This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101069732

aerOS

EUROPEAN IOT-EDGE-CLOUD

D4.3 - Software for delivering intelligence at the edge final release

Deliverable No.	**D4.3**	**Due Date**	**28-FEB-2025***
Type	**Other**	**Dissemination Level**	**Public**
Version	**1.0**	**WP**	**WP4**
Description	*Intermediate release of design and implementation of building blocks and their relationship with the rest of the architecture.* **The due date has been requested to be shifted to M31(31-MAR-2025) in the on-going amendment to the Grant Agreement.*		

Copyright

The aerOS consortium consists of the following 27 partners:

UNIVERSITAT POLITÈCNICA DE VALÈNCIA	ES
NATIONAL CENTER FOR SCIENTIFIC RESEARCH "DEMOKRITOS"	EL
ASOCIACION DE EMPRESAS TECNOLOGICAS INNOVALIA	ES
TTCONTROL GMBH	AT
TTTECH COMPUTERTECHNIK AG (*Linked third party*)	AT
SIEMENS AKTIENGESELLSCHAFT	DE
FIWARE FOUNDATION EV	DE
TELEFONICA INVESTIGACION Y DESARROLLO SA	ES
UNIVERSIDAD POLITÉCNICA DE MADRID	ES
ORGANISMOS TILEPIKOINONION TIS ELLADOS OTE AE - HELLENIC TELECOMMUNICATIONS ORGANIZATION SA	EL
EIGHT BELLS LTD	CY
INQBIT INNOVATIONS SRL	RO
FOGUS INNOVATIONS & SERVICES P.C.	EL
L.M. ERICSSON LIMITED	IE
SYSTEMS RESEARCH INSTITUTE OF THE POLISH ACADEMY OF SCIENCES IBS PAN	PL
ICTFICIAL OY	FI
INFOLYSIS P.C.	EL
PRODEVELOP SL	ES
EUROGATE CONTAINER TERMINAL LIMASSOL LIMITED	CY
TECHNOLOGIKO PANEPISTIMIO KYPROU	CY
DS TECH SRL	IT
GRUPO S 21SEC GESTION SA	ES
JOHN DEERE GMBH & CO. KG*JD	DE
CLOUDFERRO SP ZOO	PL
ELECTRUM SP ZOO	PL
POLITECNICO DI MILANO	IT
MADE SCARL	IT
NAVARRA DE SERVICIOS Y TECNOLOGIAS SA	ES
SWITZERLAND INNOVATION PARK BIEL/BIENNE AG	CH

Disclaimer

Authors

Name	Partner	e-mail
Rafael Vaño	P01 UPV	ravagar2@upv.es
Salvador Cuñat	P01 UPV	salcuane@upv.es
Ignacio Lacalle	P01 UPV	iglaub@upv.es
Fernando Boronat	P01 UPV	fboronat@dcom.upv.es
Vasilis Pitsilis	P02 NCSRD	vpitsilis@iit.demokritos.gr
Andreas Sakellaropoulos	P02 NCSRD	asakellaropoulos@iit.demokritos.gr
Harilaos Koumaras	P02 NCSRD	koumaras@iit.demokritos.gr
Ignacio Domínguez	P07 TID	ignacio.dominguezmartinez@telefonica.com
Lucía Cabanillas	P07 TID	lucia.cabanillasrodriguez@telefonica.com
Luis Bellido Triana	P07a UPM	luis.bellido@upm.es
David Martínez García	P07a UPM	david.martinezgarci@upm.es
Ioannis Chouchoulis	P10 IQB	giannis.chouchoulis@inqbit.io
Vasiliki Maria Sampazioti	P10 IQB	vasiliki.maria.sampazioti@inqbit.io
Aristeidis Farao	P10 IQB	aris.farao@inqbit.io
Joseph McNamara	P12 LMI	joseph.mcnamara@ericsson.com
Zofia Wrona	P13 SRIPAS	zofia.wrona@ibspan.waw.pl
Przemysław Hołda	P13 SRIPAS	Przemyslaw.Holda@ibspan.waw.pl
Wiesław Pawłowski	P13 SRIPAS	Wieslaw.Pawlowski@ibspan.waw.pl
Paweł Szmeja	P13 SRIPAS	Pawel.Szmeja@ibspan.waw.pl
Katarzyna Wasielewska-Michniewska	P13 SRIPAS	Katarzyna.wasielewska@ibspan.waw.pl
Yan Chen	P14 ICTFI	yan.chen@ictficial.com
Tarik Taleb	P14 ICTFI	tarik.taleb@ictficial.com
Amine Taleb	P14 ICTFI	amine.taleb@ictficial.com
Vaios Koumaras	P15 INF	vkoumaras@infolysis.gr
Pantelis Papachronis	P15 INF	ppapachronis@infolysis.gr
Eugenia Vergi	P15 INF	evergis@infolysis.gr
Eduardo Garro Crevillén	P16 PRO	egarro@prodevelop.es
Adrián Ramos Ureña	P16 PRO	aramos@prodevelop.es
Álvaro Martínez Romero	P16 PRO	amromero@proistevelop.es
Riccario Leoni	P19 DS TECH	r.leoni@dstech.it
Federico Corazza	P19 DS TECH	f.corazza@dstech.it

History

Date	Version	Change
10-01-2025	0.1	Final table of contents
09-01-2025	0.2	Start first round of contributions
17-01-2025	0.5	Finalized first round. Merging document with first round contributions.
18-02-2025	0.6	End second round of contributions
26-02-2025	0.8	Version ready for round of internal reviews
17-03-2025	0.9	Version ready for PIC review
20-03-2025	1.0	Final submitted version

Key Data

Keywords	Semantics, data fabric, data governance, AI, explainability, analytics, trustworthiness, management, federation, portal
Lead Editor	P13 SRIPAS – Katarzyna Wasielewska-Michniewska
Internal Reviewer(s)	SIEMENS CF

Executive Summary

The present deliverable D4.3 “Software for delivering intelligence at the edge final release” is the last of the three deliverables outlining the final outcomes of the aerOS Work Package 4 tasks. It encompasses and updates outcomes from the preliminary release (D4.1) and intermediate release (D4.2).

The document continues with the concept of **aerOS Minimum Viable Product (MVP)** that was further developed into its second version (**MVPv2**) and addresses it from the standpoint of WP4. The aerOS MVPv2 brings together the technologies provided by the WP3 and WP4 activities under the WP2 architecture design specifications, to deliver a functional, stable prototype of the aerOS stack. Specifically, WP4 tasks contribute to the MVPv2 by enabling the sharing of data, for a trusted and decentralized AI-based orchestration of the resources in the aerOS continuum as well as implemented use case, and providing enhanced functionalities of aerOS Management Portal and management services.

IMPORTANT: This deliverable is of type OTHER. This means that D4.3 is mostly a software deliverable. While this document reports the advances of tasks T4.1-T4.6 in the period M19-M30, it must be understood together with the software release that is uploaded alongside it.

D4.3 presents the final design of the building blocks involved in the WP4 tasks, along with their respective implementations. Building upon the outcomes summarized in D4.1 and D4.2, the document is structured around the WP4 tasks to report on their results and detail their impact on the realization of the aerOS MVPv2. These can be summarised as follows:

- **Data homogenization** task T4.1, brings two main building blocks for semantic data processing: the Semantic Annotator and the Semantic Translator. Both components, which are essential for ensuring the semantic interoperability of data, have been integrated into the aerOS ecosystem. In this regard, the Linked Open Terms (LOT) methodology for ontology development has been continued and applied for creating an ontology that enables the orchestration of the continuum. The semantic tools are integrated with the Data Fabric and can be used to build data pipelines.
- **Data governance** task T4.2, has consolidated the definition of a (semantic) data product. Building upon this definition, the architecture of the aerOS Data Fabric was specified and the building blocks that compose it have been integrated. Among these blocks, the Data Product Pipeline and the Data Product Manager have been implemented to facilitate the creation of the data products by their owners.
- **Decentralized frugal AI** task T4.3 is realized using the AI Local Executor and AI Task Controller services that can be deployed on the aerOS infrastructure. Explainability for an aerOS use case based on reinforcement learning has been implemented and an approach to include explainability as a service has been proposed. Frugal techniques have been studied for their applicability in aerOS-like scenarios.
- The **Embedded Multiplane Analytics** task T4.4 has finalized an implementation of an engine that includes templates for creating user-defined functions, in addition to a set of pre-packaged functions based on popular data science libraries.
- In the **Trustworthiness and decentralized trust management** task T4.5 trust score calculation algorithm has been enhanced to include new sub-scores, as well as adjust the overall calculation process. With respect to secure communication, a deployment of a private IOTA Tangle network has been done, and test were performed.
- Finally, the **management service es and the aerOS management portal** includes the final release of the aerOS management portal, allowing for registration of new aerOS domains. Regarding management services, the aerOS Federator has been implemented, which combined with the extended capabilities of the Orion-LD Context Broker, have enabled data sharing across aerOS domains.

