchestration of a Sliced Mobile Network - The XR service operates over a fully sliced mobile network spanning: RAN, Transport, Core. The slice represents an end-to-end logical construct rather than a single-domain configuration. The XR session itself is not cross-domain; rather, the mobile network orchestration is cross-domain.
- Vertical Application Instantiation by the Framework - The XR server (Vertical Application) is instantiated by the UNITY-6G framework on selected compute infrastructure. The application operates over the provisioned network slice and does not directly manage infrastructure provisioning.
- Tuple-Based Controllability - Instantiated components in each domain expose controllable configuration tuples. These tuples represent parameterized control surfaces (e.g., radio scheduling parameters, transport routing configurations, compute scaling settings) that can be modified by actuators under decision-engine control.
- Closed-Loop Automation - The architecture operates under a continuous monitoring and adaptive control paradigm, integrating MS, AE, DE, and ACT mechanisms.

5.2.4.1.1 Session profiles

A session profile is a versioned service-class definition that maps an XR session type to the required end-to-end treatment across the architecture: slice/QoS intent, GPU/compute intent, and edge placement constraints. UC2 should define a small set of profiles that cover the intended XR modalities:

- Interactive multi-user holographic collaboration (e.g., high interactivity, strict latency stability, multi-user synchronization)
- Single-user immersive XR (e.g., latency sensitive, less fan-out,)
- Broadcast/lecture-style holography (e.g., high bandwidth, less interactivity, different scaling strategy)
- Industrial/mission-critical XR assist (e.g., strong reliability requirements, explicit priority governance)

For each profile, the UC2 specification is defined, conceptually and vendor-neutral:

- Service intent and slice/QoS profile: The profile selects a network treatment class (e.g., "interactive low-latency XR," "high-throughput XR," "mission-critical XR"), expressed as an intent that the orchestration/control plane can translate into slice parameters and transport treatment. This includes the notion of priority, isolation, and assurance level.

- GPU/compute profile: The profile specifies the compute characteristics needed by the XR pipeline (rendering, encoding, compositing, state synchronization), expressed as a network-resource compute intent (GPU-enabled instance class, acceleration requirements, scaling mode, and whether stateless replication is permitted).

- Edge placement constraints: The profile defines placement rules, expressed conceptually as constraints and preferences, such as: "must be placed close to the user cluster," "must remain within a regulatory zone," "prefer co-location with a specific transport anchor," "allow migration only if state transfer constraints are satisfied," and "prefer resilient sites."

- Session topology and state model: The profile states whether the session is primarily point-to-point, hub-and-spoke, mesh fan-out, or hierarchical, and whether the state is centralized, partitioned, or replicated. This directly informs scaling and migration choices.

- Security and governance: The profile defines authentication posture, tenant isolation needs, and audit requirements (important for enterprise and public-sector XR), aligned with the UNITY-6G architecture's exposure capabilities and trust mechanisms.

Operationally, session profiles enable the system to perform a consistent translation: XR Session Request, Session Profile, (Network Slice/QoS Intent) + (Compute/GPU Intent) + (Placement Constraints) + (Policy Constraints). This translation is the key "bridge" between UC2 and the proposed architecture's orchestrators, RIC/SMO surfaces, and exposure APIs.

5.2.4.1.2 Adaptation policy ladder

UC2 should explicitly define an ordered ladder of adaptation actions that the system attempts when QoE risk increases (or when resources become constrained). The ladder provides two benefits: it prioritizes lower-cost, less disruptive actions first, and it reduces thrashing by making escalation deliberate and policy-driven.

A recommended conceptual ladder is:

1. Encoding and application-layer adaptation (least disruptive). Adjust media parameters (e.g., bitrate selection strategy, representation choice, forward error resilience mode), and session-level policies such as per-user stream prioritization. This is typically fast, localized, and reversible.
2. Network slice/QoS adjustment. Modify the session's network intent within allowed policy bounds: re-prioritize the session within its class, adjust transport treatment, or update slice parameters. This uses the architecture's assurance/exposure surfaces and control loops.
3. Horizontal scaling of XR service components. Scale the XR pipeline (streaming workers, encoders, session fan-out components, state sync services) while keeping placement stable where possible. Scaling is coordinated by the orchestration/LCM at the vertical XR App domain, and is constrained by the session topology and state model.
4. Placement change/migration (most disruptive). Re-place or migrate components to a different edge site (or replicate then cut over), used when locality or capacity constraints cannot be met otherwise. This step requires explicit policy checks: state transfer feasibility, session continuity guarantees, and tenant constraints.

UC2 also includes stabilization principles that prevent oscillatory adaptations:

- Stabilization timers/cooldown behaviour: After any ladder step is applied, the system waits a defined stabilization interval (policy parameter) before making further changes, unless a safety/critical condition forces immediate escalation.
- Hysteresis and escalation rules: The system escalates only if risk persists after stabilization, and de-escalates conservatively (i.e., do not immediately revert to a lower rung unless stability persists).
- Rate limiting and change budgeting: Limit the frequency and magnitude of reconfigurations per session and per region to protect overall platform stability.
- Session admission and graceful degradation: If resources cannot satisfy the session profile, the ladder may invoke admission control actions: reject new joiners, reduce session fidelity, or downgrade the session to a lower profile class according to policy.

5.2.5 Alignment with the UNITY-6G proposed architecture

5.2.5.1 Architectural elements exercised

- Service-based orchestration and lifecycle management: Instantiates and manages both the mobile network components and the XRApp server side service components (session controller, rendering/streaming servers as CNFs) and places them on suitable domains and sites within the domain (e.g., edge or cloud sites); supports scaling, migration, and rollback actions.
- Compute continuum with GPU acceleration: Provides edge/regional/cloud execution venues with GPUs and storage for caching/pre-processing 3D assets, enabling low-latency rendering close to users while retaining elastic capacity in higher tiers.
- Network exposure and API framework (NEF/CAPIF-style): Exposes standardized APIs for capabilities such as location and QoS information and provides controlled northbound access for XR applications and service managers without relying on vendor-specific interfaces.
- E2E connectivity control: slicing, transport/backhaul, and (when applicable) TN-NTN integration: Establishes low-latency, high-throughput paths and slice-level guarantees across RAN, transport, and core; supports fallback or augmentation with high-capacity satellite backhaul where terrestrial resources are constrained.
- Closed-loop assurance (MS–AE–DE–ACT): Collects telemetry (network, compute, application QoE) from the different components deployed at each domain, performs (semantic-aware) analytics, derives decisions under policy/SLA constraints, and executes changes via the orchestration and control plane.
- Open RAN control surfaces (SMO / non-RT RIC / near-RT RIC): Enables AI/ML-based RAN optimization for XR sessions (e.g., admissions and PRB allocation via rApps/xApps) to preserve low latency and reliability under load and mobility.
- Trust, security, and governance: Applies authentication/authorization for session requests and API exposure; enforces tenant isolation and policy guardrails when multiple sessions compete for shared edge and spectrum resources.

5.2.5.2 Stage-to-architecture mapping

5.2.5.2.1 Stage 1 - Intent-Based Service Request and Translation

1. XR clients submit a session request; the XR session manager at the server side authenticates and applies initial policy (service profile, target QoE).
2. Orchestrator discovers candidate edge sites and resources; it can use exposure APIs (e.g., location and QoS) to identify the best serving region and constraints.
3. Admission control evaluates current slice/transport capacity and GPU availability before accepting the session.

5.2.5.2.2 Stage 2 - Cross-Domain Resource Orchestration and Service Instantiation

1. Lifecycle management instantiates the required CNFs (render/stream servers, session control) on the selected edge or cloud site and allocates GPU, CPU, memory, and storage.
2. Transport/slice control establishes the E2E connectivity profile (low-latency path, bandwidth guarantees); when relevant, the orchestrator selects terrestrial vs TN-NTN backhaul options and configures breakout/anchoring to minimize delay.
3. Content is preloaded/cached at the edge (3D assets, scene state) to reduce join-time; secure session parameters and endpoints are returned to all participants.

5.2.5.2.3 Stage 3 - Multi-User XR Session Establishment and Operation

1. GPU-accelerated rendering produces the XR stream while session logic maintains consistent shared state across users.
2. RAN and transport components support the requested service profile; near-RT control can optimize scheduling/resources to keep latency and reliability within targets as users move and load fluctuates.
3. Application-level techniques (adaptive bitrate, compression/semantic level selection) operate in tandem with network control to preserve QoE under transient constraints.

5.2.5.2.4 Stage 4 - Monitoring and Dynamic Adaptation

1. MS gathers continuous telemetry across domains: latency/jitter/throughput and packet loss; GPU/CPU/memory utilization; FPS, dropped frames, and join/reconnect events.
2. AE correlates symptoms and predicts degradation; DE selects mitigation actions under SLA and policy constraints (scale-out/in GPU instances, migrate a session to a closer edge, adjust slice bandwidth, reroute transport, tune compression).

3 ACT executes the chosen actions via orchestration (CNF scale/migrate) and network/RAN control surfaces (slice updates, RIC-driven radio optimization) and confirms the new steady state; outcomes are logged to improve future placement and adaptation policies.

5.2.5.3 Non-functional considerations and governance

The UC2 real-time XR service is governed by strict non-functional requirements concerning latency, throughput stability, reliability, scalability, and security, all enforced through the UNITY6G intent-driven and closed-loop orchestration framework. End-to-end performance is ensured via coordinated slice configuration across RAN, transport, and core domains, combined with appropriate GPU-enabled compute provisioning and continuous SLA monitoring. Reliability and availability are supported through domain-aware placement, dynamic reconfiguration, and controlled scaling mechanisms, while scalability is achieved through elastic compute allocation and adaptive slice management. Security and trust are maintained through authenticated intent submission, policy-controlled exposure interfaces, encrypted inter-domain communication, and tenant isolation. Overall, governance is policy-driven and cross-domain coordinated, ensuring that all adaptation actions remain consistent with SLA objectives, stabilization principles, and platform-wide integrity constraints.

5.3 UC3 - DIGITAL TWIN FOR INTEGRATED 6G NETWORK EVALUATION AND FUNCTIONAL FLOW SPECIFICATION

5.3.1 Use case definition and scope

UC3 defines a digital-twin-driven lifecycle to evaluate, optimize, and safely evolve an integrated 6G network. The digital twin is built from modular libraries (RAN, Core, Transport, Edge/Cloud compute, AI/analytics, security, observability), used to run ‘what-if’ scenarios and conflict-resolution, evaluate KPIs, and then synchronize validated optimizations back to live operations via controlled deployment. The lifecycle includes an explicit service-evolution stage where ML/optimization improves models, policies, and libraries over time.

5.3.2 Operational lifecycle

The UC3 lifecycle proceeds through five operational stages. Each stage produces artifacts that feed the next stage; two feedback loops ensure that models and component libraries evolve based on systematic evaluation results and live operations.

UC3 lifecycle diagram:

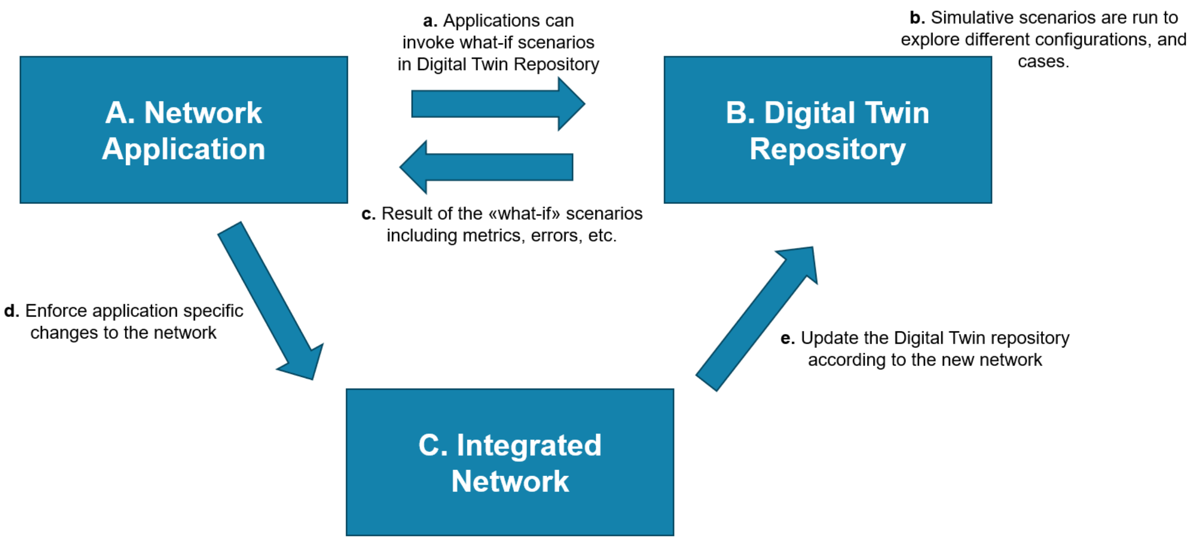


Figure 5-3. UC3 Lifecycle Diagram

5.3.3 Functional flow by operational stage

5.3.3.1 Stage 1 - Network Application

Goal: Prepare a set of libraries and a DT "Core" that composes the libraries according to user/system input, allowing Network Applications to invoke "what-if" scenarios in the Digital Twin Repository.

Processes and steps:

- Develop a modular library set and a Digital Twin (DT) Core that models the UNITY-6G system across domains:
  - RAN (ORAN modules, etc)
  - Core Network
  - Edge/Cloud Computation

    - Transport Network for data and
    - AI/Analytics Components
    - Security / Privacy Services
    - Observability (implementing OpenCAPIF)

- Architecture approach: a loosely coupled, event-driven composition where DT components exchange events and state via well-defined interfaces, enabled by a deployment configuration.
- Define a per-scenario composition profile (which libraries are invoked, their parameters, and their coupling) so each simulation run is reproducible and traceable.
- **Trigger "What-If" Scenarios (Flow a):** Network Applications invoke specific configurations within the Digital Twin Repository for testing.