Table of contents

List of tables

List of figures

List of acronyms

Acronym	Explanation
AAA	Authentication, Authorisation & Accounting
AI	Artificial Intelligence
AIGC	AI-Generated Content
API	Application Programming Interface
CNN	Convolutional Neural Network
CORS	Cross-Origin Resource Sharing
CQ	Competency Question
CRUD	Create, Read, Update, Delete
CSR	Context Source Registration
DCAT	Data Catalog Vocabulary
DevPrivSecOps	Development, Privacy, Security and Operations
EAT	Embedded Analytics Tool
EB	Entrypoint Balancer
FaaS	Function-as-a-Service
FAIR	Findable, Accessible, Interoperable, Reusable
FOAF	Friend of a Friend
FL	Federated Learning
GUI	Graphic User Interface
HLO	High-Level Orchestrator
IdM	Identity Manager
IE	Infrastructure Element
IPSM	Inter-Platform Semantic Mediator
LB	Load Balancing
LBMM	Load Balanced Min-Min
LC	Least Connection
LDAP	Lightweight Directory Access Protocol
LLO	Low-Level Orchestrator
LOT	Linked Open Terms
LOV	Linked Open Vocabularies
LSTM	Long Short Term Memory
MCDM	Multi-Criteria Decision-Making
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport

MVP	Minimum Viable Product
NAS	Neural Architecture Search
NGSI-LD	Next Generation Service Interface Linked Data
NNI	Neural Network Intelligence
OPA	Open Policy Agent
OWL	Web Ontology Language
PKCE	Proof Key for Code Exchange
Protobuf	Protocol Buffers
RDF	Resource Description Framework
RML	RDF Mapping Language
RNN	Recurrent Neural Network
RR	Round Robin
TS	Trust Score
URL/URN	Uniform Resource Locator/Name
YAML	YAML Ain't Markup Language

1. About this document

This document details the final release of the WP4 building blocks to deliver applications intelligence at the edge. Following the preliminary design provided in D4.1, extended by the intermediate version in D4.2, this deliverable proceeds in describing the final design of the components and provides the results from their implementation. Furthermore, it discusses the integration activities towards delivering the aerOS stack, including integrations with components developed by WP3.

1.1. Deliverable context

Item	Description
Objectives	**O3** (Definition and implementation of decentralized security, privacy, and trust): Design and implementation of mechanisms for data access control, trustworthiness, and decentralized trust management. **O4** (Definition and implementation of distributed AI components with explainability): Design and implementation of mechanisms to enable distributed AI with support for frugality and explainability. **O5** (Specification and implementation of a Data Autonomy strategy for the IoT edge-cloud continuum). Design, implementation, and integration of mechanisms for semantic annotation, data integration, and data governance.
Work plan	The contributions to D4.3 take input from the following tasks: • T2.2 (Formalization of use cases and requirements elicitation): Components design in WP4 are aligned with the requirements identified in the use cases. • T2.4 (DevPrivSecOps methodology specification): The implementation of components in WP4 follow the best practices defined by the DevPrivSecOps methodology. • T2.5 (aerOS architectural design, functional and technical specification): Design and integration of WP4 components align with the proposed architecture for aerOS. The outcomes of D4.3 influence the following work packages: • WP5 (integration, use case deployment, validation, evaluation, assessment): To later materialize solutions in pilot deployments. The contributions of D4.3 are coordinated with: • WP3 (infrastructure components): To define functional boundaries (e.g., networking, cybersecurity, orchestration) and interactions.
Milestones	This deliverable is the final step from WP3 towards the achievement of the milestone *MS7 – Final software components release* (M30).
Deliverables	This deliverable builds upon the baseline architecture defined by D2.6 and 2.7 (aerOS architecture). D4.3 continues from the results produced in D4.1 (Software for delivering intelligence at the edge preliminary release), which included an initial design of the WP4 building blocks and D4.2 (Software for delivering intelligence at the edge intermediate release). Additionally, this deliverable is coordinated with deliverable D3.3, which is delivered at the same time.

1.2. The rationale behind the structure

This deliverable is structured into three sections. Section 2 provides an overview of the aerOS MVPv2 from the perspective of the WP4. Section 3 includes subsections dedicated to each of the tasks within WP4 presenting the final outcomes. Finally, the document concludes with Section 4, drawing conclusions for WP4.

1.3. Outcomes of the deliverable

A set of task-specific approaches and respective technical components have been formalized in this deliverable. Formalization includes: functionality provided, components that conform each solution and used technologies and standards.

Briefly, data autonomy for homogenization addresses functionalities such as semantic data annotation and semantic translation. Data governance, provenance, traceability, and lineage covers the Data Fabric building blocks.

Decentralized frugal AI provides services to enable execution of AI tasks, proposed how to implement an explainability service and studies frugality techniques in the context of aerOS. Embedded analytics components allow to deploy functions on aerOS infrastructure according to Function-as-a-Service approach and provide output for analytics or building blocks for applications or workflows deployed on aerOS.

The main objective of the trustworthiness and decentralized trust management is to provide a high level of security in the whole aerOS ecosystem guaranteeing that malicious actions will be avoided at the highest-level degree possible.

Management services and aerOS Management Portal provide entrypoint to the aerOS Meta-OS and continuum in terms of administration, service deployment, benchmarking, user notifications.

1.4. Version-specific notes

As mentioned above, this deliverable is of type OTHER. This means that D4.3 is mostly a software deliverable. While this document reports the advances of tasks T4.1-T4.6 in the period M19-M30, it must be understood together with the software release that is uploaded alongside it.

In the compressed file that is downloaded when accessing this deliverable, the reader will be able to find two main artefacts: (i) this very document, that reflects in a narrative way the progresses achieved, and (ii) a compressed file that is, in turn, composed of several compressed GitLab repositories corresponding to the code development progress by M30.

In particular, and in order to facilitate the readability of the technical delivery, here below there is an indication of the repositories that have been included in the submission. They are structured following the task reporting that is used in this document (D4.3). This schema is also used in the submitted file. The directories contain the current advances, alongside an explanatory *README.MD* in each of them in order to describe their purpose and content.

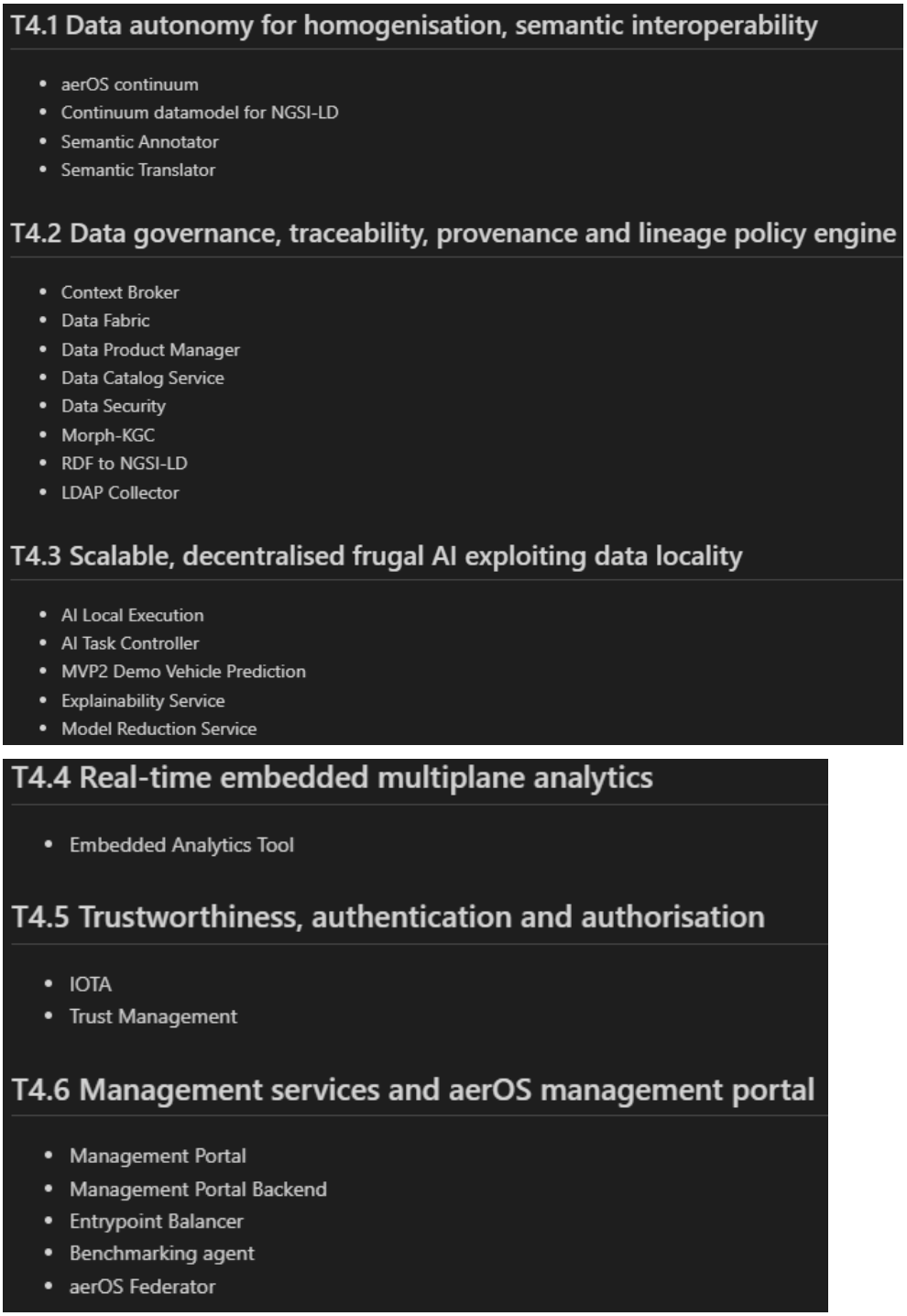


Figure 1. Software release of D4.3.