5.3.3.2 Stage 2 - Digital Twin Repository & Scenario Simulation

Goal: Run simulative scenarios within the Digital Twin Repository to explore different configurations and cases, successfully detecting conflicting resources prior to execution.

Processes and steps:

- Online identification of model network resources and dependencies within the digital twin simulation environment.
- **Run Simulations (Flow b):** Execute scenario-based simulations to test resource allocation, service requirements, and infrastructure limits (flow b).
- Analyze and detect resource conflicts specifically within the *twinned target network* (e.g., overlapping assignments, bottlenecks, or incompatible configurations of twinned xApps/rApps) during simulation execution

5.3.3.3 Stage 3 - Evaluation & Integrated Network Deployment

Goal: Assess the results of the simulated scenarios, safely enforce approved changes to the live Integrated Network, and maintain continuous synchronization with the twin.

Processes and steps:

- **Evaluate Results (Flow c):** Extract the results of the "what-if" scenarios from the Digital Twin Repository back to the Network Application to support component-level decision-making.

- **Enforce Changes (Flow d) if needed:** Deploy approved updates and optimizations from the Network Application safely to the live Integrated Network environment through automated, controlled pipelines.

- **Repository Update & Synchronization (Flow e):** Maintain continuous bi-directional synchronization from the Integrated Network back to the Digital Twin Repository so it accurately reflects the real-time baseline state and events.

- Ingest live network telemetry and simulation data to continually train models and update DT component libraries, improving future scenario evaluations.

5.3.4 Operational assumptions and policy framework

5.3.4.1 Digital Twin coupling model

The UC3 Digital Twin is specified as a hybrid twin that combines three representation modes, selected per domain and per use case objective:

- Simulated components (first-principles / discrete-event / network simulation): Used where behavior is well-characterized and controllable, and where we need “what-if” exploration under topology, traffic, mobility, or failure scenarios. Typical candidates include abstracted RAN scheduling models, transport routing/queuing abstractions, service-chain and slice models, energy-flow constraints, and resilience/partitioning behavior. The key property is reproducibility and interpretability for scenario evaluation.

- Emulated components (software-in-the-loop / hardware-in-the-loop / system emulation): Used where implementation realism matters more than abstraction, or where vendor stacks and protocol behaviors are hard to capture accurately in pure simulation. Emulation can represent real control-plane software, containerized network functions, or protocol stacks interacting under controlled conditions. The key property is fidelity to implementation artifacts and real interfaces.

- Learned components (data-driven surrogates): Used where the true behavior is complex, non-linear, or too expensive to simulate/emulate at the required scale (e.g., QoE perception mapping, RF propagation surrogates, anomaly likelihood models, traffic generation patterns, or policy outcome predictors). Learned modules act as fast approximators to enable efficient exploration and ranking of alternatives.

5.3.4.2 Coupling principle.

UC3 should state explicitly that the twin is composed per scenario and per evaluation goal: the same domain may be simulated in one experiment, emulated in another, or replaced with a learned surrogate when speed is required. The coupling model defines:

- Which subsystems are represented by which mode (simulate vs emulate vs learn),
- Interface contracts between subsystems (inputs/outputs, event semantics),
- Assumptions and validity boundaries for each representation mode,
- Fallback strategy when fidelity is insufficient (e.g., swap a surrogate with emulation).

5.3.4.3 Time alignment and calibration using live telemetry.

UC3 should define a standard alignment workflow to keep the twin synchronized with the operational network:

- Telemetry selection and observability contract: Identify the minimum telemetry set required to calibrate each modelled subsystem (KPIs, alarms, configuration state, topology/service-graph/slice definitions, resource states). This is expressed as a conceptual contract rather than fixed metrics.
- Time-based alignment: Establish consistent time semantics across sources (clock reference strategy, timestamp normalization, handling of reporting intervals, and event ordering). The spec should highlight that calibration requires mapping telemetry streams into a consistent timeline.
- State reconciliation: Convert observed network state into the twin’s state variables: configuration parameters, resource allocations, policies, and derived indicators. This produces a “twin snapshot” aligned to a chosen time reference.
- Parameter calibration and drift detection: Fit or update parameters of simulated and learned components so the twin reproduces observed behavior within an acceptable tolerance for the scenario’s purpose. Track drift (twin vs reality divergence) and trigger recalibration when divergence grows or when configuration changes significantly.

- Validation step prior to decision use: UC3 includes a formal "twin validity" gate: before UC3 results are used for promotion or as decision support, the twin must demonstrate sufficient alignment to reality for the targeted KPIs and time horizon. If not, UC3 outputs remain advisory only.

5.3.4.4 Artifact promotion policy

UC3 should define a controlled, auditable pathway for promoting Digital Twin outputs into operational assets. The intent is to ensure that any rApp/xApp logic, ML models, policies, or configuration packages derived from UC3 evaluation are introduced safely and traceably, with explicit rollback posture and tenant-aware governance.

Artifact scope and packaging. Promotable artifacts include (non-exhaustive) rApps for non-RT orchestration, xApps/dApps for near-RT optimization, ML models (forecasting, anomaly/RCA, surrogates), policy packages (e.g., admission, placement, adaptation, safety constraints), and configuration bundles (e.g., slice/service templates and domain-controller parameter sets). Each artifact should be versioned and packaged with metadata: intended scope, assumptions, required inputs, dependencies, operational bounds, and applicable stakeholders/owners (operator, enterprise tenant, domain owner).

Governance and authorization. Promotion should be bound to explicit authorization and ownership rules, including tenant isolation, policy precedence/conflict resolution, and the minimum telemetry/observability required to operate the artifact safely. Where multiple controllers may act on the same resource, UC3 should require a declared control contract (who may change what, under which conditions, and with which escalation/override rules).

Validation gates (conceptual). UC3 should define gating logic rather than fixed thresholds: (i) Reproducibility and provenance gate (versioning, dependencies, dataset/telemetry lineage); (ii) Safety and policy-compliance gate (explicit constraints, invariant checks, and guardrails aligned to the architecture); (iii) Twin-based validation gate (simulation/emulation/learned surrogates as applicable, stress and corner-case exploration, and sensitivity analysis); and (iv) Operational readiness gate (monitoring plan, alerting hooks, and rollback plan).

Deployment patterns. UC3 should recommend progressive introduction patterns such as canary deployment (limited scope), staged rollout (incremental expansion), and rollback (revert to previous artifact versions). Promotion should also include a 'safe default' posture (fallback policies and conservative parameters) to ensure graceful behavior under uncertainty or missing telemetry.

Post-deployment verification and feedback. UC3 should require post-promotion monitoring that compares observed outcomes to expected behavior (as predicted by the Twin), captures deviations and unintended interactions, and feeds these observations back into (a) Twin calibration, (b) artifact refinement, and (c) future promotion decisions. This closes the loop between offline evaluation and online operation without assuming that offline results automatically transfer to the live network.

5.3.5 Alignment with the UNITY-6G proposed architecture

5.3.5.1 Architectural elements exercised

- Digital Twin Platform (DT Core + modular libraries)
  - DT Core composes libraries for RAN, Core, Transport, Edge/Cloud, AI/Analytics, Security/Privacy, and Observability.
  - Scenario runner and evaluation harness produces comparable KPI outputs for alternative configurations.
- MS + data/knowledge fabric
  - Unified ingestion of live telemetry and simulation outputs into a common, versioned schema with provenance.
  - Metadata enrichment (site/slice/service/device-version) to keep evaluation traceable and reproducible.
- AE/DE (analytics + decision)
  - AE performs KPI scoring, regression checks, sensitivity analyses, and reporting.
  - DE performs optimization and conflict resolution under capacity/policy/safety/energy constraints and emits candidate intents.
- ACT + orchestration/LCM
  - Packages validated outputs as deployable artifacts (rApps/xApps, configuration bundles, container images, model versions).

  - Deploys through controlled rollout/rollback pipelines and closes the loop using post-action telemetry.

- Security/trust/governance
  - Authorization for simulation, approval and deployment; audit trails for decisions and model/policy versions.
  - Policy compliance checks across stakeholders and safety bounds before promotion to production.

5.3.5.2 Stage-to-architecture mapping

5.3.5.2.1 Stage 1 - Modular Digital Twin Design & Setup

- Inputs: blueprint/topology, service graphs, policies/constraints, baseline inventories and telemetry.
- DT Core composes domain libraries into a scenario-ready system model; outputs a versioned DT configuration manifest.

5.3.5.2.2 Stage 2 - Scenario Simulation & Conflict Resolution

- Runs what-if scenarios (faults/traffic/mobility/energy/emergency priorities) and detects conflicts (bottlenecks, overlaps, incompatibilities).
- DE produces ranked candidate configurations/intents with predicted KPI impact.

5.3.5.2.3 Stage 3 - Performance Evaluation - KPI Assessment

- AE scores KPIs and checks regressions vs baselines/targets; outputs acceptance gates and refinement guidance.

5.3.5.2.4 Stage 4 - Safe Deployment, Live Synchronization

- ACT deploys accepted artifacts through orchestration/LCM and domain controllers (e.g., SMO/RIC surfaces where applicable).
- MS confirms live status; DT is synchronized with observed deltas to keep the baseline aligned with reality.

5.3.5.2.5 Stage 5 - Machine Learning, Service Evolution

- Trains/updates models and policies using curated simulation + live datasets; validates in DT before promotion.
- Packages/version-controls updates and feeds improved libraries/templates back into Stage 1/2 for future runs.

5.3.5.3 Non-functional considerations and governance

UC3 operationalizes the proposed UNITY-6G architecture as a Digital Twin-centric integration loop across domains. It composes modular system models, runs scenario-driven simulations and optimizations, evaluates KPIs, and synchronizes validated updates back to the live network through controlled deployment. UC3, therefore, acts as a shared platform capability that other UCs consume to validate changes and govern rollout.

5.3.5.4 Cross-use-case interactions

- Supports UC1: Pre-validates disaster playbooks and energy-aware recovery intents; quantifies survivability under stress.
- Supports UC2: Validates placement and capacity policies for edge/GPU and transport/slicing; regression-tests adaptation strategies.
- Supports UC4: Validates multi-RAT switching and QoS knob policies against time-sensitive traffic models before rollout.

5.4 UC4 - MULTI-RAT O-RAN ENABLED NPN FOR TIME-SENSITIVE INDUSTRY 4.0 APPLICATIONS, FUNCTIONAL FLOW SPECIFICATION

5.4.1 Use case definition and scope

UC4 addresses time-sensitive Industry 4.0 traffic over a Non-Public Network (NPN) that supports multiple Radio Access Technologies (multi-RAT), under O-RAN-enabled control. The target is deterministic and dependable connectivity (bounded latency/jitter, high reliability) by combining at least two radio access technologies (6G and Wi-Fi) and applying closed-loop monitoring, configuration, and traffic steering. The lifecycle focuses on: (i) deploying and configuring the multi-RAT NPN, (ii) enabling time-sensitive connectivity across the available RATs, (iii) monitoring traffic flows and network conditions in real-time, and (iv) reconfiguring and switching/steering traffic to maintain QoS/QoE under load, coverage changes, and mobility.

5.4.2 Operational lifecycle

The UC4 lifecycle progresses through four operational stages (A–D) with explicit feedback loops. A deploys the multi-RAT NPN. B establishes time-sensitive connectivity across RATs. C monitors traffic flows and detects changes. D (re)configures RATs and policies to maintain QoS, including RAT switching and traffic steering. The system loops as conditions change.

UC4 lifecycle diagram:

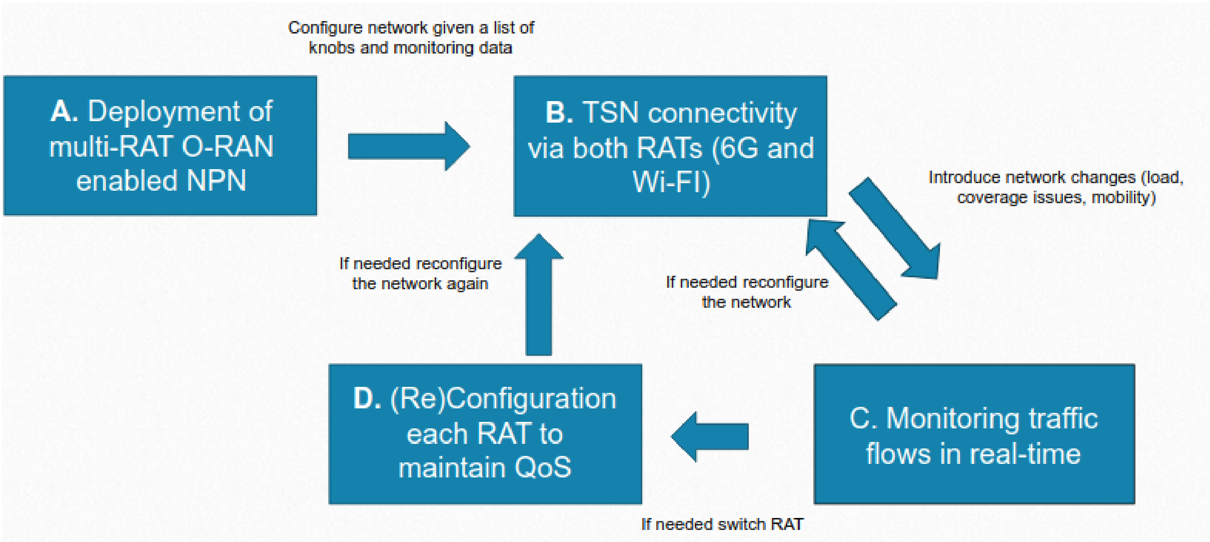


Figure 5-4. UC4 Lifecycle Diagram

5.4.3 Functional flow by operational stage

5.4.3.1 Stage A - Deployment of multi-RAT O-RAN enabled NPN

Goal: Instantiate a private industrial network with multi-RAT, O-RAN-based control, and the baseline services required for time-sensitive traffic.