2. MVP 2 Overview

Over the course of its realization, aerOS has carefully designed an architecture aimed at providing IoT developers with a coherent environment to leverage distributed capabilities across the entire continuum. This architecture delivers a unified execution environment to support the deployment and reuse of IoT services seamlessly. With a vision to functionally unify a diverse range of computing and network resources -from cloud to edge and even IoT devices- the project has employed and integrated numerous state-of-the-art concepts and technologies.

Building upon the foundational architecture, significant advancements beyond the state of the art have been achieved by M30 through research, development, and implementation in key technical domains. These include advancements in compute and network fabric, service fabric, and data fabric, which collectively underpinned the development of new components and additional functionalities. Extending the initial MVP, which was delivered by M18, MVPv2 consolidated recent project advancements and addressed various development and integration complexities, providing an enhanced platform which can validate and demonstrate the final technological achievements of aerOS.

aerOS Meta-OS encompassed a wide area of technologies in the field of programmable networks for enhanced connectivity, resources and service management and orchestration, resilient and self-adapting runtime layers that need to be employed in order to provide the minimum for the execution environment that aerOS requires. Additionally, Cybersecurity tools and trust management are essential to ensure private and secure communications and access to services over all the aerOS continuum. All IEs and aerOS domains seamlessly expose APIs for fully defined communication among components and services. Respectively, Data Fabric technologies and integrated components are designed to support the transition from heterogeneous IoT data to a unified data fabric, and while monitoring capabilities should extract all information produced and needed for the self-adaptation of the ecosystem, analytics are foreseen to support events recognition and healing processes' triggering. Even more, AI tasks are designed to run over different Infrastructure Elements (IEs) in the continuum with optional use of frugality techniques and inclusion of explainability and interpretability.

Above mentioned technologies represent the primary aerOS technologies and tools employed to realize the continuum and all these are implemented encompassing assimilable cloud native practices to enable stakeholders to design, deploy, and operate scalable and resilient applications over the aerOS Meta-OS. The goal is to encompass cloud-native techniques naturally in continuum deployments, where infrastructure (physical and virtualized) ranges from IoT devices all the way up to cloud data centers (and not only the latter, which is the usual cloud-native case). The complex nature of the above tasks and the integration of so many diverse technologies and implementing components introduced the requirement for an iterative development which would consider and integrate early implementation evaluations, and which should optimize functionalities based on feedback emerging both from development teams and from targeted audience, i.e. IoT developers.

It is worth mentioning that addressing all the complexities and successfully achieving the project's goals could not be accomplished in a single stage. Thus, following the agile methodology of the project, a clear, staged strategy was defined and implemented. Initially, the aerOS team developed a Minimum Viable Product (MVP) by M18 to integrate the aforementioned technologies and tools into a functional prototype. By month 30, this approach has evolved further, leading to the completion of MVPv2. Building upon the insights gained from the initial MVP, the aerOS team refined the architecture concepts and expanded the platform's capabilities, addressing new use cases and challenges. MVPv2 not only realizes all the core functionalities of a Meta-OS for continuum, as designed by aerOS, but also introduces additional components and features that enhance the overall system's performance and scalability. Throughout this process, aerOS has maintained a focus on resource efficiency, validating with MVPv2 that aerOS remains a lightweight implementation while preserving the platform's core functionalities. MVPv2 also includes advanced safeguards and mitigations, enabling seamless deployment to pilot locations and allowing the team to validate and fine-tune real-world scenarios. This iterative development approach has proven invaluable in demonstrating the feasibility, viability, and effectiveness of aerOS's architecture in real and diverse environments.

While initial MVP encompassed the most compelling aerOS functionalities, MVPv2 integrates all components of architecture building blocks and is thus a valuable ecosystem for demonstrating core concepts of aerOS

architecture for a continuum Meta-OS. MVPv2 integrates two aerOS domains, which are deployed in two distinct locations, in geographic and administration terms, to demonstrate its functionality over the public cloud. Additionally, a mobile domain, although it is not a part of the continuous development and deployment process, is ad-hoc integrated when needed to exhibit the process (and its simplicity) of integrating new infrastructure and extending the continuum.

One of the core two domains, is designed to be the entrypoint domain, while the other a plain aerOS domain, which could be deployed anywhere across the continuum. The entrypoint domain is located in the common development and integration infrastructure of the project (a space provided by the partner, cloud provider, CloudFerro), while the plain one resides in the premises of the Technical Coordinator - NCSRD. This diverse topology of the MVP allows the evaluation of aerOS federation mechanisms for expanding in an agile way the aerOS continuum domains with additional/new ones, and this is the purpose of supporting a third one mobile domain which is provided with minimal legacy equipment from UPV.

Like its predecessor, MVPv2 builds upon outcomes from both WP3 and WP4 which constitute the two technical work packages of the aerOS project. WP4 undertakes all data management activities and builds the aerOS "Data Fabric", which enables the seamless integration and exploitation of a variety of data from various heterogenous resources, with the aim to support the delivery of intelligence across aerOS continuum, and over a diverse set of infrastructure resources, by optimizing usage of data without sacrificing control over it. WP4 encompasses several technologies and is related to several components in the aerOS stack. As already presented in D4.1, next figure represents the building blocks which WP4 addresses.

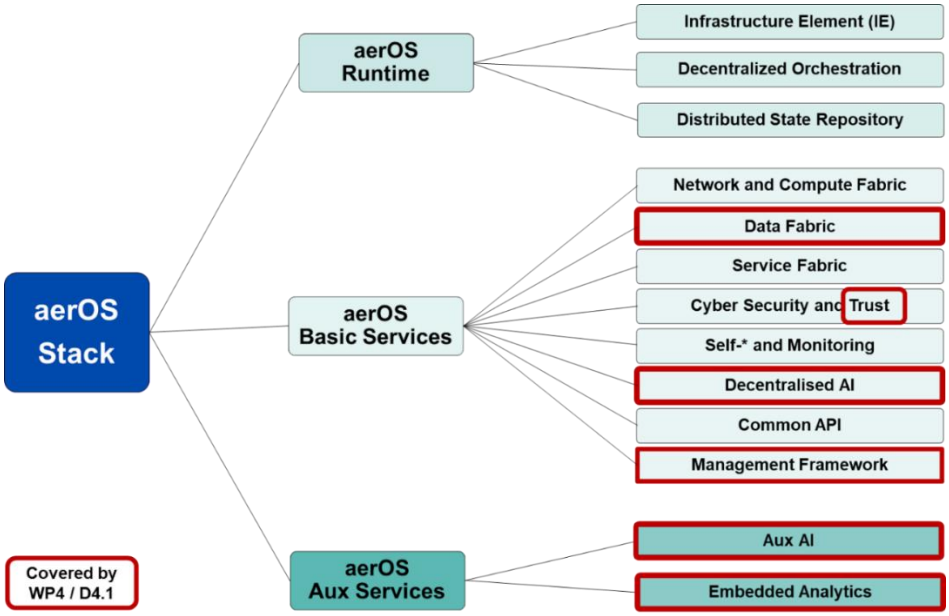


Figure 2. Technical components of WP4.

Distributed over 6 tasks, WP4 encompasses a wide range of diverse technologies, each task focusing on specific domains and their relevant technological advancements. During the initial MVPv1 development, priority was given to integrating core components essential for establishing the aerOS continuum, ensuring foundational functionalities were in place to demonstrate the system's overall viability. With the development of MVPv2, the focus has expanded significantly, integrating a broader set of functionalities that unlocks the full potential of aerOS Data Fabric and smart operations building on top of this. This phase has introduced enhanced data pipelines and data products integration, advanced data analytics, and AI-driven capabilities, allowing for more intelligent resource management and decision-making across the continuum. By incorporating these additional layers of functionality, MVPv2 moves beyond demonstrating aerOS core operations and showcases its ability to support scalable, data-driven, and AI-enhanced services across heterogeneous environments. The aerOS Portal, serving as the central point of presence, has undergone significant development, evolving into a fully integrated platform that unifies all capabilities offered across the continuum. Notably, federation mechanisms have been extended to enable seamless and automated integration of new domains, ensuring smooth onboarding

and interoperability across heterogeneous environments. These advancements reinforce aerOS ability to provide a cohesive and adaptive ecosystem for managing distributed resources efficiently.

With the aim of establishing a Data Fabric that supports federated orchestration, across the continuum, while enabling seamless integration, retrieval, and transparent reuse of IoT data across the continuum a comprehensive set of components, tools, and services has been developed, enhancing data value and interoperability. Additionally, AI-driven capabilities and advanced analytics have been integrated, unlocking new possibilities for intelligent data processing and decision-making. These advancements ensure aerOS can efficiently manage, analyze, and utilize distributed data, reinforcing its role as a powerful and adaptive continuum framework. These mechanisms have been further refined and validated by month 30, ensuring their seamless integration into existing isolated computing infrastructures and their smooth transformation into aerOS-capable domains, as exemplified by the aerOS pilot sites. Efforts during this period have been directed towards delivering:

- Appropriate ontologies and graphs which can reflect continuum state as required for orchestration decisions.
- A mesh of interconnected aerOS Context Brokers, as the aerOS federated data catalogue, able to propagate queries and data across the continuum.
- Components which can support all the chain towards ingesting and sharing interoperable data.
- AI mechanisms such as federated learning, explainability and frugality to convey a fully functional and more trustworthy service chain of AI applications.
- Tools which can interact and proxy queries towards the Data Fabric and leverage the aerOS data to provide real-time analytics.
- aerOS management portal which acts as an entry-point to the aerOS ecosystem.
- Management service to establish federation mechanisms among the multiple aerOS domains that form the continuum.

The following paragraphs provide a summary of the outcomes of each task, of WP4, which have been delivered and included in MVPv2 realization. Additionally, their relevance in establishing aerOS continuum establishment, is roughly presented.

In the second release of the MVP, the semantic interoperability tools – Semantic Annotator and Semantic Translator – have been developed (in **T4.1**) and integrated into the aerOS Data Fabric, allowing for efficient, stream-oriented semantic data processing. Additionally, aerOS continuum ontology has been created, encapsulating the essential concepts, relationships, and properties relevant to data management, processing and orchestration within aerOS-based distributed computing environment.

During this second release of the MVP, the **T4.2** task focused on consolidating the Data Catalog and Data Security building blocks. The former block has unlocked the capability for searching data products created in the Data Fabric, whereas the latter adds authorization into the search and consumption of data products. Along with the Data Product Pipeline framework introduced in MVPv1, these two blocks complete the aerOS Data Fabric.

T4.3 provided implementation of an explainability mechanism for HLO allocator that allows to analyse what attributed to the selection of specific IEs for service deployment. Additionally, federated learning, explainability and frugal AI techniques have been the focus of the task in this period. Moreover, the example analytics function consuming data from the Data Fabric was implemented and integrated within MVPv2.

The Embedded Analytics Tool from **T4.4** provides a FaaS (Function as a Service) platform for the execution of analytic and workflow operations. Exploratory Data Analysis and Prediction functions have been deployed and tested in the demonstrator accompanying the delivery of MVPv2, running over project's infrastructure supporting the visualization of data collected and processed from the Data Fabric.

During the period leading up to the second version of the MVP, the task **T4.5** focused on revising and enhancing the Trust Component architecture. The objective was to refine the trust algorithm and produce a robust Trust Score that effectively represents trustworthiness through a multi-layered approach. In the MVPv2, the calculation of Trust Score incorporates three sub-scores that evaluate the reliability, security, and reputation of

an Infrastructure Element (IE). The third aspect that enhances the proposed model is the addition of a penalty mechanism which adjusts the trust score based on the susceptibility of the IE to critical events that negatively impact its trustworthiness. The Total Trust Score (TS) thus provides a comprehensive assessment of an IE's overall trustworthiness. This trustworthiness information is securely distributed across the network via IOTA, a Distributed Ledger Technology (DLT) that ensures secure and verifiable data transactions between different nodes in the Tangle. The Tangle serves as the underlying data structure, maintaining all necessary information to track messages and guarantee the traceability of distributed payloads across the network. Both of these components are deployed across the aerOS domains.

The actions performed in **T4.6** can be summarized in two action points: (i) The improvement of the aerOS Management Portal with the pending functionalities and (ii) The implementation of the aerOS Federator along with the improvement of federation mechanisms provided by Orion-LD.

On the one hand, new features have been implemented in the Management Portal: a real-time notifications system based on WebSockets that can be used by any service of the continuum, aerOS AAA management (users, roles and organizations) which interacts with LDAP, and aerOS Data Fabric's data products retrieval and management. Moreover, the features that were implemented for the first MVP have been fine-tuned and enhanced to improve the user experience and be aligned with the latest versions of the other aerOS Basic Services. Not only is this block limited to the Management Portal itself but also includes two new components that interact with the portal. First, the aerOS Entrypoint Balancer has been developed leveraging an Improved Weighted Least Connection load balancing algorithm to fairly distribute the service orchestration request among the HLOs of all the domains. Second, the Benchmarking tool provides a way to depict the performance of an IE in terms of CPU and RAM processing or network connectivity, along with the values of technical KPIs that are directly measurable from the data provided by the aerOS distributed state repository.

On the other hand, after the NGSI-LD federation mechanisms were successfully tested in the first MVP, thus methodology was delivered to create the needed Context Source Registrations to achieve domains federation. The custom aerOS Federator component has been implemented following this methodology and its main purpose is to automate and coordinate the federation process among the multiple aerOS domains that build the continuum. Finally, the Orion-LD context broker has been constantly improved to meet the goals of the project. For instance, its publication and subscription mechanism can now be stablished in distributed way.

3. Final implementation

3.1. Data autonomy for homogenization

The challenge of managing diverse data without centralization is addressed through the implementation of an aerOS data autonomy for homogenization solution. Specifically, data autonomy is related to the ability of homogenizing data models at the edge, i.e., to query, interoperate or prepare data to be used by other modules. In this section, two solutions to support data homogenization process in aerOS are described. They are based on the concept of semantic data annotation and semantic translation.

3.1.1. Semantic annotation

The aerOS infrastructure utilizes semantic annotation to ingest "raw data" from external sources and enrich it with appropriate semantic information, to enable further processing. The Semantic Annotator component is an enhanced and extended version of the semantic annotation enabler developed within the ASSIST-IoT project[1].

The source code is published in the aerOS official code repository, including example integrations. The annotation engine provides support for annotation of JSON, XML or CSV directly into RDF, based on transformation rules expressed in YARRML/RML, JSONPath and XPath – Figure 3.

[1] http://assist-iot.eu/

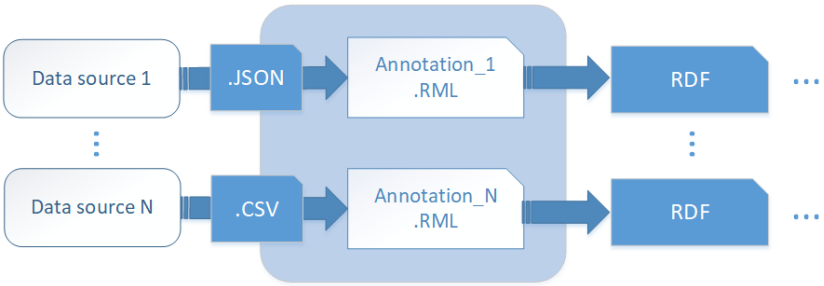


Figure 3. Semantic Annotator – data processing workflow.

The tool can perform "single-message" annotation tasks – using REST API, as well as "streaming" annotation with the help of Kafka/MQTT message brokers as mediators. The tool is designed around annotation of messages in persistent streams of data, organized into "channels". Each channel is independently configured with a CARML transformation and connects to Kafka or MQTT message brokers to ingest and output data, as well as channel monitoring messages and optional errors. REST API is exposed for the purposes of channel management. The annotator can connect to any number of brokers to consume and produce data, so cross-broker annotation channels are possible. Figure 4 depicts an overview of the components that compose the Semantic Annotator and the interactions among them.