Processes and steps:

- Define the industrial service requirements (e.g., TS class, maximum latency/jitter, reliability targets, traffic profiles, and critical endpoints).

- Deploy the NPN building blocks (RAN nodes for each RAT, local core / user-plane functions as needed, edge compute for control/analytics).
- Instantiate O-RAN management and control components (SMO and policy/configuration control; near-RT control for RAN tuning where applicable).
- Integrate time synchronization and determinism prerequisites (clock domain strategy, time sync distribution, admission control baseline).
- Onboard endpoints (UEs/industrial gateways) and register devices/flows into the management plane with identities and policy context.
- Establish initial QoS/QoE policies per RAT and per flow, including default preference rules and failover eligibility.
- Run baseline verification (connectivity, basic latency/jitter checks, and initial KPI collection) to validate readiness for TS operation.

5.4.3.2 Stage B - TSN connectivity via both RATs (6G and Wi-Fi)

Goal: Enable time-sensitive connectivity with redundancy across available RATs, so critical flows can survive degradation or mobility events.

Processes and steps:

- Select the TS-capable paths on each RAT (radio + transport) and bind them to the industrial application flows.
- Configure per-flow QoS treatment (prioritization, shaping, scheduling, and admission control) for both RATs.
- Establish redundancy mechanisms (active/standby or parallel paths; duplication/selection policies) and define the preferred/default RAT.
- Provision UE/industrial gateway behavior (multi-RAT client logic) for path selection and fast switchover without application disruption.
- Validate end-to-end TS performance for each RAT under nominal conditions and record baseline KPIs per flow (latency, jitter, packet loss, availability).

5.4.3.3 Stage C - Monitoring traffic flows in real-time

Goal: Continuously observe the service and network state to detect load, coverage issues, mobility effects, and QoS/SLA violations.

Processes and steps:

- Collect real-time telemetry per RAT and per TS flow (latency/jitter, packet loss, retransmissions, queue/buffer depth, scheduling KPIs).
- Monitor radio conditions and mobility indicators (signal quality, interference, handover events) and Wi-Fi-specific indicators when applicable.
- Detect network changes and anomalies (load spikes, coverage holes, interference bursts, backhaul congestion) that may threaten TS constraints.
- Correlate multi-domain telemetry (RAN, transport, edge/core) to localize the constraint (where the determinism budget is being consumed).
- Trigger reconfiguration decisions when KPIs cross thresholds or predictive indicators suggest imminent violation.

5.4.3.4 Stage D - (Re)Configuration of each RAT to maintain QoS

Goal: Apply policy and configuration changes, including RAT switching and traffic steering, to restore or preserve time-sensitive performance.

Processes and steps:

- Select corrective actions based on the detected condition: tune RAT parameters, adjust scheduling/resource allocation, or shift traffic to the alternate RAT.
- Execute RAT switch/traffic steering for the affected flows (switching preferred/active path while maintaining TS constraints and minimizing disruption).
- Reconfigure per-RAT policies (band/resource assignment, handover thresholds, transmit power, Wi-Fi channel selection, TS shaping/admission rules).
- Verify post-action KPIs and stability; if the issue persists, iterate (reconfigure again) or escalate to a higher-level policy change.

- Update policy baselines and learning/analytics artifacts so subsequent decisions improve (feed back into Stage B establishment and Stage C monitoring).

5.4.4 Operational assumptions and policy framework

5.4.4.1 Time synchronization/determinism assumptions and safe multi-RAT switching

UC4 (Multi-RAT O-RAN Enabled NPN for Time-sensitive Industry 4.0 Applications) articulates two foundational aspects that are implicit in time-sensitive networking but often underspecified in functional flows: (i) the time synchronization and determinism assumptions that make bounded latency/jitter achievable end-to-end, and (ii) a RAT switching safety framework that preserves TS constraints without causing oscillations when conditions fluctuate. The intent is to remain conceptual and vendor-neutral while providing an implementable structure.

5.4.4.2 Time synchronization and determinism assumptions

Time-sensitive Industry 4.0 applications depend on predictable delivery, which requires not only QoS enforcement but also accurate time synchronization across endpoints, network elements, and control loops. UC4 therefore define the assumed time model and how it is maintained across a multi-RAT NPN.

- Time synchronization distribution model. UC4 describes, at a minimum:
    - A time authority hierarchy (grandmaster source(s) and how time propagates across domains), including how the NPN treats multi-site deployments and edge/cloud controllers.
    - Distribution paths for time to cellular RAN elements, Wi-Fi APs, transport elements, and industrial endpoints/controllers.
    - Redundancy expectations (what happens if the primary time source is degraded; how the NPN maintains continuity under partial failures).
- Time verification and health monitoring. UC4 defines a conceptual verification loop:
    - Network elements and controllers continuously report time health indicators (e.g., synchronization state, stability/quality indicators, and alarms on loss-of-sync).
    - The monitoring/assurance loop correlates time health with TS flow performance to distinguish “radio/transport congestion” from “time instability”.

    - A policy exists that defines acceptable time health for TS operation and the operational posture when time quality degrades (e.g., restrict certain TS classes, activate fallback paths, or enter a safety mode).

- Determinism model and “TS budget” decomposition. To connect UC4 to the architecture, UC4 should define determinism as a budgeted end-to-end contract rather than a single KPI. The contract is expressed as a TS budget that is decomposed per segment, for example:
    - Endpoint/interface budget (industrial device to access point/base station),
    - Access segment budget (per RAT),
    - Transport segment budget (backhaul/midhaul, including queuing and routing constraints),
    - Core/edge processing budget (local UPF/edge processing if applicable),
    - Control/management impact budget (ensuring control actions do not destabilize runtime determinism).

This UC does not need numbers; it needs the concept that a TS flow’s acceptability is evaluated by verifying that each segment remains within its allocated portion of the overall budget, and that the architecture can observe and act on segment-level contributors.

- Architectural integration points. These assumptions map directly to the proposed architecture:
    - MS must collect time-health indicators and TS flow performance telemetry.
    - AE/DE must attribute budget consumption to segments (RAT vs transport vs edge processing) and select actions consistent with determinism.
    - ACT must enforce actions via SMO/RIC and relevant domain controllers while respecting safety constraints.

5.4.4.3 RAT switching safety

Multi-RAT TS connectivity implies that switching/steering is a powerful lever, but it can also introduce instability (ping-pong behavior) and transient disruption that violates TS guarantees.

UC4 should therefore specify a safety framework for switching that is policy-driven and flow-aware.