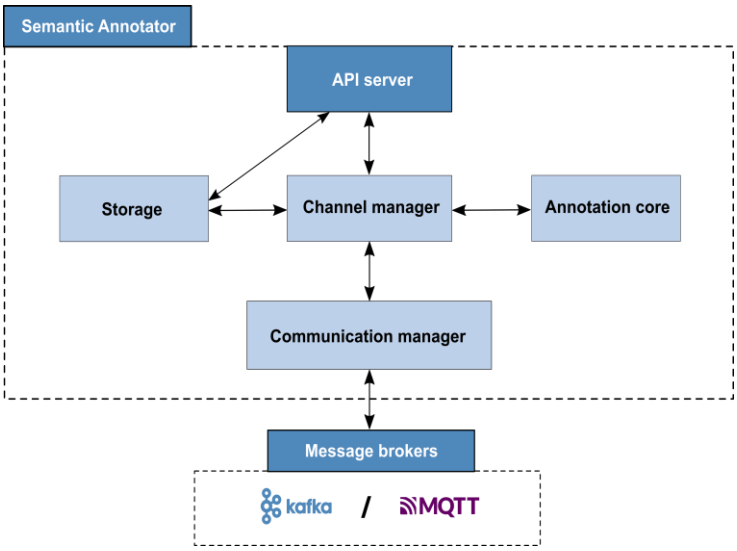


Figure 4. Semantic Annotator components overview.

3.1.1.1. Technologies and standards

Table 1. Technologies for Semantic Annotator implementation.

Technology/ Standard	Description	Component
JVM	Java Virtual Machine	All
Scala	A modern "multi-paradigm" programming language, targeting (among others) the JVM.	All
Apache Pekko	Actor-based framework for creating highly concurrent, resilient, message-driven applications.	All
Apache Pekko HTTP	HTTP request handling for configuration, reporting status, etc.	API server

Apache Pekko connectors	Apache Kafka and MQTT connectors	Communication manager
CARML	RDF Mapping library	Annotation core
MongoDB	Configuration & annotation files persistence	Storage

3.1.2. Semantic translation

The semantic translation process within aerOS is realized via the Semantic Translator component, which is based on the Inter-Platform Semantic Mediator (IPSM), a generic streaming semantic translation software developed by INTER-IoT[2], and further enhanced within the ASSIST-IoT European projects. The workflow processing diagram of the Semantic Translator, including the relevant components is depicted in Figure 5.

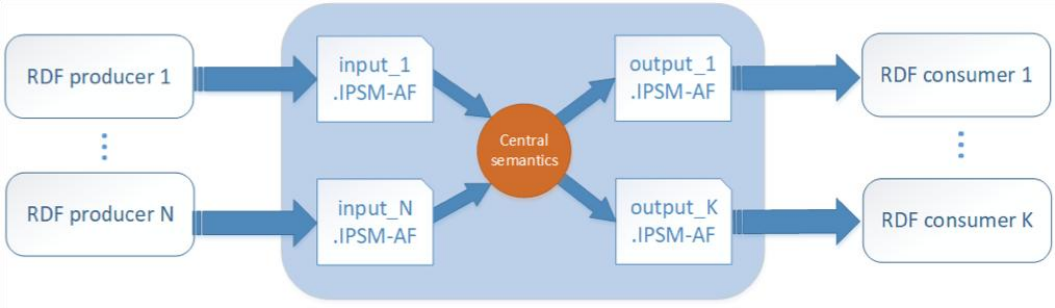


Figure 5. Semantic Translator – data processing workflow.

RDF producer is an arbitrary source of RDF data. Semantic Translator can handle many RDF producers concurrently. Using translation rules contained in the input alignment represented in the IPSM Alignment Format (IPSM-AF) it expresses the input data in terms of the "central semantics". By applying the appropriate output alignment, again expressed in IPSM-AF, the data can be offered using the consumer's preferred semantics.

More technically, Semantic Translator offers a highly-scalable, efficient translation architecture, with two main interfaces: REST and reactive streaming. The first option offers quick, one-message-at-a-time, service, while the latter is meant for handling large asynchronous streams of data.

In both cases, translation is done transactionally "per message" by applying rules defined in IPSM-AF alignment files. The overview of the architecture of the Semantic Translator component is depicted in Figure 6.

[2] https://inter-iot.eu/

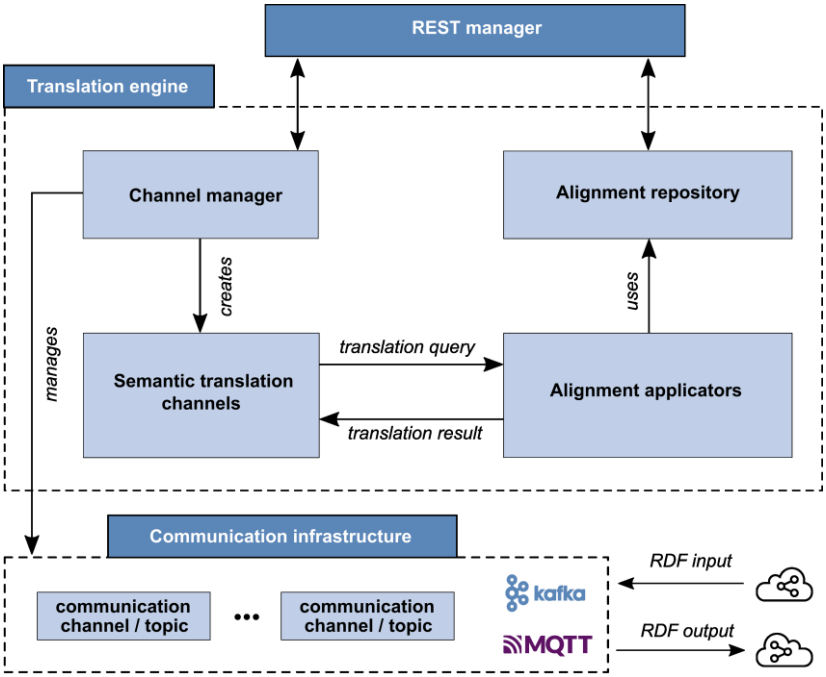


Figure 6. Semantic Translator component architecture.

Currently, the semantic translator's communication infrastructure is able to utilize two types of streaming communication channels – Kafka-Kafka and MQTT-MQTT. The component was also provided with a Kubernetes configuration, which enabled its seamless incorporation into the aerOS Data Fabric infrastructure.

3.1.2.1. Technologies and standards

The implementation of the aerOS Semantic Translator component utilizes the JVM environment, Scala 3 programming language, and Apache Pekko framework, that replaced the now commercial Akka, which was used to form the backbone of the earlier versions of the tool. The RDF translation engine, responsible for executing the translation rules expressed in the IPSM-AF uses the Apache Jena framework.

Table 2. Technologies for Semantic Translator implementation.

Technology/ Standard	Description	Component
JVM	Java Virtual Machine	All
Scala 3	The newest version of the Scala programming language	All
Apache Pekko	Actor-based framework for creating highly concurrent, resilient, message-driven applications.	All
Apache Pekko HTTP	HTTP protocol handling component for Apache Pekko	REST manager
Apache Pekko connectors	Apache Kafka and MQTT connectors	Communication infrastructure
Apache Jena	A mature, JVM-based RDF handling framework	Translation engine

3.1.3. Ontology development

3.1.3.1. Linked Open Terms (LOT) methodology

Linked Open Terms (LOT) [D1] is a lightweight methodology for developing ontologies, with special focus on industrial use cases. The LOT methodology has been applied in several European projects such as BIMERR,

DELTA, or VICINITY. Additionally, it has been followed for the development of the standard ETSI SAREF ontology and related domain-specific extensions like SAREF [D2].

The LOT methodology, which evolves from the NeOn methodology [D3], aligns the ontology development process with software development agile practices such as sprints and continuous integration. The LOT methodology iterates over a base workflow, as illustrated in Figure 7, which is composed of the following main activities: 1) Ontology requirements specification; 2) Ontology implementation; 3) Ontology publication; 4) Ontology maintenance.

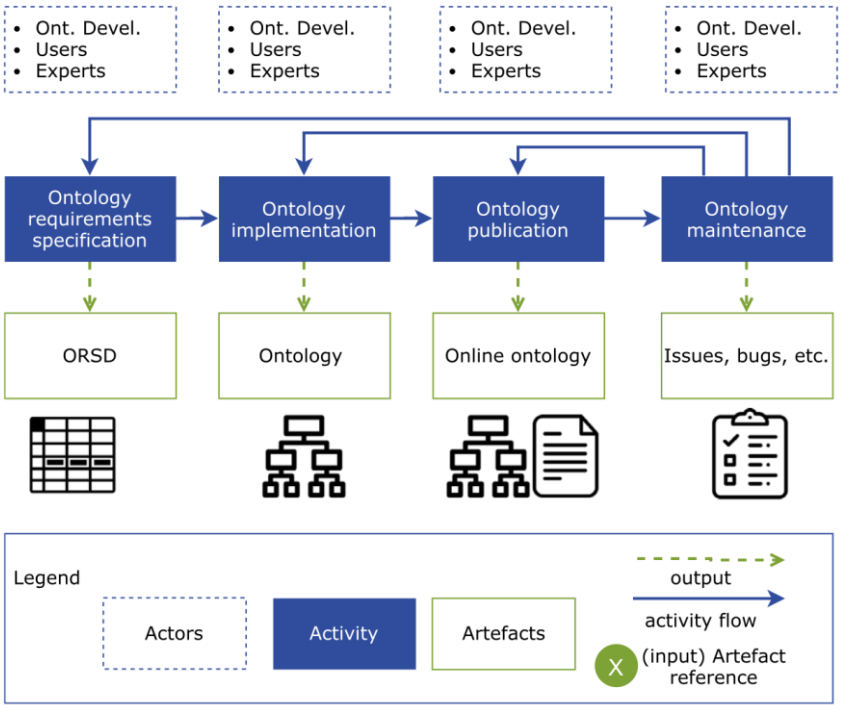


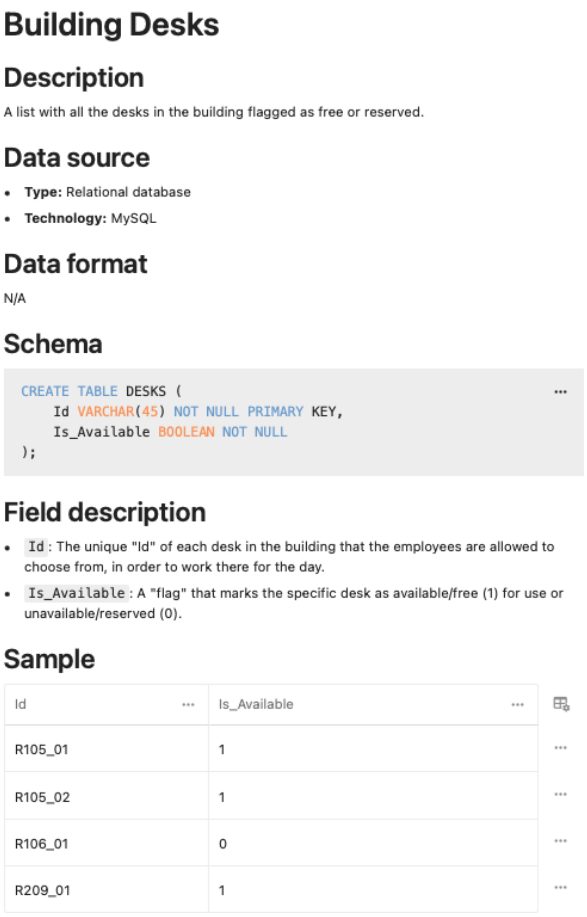
Figure 7. High-level workflow of the LOT methodology. Source [D1].

Each activity of the workflow produces an artifact that serves as input for the following activity. In addition, the methodology identifies three different roles participating in the workflow: 1) domain experts; 2) ontology developers; 3) users. On the one hand, in the scope of aerOS, the roles of domain experts and users are assigned to people involved in the implementation of aerOS internal services as well as data experts in each of the pilots. On the other hand, the role of ontology developer is performed by partners participating in T4.1.

In the following paragraphs, each LOT activity is briefly introduced, including its sub-activities and the artifacts produced, and exhibiting its aerOS implementation and the tools used.

3.1.3.1.1. First activity: Ontology requirements specification

This activity refers to the collection of requirements to be fulfilled by the ontology. To determine the requirements needed, the **use case specification** sub-activity is achieved with the collaboration from domain experts, users, and ontology developers. This sub-activity has been supported with videos calls and documents provided by the domain experts. In parallel runs the **data exchange identification** sub-activity, which focuses on collecting technical documentation about the data of the domain to be modelled, i.e., schemas, formats, standards, datasets. To help domain experts in cataloguing their datasets, a new template based on markdown language has been developed. A snapshot of a sample SQL dataset catalogued with the proposed template is shown in Figure 8. Each dataset is catalogued in a separate markdown file, which is uploaded to a GitLab repository along with the rest of the ontology artifacts.



Building Desks

Description

A list with all the desks in the building flagged as free or reserved.

Data source

- **Type:** Relational database
- **Technology:** MySQL

Data format

N/A

Schema

```
CREATE TABLE DESKS (
    Id VARCHAR(45) NOT NULL PRIMARY KEY,
    Is_Available BOOLEAN NOT NULL
);
```

Field description

- `Id` : The unique "Id" of each desk in the building that the employees are allowed to choose from, in order to work there for the day.
- `Is_Available` : A "flag" that marks the specific desk as available/free (1) for use or unavailable/reserved (0).

Sample

Id	Is_Available
R105_01	1
R105_02	1
R106_01	0
R209_01	1

Figure 8. Sample of catalogued SQL dataset.

Based on this information, the next sub-activities aim at defining and agreeing on **functional ontological requirements**. The proposal of ontological LOT offers different ways of capturing these requirements, namely competency questions (CQs), natural language statements and tabular information. Due to the technical background of domain experts (data owners) and their lack of skills in ontology querying, aerOS has followed the tabular information approach inspired by the BIMERR project [D4]. Domains experts found themselves more comfortable mapping their datasets to tables of concepts, properties, and relationships. This information has been captured in collaborative Excel spreadsheets (see Figure 9 to Figure 11) that enable iterations between the domain experts and ontology developers until the list of requirements has been completed. To ensure version control and improve readability, it is planned that future releases will migrate these Excel spreadsheets to markdown tables and upload them to GitLab.

Concept	Other names	Description
Domain		A set of one or more IEs, functionally connected and sharing a common instance of aerOS basic services among them, constituting an administrative domain able to be managed and orchestrated by aerOS Meta-OS and thus be part of the IoT-Edge-Cloud continuum.
InfrastructureElement	IE	The fundamental building block within aerOS Meta-OS. A physical or virtual computing resource providing the necessary processing power, storage capacity, and network connectivity to support containerised workloads and services. Exposes aerOS runtime on top of provided capabilities being thus the minimum execution unit within the IoT-Edge-Cloud continuum.

Figure 9. Sample "Concepts" table of ontology requirements.

Concept (must appear in "Concept list")	Property	Description	Datatype expected (Integer, boolean, string, float, list of expected values)	Max cardinality (None for unlimited cardinality)	Ordering (for properties with max cardinality above 1)	Unit of measure (if applicable)	Sensitive data (if applicable)
Domain	id	Unique identifier for the domain.	string	1..1		-	
Domain	description	Description of the domain.	string	0..1		-	
Domain	status		string	1		-	
Domain	publicUrl	The public URL associated with the domain.	string	1		-	

Figure 10. Sample "Attributes" table of ontology requirements.

Concept (must appear in "Concept list")	Relationship	Description	Target concept (must appear in "Concept list")	Max cardinality (None for unlimited cardinality)	Ordering (for properties with max cardinality above 1)
InfrastructureElement	domain	Associates an infrastructure element with a domain.	Domain	1	-
InfrastructureElement	LLO	Links an infrastructure element to a Low-Level Orchestrator.	LowLevelOrchestrator	1	-

Figure 11. Sample "Relations" table of ontology requirements.

The generation and completion of these tables concludes the ontology requirement specification activity. The LOT methodology also proposes the creation of the Ontology Requirements Specification Document (ORSD), but this sub-activity has been skipped in aerOS. Instead, all the ontology artifacts are stored together on GitLab.

3.1.3.1.2. Second activity: Ontology implementation

The goal of this activity is to build the ontology using a formal language based on the ontology requirements produced in the previous activity. First, the **ontology conceptualization** sub-activity produces a conceptual model that captures the concepts, and the relations identified in the domain of the ontology. This task typically represents this conceptual model in a diagram. In this sense, aerOS has chosen Chowlk [D5] as the standard notation and the draw.io[3] tool for representing the conceptual models. This process is conducted by the ontology developers, with optional support from domain experts and users of the use case. The resulting diagram is uploaded to the corresponding GitLab repository.

Based on the diagram of the conceptual model, the next step is the **ontology encoding** using an implementation language like OWL. The Chowlk website offers an online service that allows for bootstrapping the ontology code based on the provided diagram. Then the generated code is processed with the interactive tool Protegé to further refine the code (e.g., fix wrong labels) and to extend the code as well (e.g., adding description, language, metadata). Similarly, the ontology developer uploads the final code of the ontology to GitLab with the rest of the ontology artifacts.