- Per-flow switching policy model. UC4 should define that switching decisions are made per TS flow (or flow class), not only per device, because different flows have different tolerance to disruption. A per-flow policy includes:
    - Whether the flow is switchable at runtime or must remain pinned to a RAT unless a hard failure occurs.
    - Whether switching is allowed only as make-before-break (replicate/parallelize then cut over) versus break-before-make.
    - Whether the flow supports redundancy (duplication across RATs with selection at the receiver/controller) as a default strategy.
- Hysteresis rules (conceptual). UC4 should state that switching is governed by hysteresis principles:
    - The candidate RAT must demonstrate a sustained advantage (in predicted TS budget compliance) before switching is permitted.
    - The current RAT must demonstrate sustained risk (continued inability to meet TS budget) before escalation occurs.
    - Decision logic must incorporate confidence and stability of measurements to avoid reacting to brief fluctuations.
- Hold-down / stabilization timers. UC4 should define stabilization timers conceptually:
    - After a switch, the system enters a hold-down state where further switching is restricted unless a critical condition occurs.
    - After any reconfiguration action (e.g., scheduling/QoS knob changes), the system allows a stabilization period before considering additional disruptive actions.
    - The hold-down and stabilization behaviors are configurable by policy and may differ by flow class (e.g., ultra-critical flows are more conservative).

- Safety ladder for TS assurance before switching. Switching should be positioned as part of a controlled ladder of actions:
  - Apply local RAT tuning (scheduling/resource adjustments) within the current RAT.
  - Apply transport treatment adjustments (path selection or traffic influence within allowed bounds).
  - Apply redundancy activation (parallel paths or duplication) if available and permitted.
  - Execute RAT switching only when required or demonstrably beneficial under stable evidence and within the per-flow policy envelope.
- Post-switch verification and rollback. UC4 should include a verification step:
  - Immediately after switching, MS/AE confirms that the TS budget is being met and that time health remains acceptable.
  - If post-switch behavior is worse or unstable, the system may invoke a rollback (revert to prior RAT, revert to prior configuration, or re-enter redundancy mode), governed by policy.

5.4.5 Alignment with the UNITY-6G proposed architecture

5.4.5.1 Architectural elements exercised

- Unified multi-RAT management and control via O-RAN
  - O-RAN management/orchestration (SMO) drives lifecycle and policy distribution across the NPN and the participating RAT domains.
  - RIC applications (dApps/xApps/rApps) can be used to manage both cellular RAN and O-RAN-enabled IEEE 802.11 APs under a common control model.
- Time-sensitive flow control and exposure (NEF + CAPIF/OpenCAPIF)
  - NEF-based QoS monitoring and traffic influence expose per-flow QoS status and allow authorized Application Functions (e.g., TSN controller, NPN manager, RIC apps) to request QoS adaptations for TSN/URLLC-class flows.

    - CAPIF/OpenCAPIF provides API discovery/publication, invoker/provider management, and logging/auditing so industrial AFs can securely consume exposure capabilities without hard-coded endpoints.
- Closed-loop assurance: MS, AE/DE, ACT
    - MS provides real-time telemetry per RAT and per TS flow (latency, jitter, loss, scheduling, congestion, mobility indicators).
    - Analytics/Decision (AE/DE) correlates multi-domain telemetry, detects/predicts violations, and selects actions under determinism, capacity and policy constraints.
    - ACT executes the chosen actions through SMO/RIC and domain control APIs and verifies outcomes (with rollback/safe-update support where required).
- NPN/TSN control plane integration
    - An NPN/TSN controller (or equivalent industrial controller function) binds application flows to TS-capable paths, enforces admission/shaping/scheduling policies, and triggers steering/switching policies based on exposure + telemetry.
- Digital Twin support for UC4
    - DT elements can be used to test multi-RAT knob policies and predict whether planned actions preserve TS constraints before executing at runtime.

5.4.5.2 Stage-to-architecture mapping

5.4.5.2.1 Stage A - Deployment of multi-RAT O-RAN-enabled NPN

- SMO/orchestration deploys and configures the NPN building blocks (RAN nodes per RAT, local core/UPF where needed, edge compute for control/analytics).
- O-RAN control surfaces are instantiated (SMO + RIC stack), and Wi-Fi APs are onboarded as O-RAN-manageable elements if applicable.
- Time determinism prerequisites are established (time synchronization strategy, determinism budgets, admission control baseline).

- Security and governance are applied for industrial AF onboarding and policy ownership within the NPN.

5.4.5.2.2 Stage B - TSN connectivity via both RATs (6G and Wi-Fi)

- Industrial AFs bind TS flows to TS-capable paths; per-flow QoS policies are configured across RATs and transport.
- Exposure plane is prepared: NEF QoS monitoring/traffic influence is enabled and registered via CAPIF/OpenCAPIF for authorized invokers.
- Redundancy policies are applied (active/standby or parallel duplication/selection), including the default/preferred RAT and switchover constraints.

5.4.5.2.3 Stage C - Monitoring traffic flows in real-time

- MS collects real-time, per-flow/per-RAT telemetry and mobility/radio indicators; SBA/service bus transports events to AE/DE and controllers.
- AE/DE correlates RAN + transport + edge/core to localize where the determinism budget is being consumed and to predict imminent violations.

5.4.5.2.4 Stage D - (Re)Configuration of each RAT to maintain QoS (switching/steering)

- AE/DE selects an action plan under strict timing constraints (e.g., scheduling/resource allocation changes, policy updates, steering, or RAT switching).
- ACT executes through RIC apps (xApps/dApps/rApps), SMO workflows, and exposure APIs (NEF traffic influence), then confirms post-action KPIs; iterates or rolls back if needed.
- Logging/auditing from CAPIF/OpenCAPIF supports SLA verification and troubleshooting for industrial stakeholders.

5.4.5.3 Non-functional considerations and governance

- Determinism: actions and telemetry loops must preserve bounded latency/jitter for critical flows; control loop timing should support near-real-time adaptation where required.

- Multi-tenancy and governance: NPN policies must clearly define which industrial AFs can subscribe/invoke exposure and which can influence QoS.
- Security: API invokers/providers are authenticated/authorized; configuration changes are auditable; device onboarding is controlled.
- Safety: RAT switching/steering must be constrained to avoid oscillations; fallback policies must ensure safe industrial operation.

5.4.5.4 Cross-use-case interactions

UC4 validates how the UNITY-6G architecture supports deterministic, low-latency and highly reliable Industry 4.0 communications over a NPN using multi-RAT access (e.g., cellular 6G plus IEEE 802.11 Wi-Fi) under a unified O-RAN-enabled control plane. The architecture contribution is the ability to (i) manage both RAT domains consistently, (ii) expose per-flow QoS state and allow controlled traffic influence for time-sensitive flows, and (iii) close the loop fast enough to preserve bounded latency/jitter under load, coverage and mobility dynamics.

5.5 OVERALL ARCHITECTURE PERSPECTIVE ACROSS UC1-UC4

Across UC1–UC4, the proposed architecture is exercised as an AI-native, service-based closed loop spanning the edge–cloud continuum. The shared pattern is: (i) MS collects multi-domain telemetry, (ii) AE correlates, predicts, and scores KPI/QoE outcomes, (iii) DE selects intents under policy and safety constraints, and (iv) ACT enforces decisions through orchestration and domain controllers (e.g., SMO/non-RT RIC, near-RT RIC, transport/core controllers). Where applicable, a Digital Twin (UC3 explicitly; UC1/UC2/UC4 optionally) provides 'what-if' evaluation and pre-validation of changes before rollout.