Lastly, in parallel to the ontology conceptualization and encoding, the **ontology reuse** sub-activity is conducted. This task focuses on finding concepts already defined in existing ontologies that can be reused in the ontology under development. To this end, aerOS leverages the recommended Linked Open Vocabularies (LOV) service [D6], which provides a tool that can search for concepts among all registered ontologies. However, this sub-activity is still under exploration in aerOS because its implementation is challenging due to the wide variety of existing ontologies. Additionally, aerOS aims to align with standard ontologies, albeit interoperability in the aerOS Data Fabric with external data services is not a top priority.

3.1.3.1.3. Third activity: Ontology publication

This activity focuses on publishing a release candidate of the ontology by means of human-readable documentation and machine-readable files, which can be accessed online. In aerOS, we have chosen the WIDOCO open-source tool to generate HTML-based documentation from the ontology [D7]. This documentation includes further descriptions, examples, and the conceptual diagram from the previous activity.

[3] https://app.diagrams.net

Given that all ontology artifacts are uploaded to a common GitLab repository, the GitLab CI feature was leveraged to automatically generate the documentation every time a new release of the ontology is rolled out. The GitLab CI script generates the HTML code and publishes the website by making use of the GitLab Pages feature.

3.1.3.1.4. Fourth activity: Ontology maintenance

The last activity in the methodology is the maintenance of the ontology. The incorporation of new ontology requirements or the identification and fixing of bugs found in the ontology, are discussed among partners via the Mattermost messaging tool. Additionally, as mentioned before, all the ontology artifacts are uploaded to GitLab and versioned with tags, thus facilitating continuous integration and version control over the ontology.

3.1.3.2. aerOS continuum ontology

The IoT-Edge-Cloud continuum, managed by the aerOS Meta-OS, represents a distributed computing architecture where data flows seamlessly from the IoT devices at the edge of the network to a centralized cloud infrastructure. The inherent complexity of this computing continuum needs to be modelled into a data ontology as easily as possible, being understandable by humans and efficient for machine communications. In addition, there is a clear lack of existing ontologies for the computing continuum, and the minimal initiatives that have been found did not fit into the continuum conceived in aerOS. Therefore, an ontology for the IoT-Edge-Cloud continuum has been created from scratch for aerOS, inspired by some existing ontologies (e.g., FOAF [D8]) and standardization initiatives such as OASIS TOSCA[4]. This ontology, as shown in Figure 12, is intended to encapsulate the essential concepts, relationships, and properties relevant to data management, processing and orchestration within this distributed computing architecture.

The entities of this ontology can be divided into three blocks: (i) aerOS AAA, (ii) resource orchestration and (iii) service orchestration. On the one hand, the aerOS AAA block aims to represent the human side of the continuum, which is the relationship between them and the services and resources that conform the continuum through the definition of users, roles, and organizations. On the other hand, the physical computing resources (Infrastructure Elements) must be represented to show the current state of the continuum by taking advantage of the defined monitoring processes. However, conceptual layers must be added on top of them to depict the defined continuum in aerOS, such as domains and Low-Level Orchestrators. The last block represents the deployed services in the IEs of the continuum.

These entities not only represent the status of this deployed services, but also all the stages of the services orchestration process, from service requirements specifications (SLAs, minimum computing resources, etc) to execution parameters (network ports, container images, environment variables, etc). Thus, the complete relationship among all the entities of the ontology is depicted in the aerOS orchestration process, which is described in detail in section 4.3.1.1 of D3.2.

The ontology documentation was generated using WIDOCO and was published online at http://w3id.org/aerOS/continuum.

[4] https://www.oasis-open.org/committees/tosca/

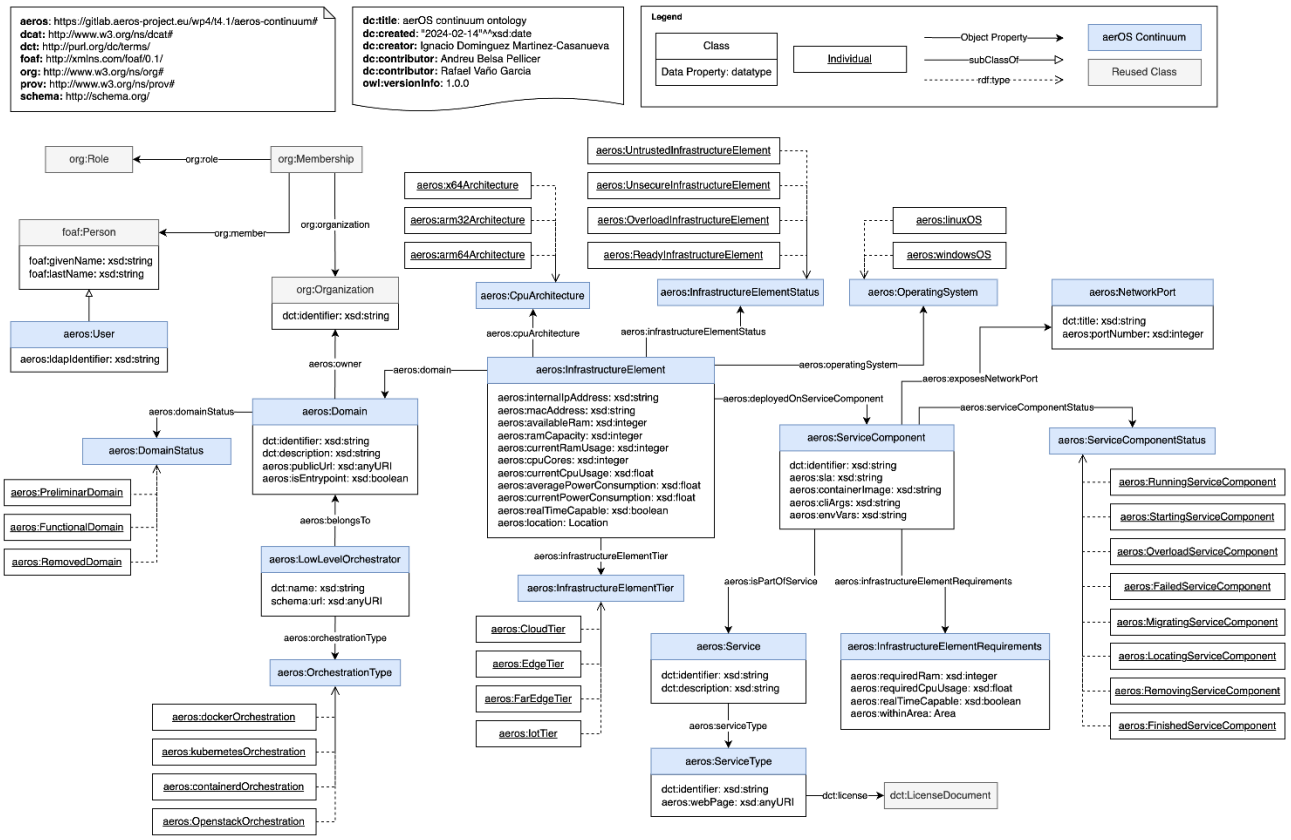


Figure 12. Conceptual model for the aerOS continuum ontology.

3.1.3.3. aerOS data catalog ontology

The aerOS Data Catalog Ontology provides a representation for registering data products created using the aerOS Data Fabric and advertised to the rest of the continuum. The ontology builds upon the standard DCAT 3.0 Ontology [D9], and extends it with concepts defined within the scope of aerOS using the `a4dcat` prefix. The concept of `dcat:Dataset`, which provides a logical representation of a collection of data, is further extended with `a4dcat:DataProduct` to explicitly capture a collection of semantically-annotated NGSI-LD data stored in an NGSI-LD Context Broker. In this sense, the `dcat:DataService` is also extended with the `a4dcat:ContextBroker` concept, to indicate a specific source of data which in the case of aerOS is the NGSI-LD Context Broker.

The ontology extends DCAT with additional concepts derived from the principles of domain ownership as per the data mesh paradigm. Data products are owned both by a particular aerOS user, but also by the organization which the user belongs to. Additionally, `a4dcat:DataProductOwner` is defined as a new role that users can take on under their organizations.

Lastly, data products can be related with glossary terms that have been formally defined in a business glossary. This idea was already envisioned and described by DCAT 3.0 and has been leverage in the scope of aerOS to improve the discovery of data products within the Data Catalog.

The ontology documentation was generated using WIDOCO and was published online at https://w3id.org/aerOS/data-catalog.