5.5.1 Architecture responsibilities

5.5.1.1 Service lifecycle and placement (edge–cloud continuum)

- Orchestration/LCM places workload where it best meets latency, energy, and resilience constraints: UC1 prioritizes survivability; UC2 prioritizes proximity and GPU capacity; UC3 prioritizes safe promotion of changes; UC4 prioritizes determinism.

- Exposure and APIs (northbound consumption): Standardized exposure (NEF/CAPIF-style) provides capability discovery and controlled consumption of QoS, location, and configuration actions by UC-specific service managers.
- Assurance and closed-loop control: MS–AE–DE–ACT forms the runtime control loop, while governance prevents oscillations and enforces stakeholder priorities (public safety vs consumer vs industrial).
- Trust, security, and governance: Identity, attestation, authorization, and auditing ensure that only authorized actors can influence QoS and deploy artifacts, and that decisions remain traceable.

6 CONCLUSIONS AND NEXT RELEASE

6.1 CONCLUSIONS

The first release of the UNITY-6G architecture presented in this deliverable establishes a unified, AI-native, service-based management and orchestration framework across heterogeneous 6G domains. It consolidates insights from state-of-the-art initiatives, 3GPP, TM Forum ODA/ANP, aerios continuum, and emerging NTN architectures, and integrates them into a coherent, multi-layered and multi-domain architectural blueprint.

The architecture introduces a consistent cross-domain service-based management bus (SBMA) that enables federated orchestration across Core, O-RAN, NTN, Transport, Non-3GPP, AI/ML, and Vertical Application domains. Central to this framework are the IDMO, DMOs, and IDMs, which together ensure seamless lifecycle management of services and functions. The AI-native paradigm, built on the MS–AE–DE–ACT tuple, provides a unifying intelligence model across domains, supporting proactive optimization, closed-loop control, and distributed learning.

Furthermore, the integration of semantic communications, sustainability mechanisms (VPP and energy intelligence), and a strong trust and security layer ensures that UNITY-6G addresses key 6G drivers such as energy efficiency, resilience, openness, and multi-stakeholder governance.

The architecture is also aligned conceptually with the four representative UNITY-6G use cases. These use cases collectively demonstrate that the architecture supports end-to-end automation, multi-domain interoperability, deterministic operation, and intelligent service lifecycle management.

Overall, this first release defines a strong foundation for achieving unified, AI-native 6G management and orchestration, while identifying key areas that will evolve toward the final architecture in D2.4.

6.2 PLANNED ENHANCEMENTS FOR NEXT RELEASE

The next release of the UNITY-6G architecture (D2.4) will focus on consolidating and refining the cross-domain orchestration framework. This includes a more detailed definition of the operational interplay between the IDMO and the various DMOs, as well as the completion of per-domain service catalogues, descriptors, and dependency models that reflect implementation feedback from WP3 and the PoCs in WP6.

A significant emphasis will also be placed on extending the AI-native control framework. The final version will incorporate a fully detailed model of tuple deployments, centralized, distributed, and hierarchical, and will specify domain-specific AI agents with clear decision scopes and authority boundaries. Support for multi-agent cooperation, reinforcement learning, and conflict-aware reasoning will be enhanced, connecting tightly with the AI/ML subsystem's training, inference, and evaluation pipelines.

The Data Continuum and Digital Twin components will also be expanded. The next release will unify telemetry schemas, governance policies, and minimal observability contracts across all domains while finalizing the coupling between simulated, emulated, and surrogate models within the Digital Twin. Full alignment between Digital Twin outputs and operational decision-making in the orchestration layer will be detailed, ensuring that system optimizations can be validated safely before deployment.

In the domain of sustainability and energy awareness, the architecture will incorporate more detailed coordination between the Virtual Power Plant (VPP) and orchestration layers. This will reinforce UNITY-6G's ambition to treat energy as an optimization objective.

The trust, security, and governance layer will also be significantly matured. This includes completing identity, attestation, and authorization flows across domains, integrating Level-of-Trust metrics directly into orchestration and decision engines, and finalizing the use of distributed ledger technologies for auditability and trust-claim exchange.

Finally, the architecture will be strengthened through insights obtained from early PoC deployments. Lessons learned from UC1 to UC4 will be incorporated to close architectural gaps, enhance the practical applicability of the proposed framework, and validate interoperability requirements. These improvements will help ensure that the final UNITY-6G architecture is practical, reliable, and grounded in real operational needs.

REFERENCES

[1] "5G-PPP, “MonB5G: Zero-Touch Network and Service Management with End-to-End SLAs,” 5G Infrastructure Public-Private Partnership. [Online]. Available: https://5g-ppp.eu/monb5g/. [Accessed: Mar. 31, 2026]," [Online].

[2] "Eclipse Foundation, “Eclipse aeriOS,” *Eclipse IoT Project*. [Online]. Available: https://projects.eclipse.org/projects/iot.aerios. [Accessed: Mar. 31, 2026].," [Online].

[3] “6G Study Item Descriptions (SIDs) and Work Plan entries on 6G System Architecture,” *3GPP TSG SA,* 2023-2025.

[4] TMForum, "ODA Functional Architecture Guidebook (TM Forum guideline)," no. GB1022.

[5] "Blanco, Luis, et al. "AI-driven framework for scalable management of network slices." IEEE Communications Magazine 61.11 (2023): 216-222.".

[6] "EUCloudEdgeIoT, “European Cloud, Edge & IoT Continuum,” EUCloudEdgeIoT.eu. [Online]. Available: https://eucloudedgeiot.eu/. [Accessed: Mar. 31, 2026].," [Online].

[7] "D2.2-OpenContinuum-Landscape-v2-and-recommendations-1.pdf," [Online]. Available: https://eucloudedgeiot.eu/wp-content/uploads/2024/11/D2.2-OpenContinuum-Landscape-v2-and-recommendations-1.pdf.

[8] aerOS Project, “Deliverable 2.7: aerOS Architecture,” aerOS, 2025. [Online]. Available: https://aeros-project.eu/wp-content/uploads/2024/07/D2.7.pdf. [Accessed: Mar. 31, 2026]..

[9] "3GPP, "TS 23.501 V18.x.x, System Architecture for the 5G System (5GS)," 3rd Generation Partnership Project, Rel. 18, 2026.".

[10] F. &. H. X. &. L. S. Guillemin, "5G Evolutions for a Smooth Migration to 6G: The Way Forward for the Core Network.," *4th International Conference on 6G Networking (6GNet),* pp. 131-135, 2025.

[11] 3GPP, "The Principles and Scope of 6g System Architecture and Core Network Studies," no. 3GPP-6GWS-250014, 2025.

[12] CAMARA, "CAMARA Open Gateway, API specifications and documentation," p. https://camaraproject.org/.

[13] "TM Forum, "IG1358 Intent Based Operation User Guide v1.1.0," TM Forum, Nov. 2025.".

[14] E. Zeydan, L. Blanco, C. J. Vaca-Rubio, M. Caus and K. Dev, "Semantic Non-Terrestrial Communications With Open RAN-Enabled 6G," in IEEE Communications Standards Magazine, vol. 9, no. 4, pp. 72-78, Dec. 2025, doi: 10.1109/MCOMSTD.2025.3606165..

[15] 6G-GOALS project, „Deliverable D2.5 - Initial 6G-GOALS Architecture and Operation Analysis", Zenodo, Juni 2025. doi: 10.5281/zenodo.18160889..

[16] A. Casparsen, V. -P. Bui, S. R. Pandey, J. J. Nielsen and P. Popovski, "Experimental Study of Low-Latency Video Streaming in an ORAN Setup With Generative AI," in IEEE Networking Letters, doi: 10.1109/LNET.2026.3653203..

[17] J. Deng, S. F. Hasan, H. Zhou, S. Al-Ahmadi, M. -S. Alouini and D. B. D. Costa, "AI-Native Open RAN for Non-Terrestrial Networks: An Overview," in IEEE Open Journal of the Communications Society, doi: 10.1109/OJCOMS.2026.3656366..

[18] A. Arnaz, J. Lipman and M. Abolhasan, "Transformer xApps for Semantic URLLC in Open RAN," in IEEE Open Journal of the Communications Society, vol. 7, pp. 802-820, 2026, doi: 10.1109/OJCOMS.2026.3652531..

[19] B. Buzcu, L. Finarelli, E. Zeydan, J. Baranda, N. Tzanis, N. Stamatiou, N. Kanakaris and G. Rizzo, "Energy-aware multi-domain orchestration architecture for sustainable 6G networks," in *21st Wireless On-demand Network systems and Services Conference (WONS 2026)*, Crans-Montana, Switzerland, 2026, March 2–4.

[20] ETSI, "ETSI TS 128 533 V16.6.0, 5G; Management and orchestration; Architecture framework (3GPP TS 28.533 version 16.6.0 Release 16)," European Telecommunications Standards Institute, Jan. 2021. [Online]. Available: https://www.etsi.org/deliver/etsi_ts.

[21] Blanco, Luis, et al. "UNITY-6G: UNified archITecture for Open RANenabled Distributed, Scalable and SustainabilitYenhanced 6G Networks." 2025 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit). IEEE, 2025..

[22] C. Adamczyk and A. Kliks, "Conflict Mitigation Framework and Conflict Detection in O-RAN NearRT RIC," IEEE Communications Magazine, vol. 61, no. 12, pp. 199–205, 2023..

[23] C. Adamczyk and A. Kliks "Detection and mitigation of indirect conflicts between xApps in Open Radio Access Networks," in IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2023, pp. 1–2..

[24] M. A. Shami, J. Yan, and E. T. Fapi, "O-RAN xApps Conflict Management using Graph Convolutional Networks," 2025, doi:10.48550/arXiv.2503.03523. [Online]. Available: https://arxiv.org/abs/2503.03523.

[25] A. E. Giannopoulos, S. T. Spantideas, G. Levis, A. S. Kalafatelis, and P. Trakadas, "COMIX: Generalized Conflict Management in O-RAN xApps—Architecture, Workflow, and a Power Control Case," IEEE Access, vol. 13, pp. 116 684– 116 700, 20.

[26] A. Mekrache, A. Ksentini, and C. Verikoukis, “Intentbased management of next-generation networks: an llm-centric approach,” IEEE Network, vol. 38, no. 5, pp. 29–36, 2024..

[27] M. Polese, L. Bonati, S. D’Oro, S. Basagni and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," in IEEE Communications Surveys & Tutorials, vol. 25, no. 2, pp. 1376-1411, Secondquarter 2023, doi:.

[28] O-RAN Working Group 4, “O-RAN Fronthaul control, user and synchronization plane specification 7.0,” O-RAN, Alfter, Germany, document ORAN-WG4.CUS.0-v07.00 Technical Specification, Jul. 2021.

[29] O-RAN Working Group 3, “O-RAN near-RT RAN intelligent controller near-RT RIC architecture 2.00,” O-RAN, Alfter, Germany, document O-RAN.WG3.RICARCH-v02.00, Mar. 2021.

[30] O-RAN Working Group 2, “O-RAN non-RT RIC architecture 1.0,” O-RAN, Alfter, Germany, document O-RAN.WG2.Non-RT-RIC-ARCH-TS-v01.00 Technical Specification, Jul. 2021.

[31] Software Radio Systems. (n.d.). OCUDU. Retrieved February 9th, 2026, from https://srs.io/building-ocudu-the-linux-of-ran/.

[32] Software Radio Systems. (n.d.). srsRAN Project. Retrieved February 9th, 2026, from https://www.srsran.com/5g.

[33] Software Radio Systems. (n.d.). srsRAN Enterprise 5G. Retrieved February 9th, 2026, from https://srs.io/srsran-enterprise-5g/.

[34] Eager white paper 'Architectures, services, and technologies towards 6G Non-Terrestrial Networks' https://www.eagerproject.eu/wp-content/uploads/sites/82/2023/02/EAGER_White_Paper_long_version_v3.pdf.

[35] 5G STARDUST 'D3.1 System Requirements Analysis and Specifications' https://www.5g-stardust.eu/wp-content/uploads/sites/97/2023/08/5G-STARDUST-D3.1_v.1.3.F.pdf.

[36] 6G-NTN 'Report on 3D multi layered NTN architecture (3rd version)' 6g-ntn.eu/wp-content/uploads/sites/94/2025/06/6G-NTN_D3.6_r1_v00.pdf.

[37] A. Farrel et al., 'A Framework for Network Slices in Networks Built from IETF Technologies', no. 9543. RFC Editor, Mar-2024..

[38] ETSI, "ETSI TeraFlowSDN (TFS)," European Telecommunications Standards Institute. [Online]. Available: https://tfs.etsi.org/. [Accessed: Mar. 31, 2026]..

[39] 3GPP, "TS 29.522 V19.5.0, 5G System; Network Exposure Function Northbound APIs; Stage 3," 3rd Generation Partnership Project, Dec. 2025..

[40] M. T. Delgado, 'Eclipse aeriOS | projects.eclipse.org'. Accessed: Feb. 13, 2026. [Online]. Available: https://projects.eclipse.org/projects/iot.aerios.

[41] Rafael Vaño, Ignacio Lacalle, and Carlos E. Palau. 2025. Federation of distributed domains in the Cloud-Edge-IoT Continuum. In Proceedings of the 2nd International Workshop on MetaOS for the Cloud-Edge-IoT Continuum (MECC '25). Association for Computing M.

[42] Guidotti, Alessandro, et al. "Role and evolution of non-terrestrial networks toward 6G systems." IEEE Access 12 (2024): 55945-55963..

Co-funded by the European Union

[43] Rubaltelli, Chiara, et al. "Multi-Band Wireless Access-and-Backhaul (WAB) for 5G: Implementation and Experiments." arXiv preprint arXiv:2511.16259 (2025)..

[44] 6G-NTN consortium, “6G-NTN Project”. [Online]. Available: https://www.6g-ntn.eu/about-6g-ntn/. [Accessed: Mar. 31, 2026]..

[45] 3GPP TR 38.799 v19 “Study on additional topological enhancements for NR (Release 19)”.