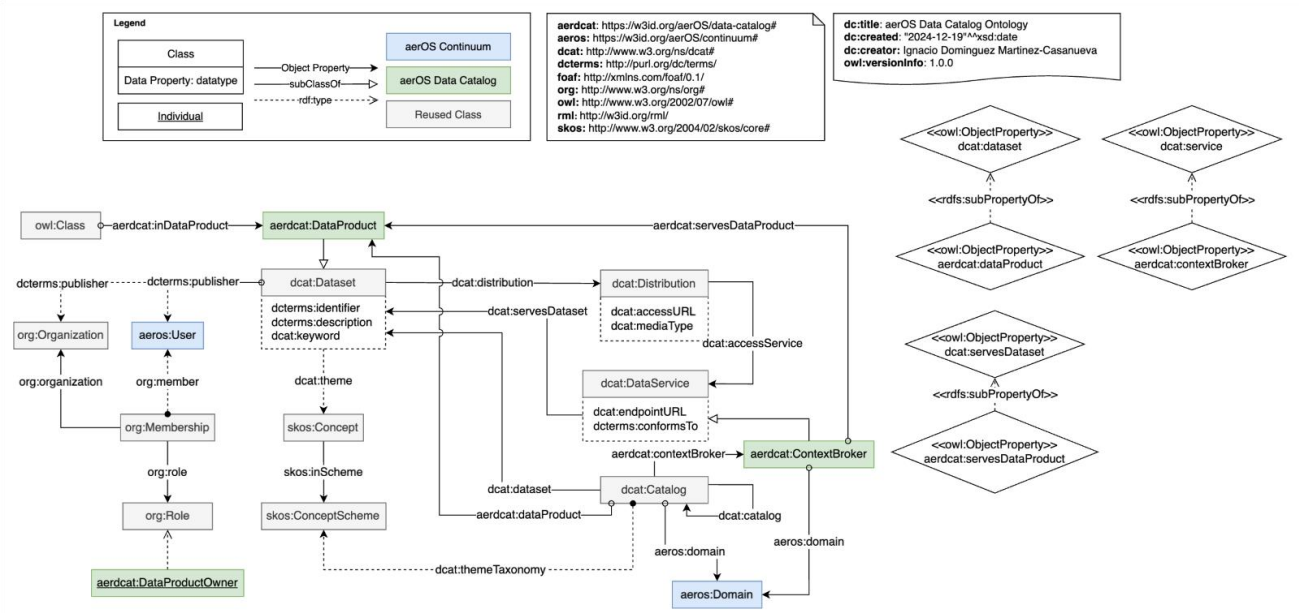


Figure 13. Conceptual model for the aerOS data catalog ontology.

3.2. Data governance, traceability, provenance, and lineage

The aerOS Data Fabric has been designed upon the data mesh principles, especially the management of data as a product. The final architecture of the Data Fabric and the components that comprise it, were constituted based on the consolidated definition of a data product according to the aerOS project.

A data product in aerOS represents the combination of data, metadata and software that make the data align with the FAIR principles. The data should be findable, keeping track of which data are available, who is accountable for them, and where they can be found (i.e., which data sources expose the data). The data must be accessible in a uniform way across the continuum, by means of a shared data fabric infrastructure, but only exposed to authorized consumers. The data must be easily interpreted (i.e., understood) by any consumer in the continuum, leveraging the semantic data models commonly agreed. Lastly, data must be reusable, avoiding ad-hoc integrations, but open and interoperable across use cases.

Therefore, based on this data product definition, the final architecture of the aerOS Data Fabric has been designed as depicted Figure 14. The diagram illustrates the high-level architecture of the Data Fabric, which is composed of the following blocks:

- **Context Broker:** Core component that maintains the Knowledge Graph of the Data Fabric. In distributed or federated scenarios where multiple Data Fabric instances are interconnected (e.g., between multiple aerOS domains), each Context Broker will provide a fragment of the global Knowledge Graph of the continuum.
- **Data Product Manager:** Main interface of the Data Fabric towards data products owners for the onboarding and registration of data products. Orchestrates the Data Product Pipeline for the creation of data products from raw datasets, and coordinates with the Data Catalog and Data Security components to govern the new data products.
- **Data Product Pipeline:** Data pipeline that transforms raw datasets into interoperable, semantic data products that become part of the knowledge graph. This pipeline is not needed for those data sources that are considered “native”, i.e., expose datasets that already follow the NGSI-LD format and are semantically annotated according to the ontologies agreed in the continuum.
- **Data Catalogue:** Maintains a registry of all the available data, their data sources, and additional governance metadata such as ownership or domain. Collects metadata from external sources such as

Identity Management systems but also receives governance input from the data product owners through their interactions with the Data Fabric services.

- **Data Security:** Implements access control policies for new data products based on the security requirements indicated by the respective data product owner. Coordinates with cybersecurity tools from aerOS stack to enable authentication and authorization in the Data Fabric.

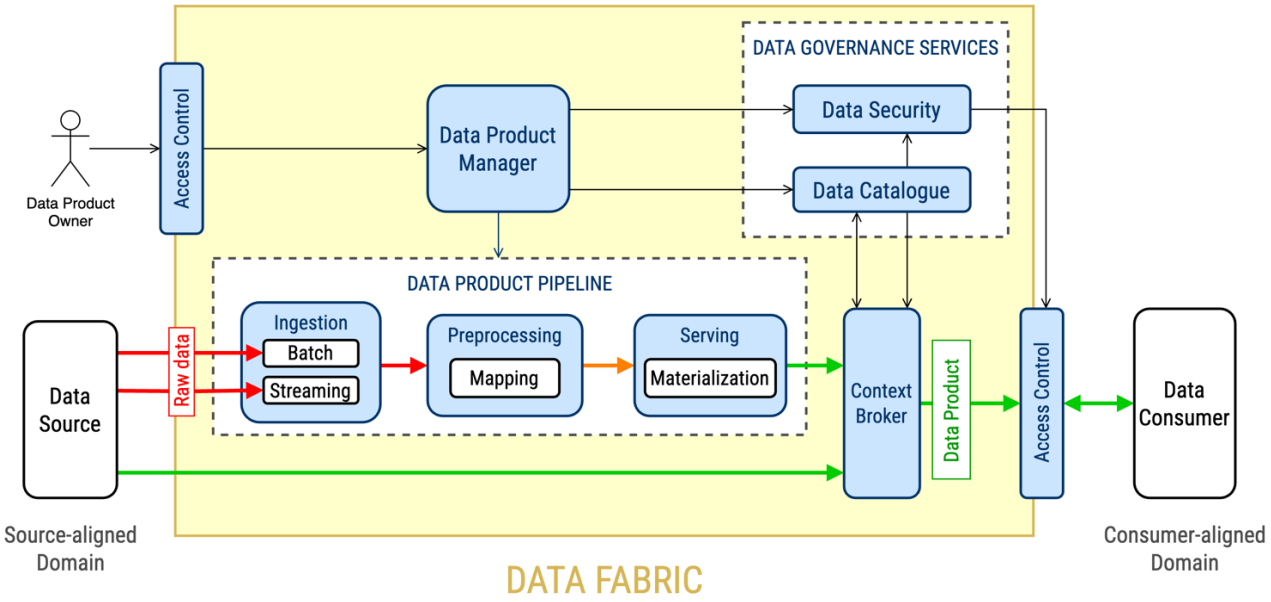


Figure 14. High level architecture of the aerOS Data Fabric.

The architecture of the aerOS Data Fabric is envisioned to run at an aerOS domain basis, meaning, in the practice one instance of each building block will be deployed per aerOS domain, except for the Data Product Pipeline components, which might run multiple instances in different IEs to improve scalability.

This atomic approach relies on NGSI-LD capabilities for distributing requests mong Context Brokers to realize a federated Data Fabric across the continuum.

3.2.1. Context Broker

The NGSI-LD Context Broker is the component that stores the knowledge graph of the aerOS Data Fabric. Orion-LD[5], a core product of the FIWARE's stack based on the NGSI-LD standard, has been selected as the open-source implementation of the Context Broker. Additionally, depending on the setup of the use case, Orion-LD can be combined with the Mintaka component to provide the temporal NGSI-LD API of the Context Broker.

This was already presented in D4.2, and is further described in the relation with the federation in Section 3.6.

3.2.2. Data Product Pipeline

Semantically annotated datasets that follow an ontology, complying with the NGSI-LD structure, can be directly sent to the Context Broker. However, in most of the use cases, the Data Fabric will integrate "raw" datasets as new data products in the knowledge graph. For these kinds of datasets, the Data Fabric includes the Data Product Pipeline: a framework based on generic, open-source tools based on standards from the Semantic Web, to facilitate the creation of data products.

Depending on the source, data products can be **batch** type (such as files or SQL databases) or **streaming** type (data coming from Kafka or MQTT).

[5] https://github.com/FIWARE/context.Orion-LD

The Data Product Pipeline, as illustrated in Figure 14 and previously presented in deliverables D4.1 and D4.2, comprises three main stages: **Ingestion**, **Preprocessing** and **Serving**. These stages have been implemented by combining new developed components and existing open-source projects.

In this sense, Figure 15 depicts a low-level architecture of the Data Product Pipeline, indicating which technologies have been used for components involved in each stage. In this final release of the aerOS Data Fabric, the integration of the Semantic Tools (Semantic Annotator and Semantic Translator) has been completed, and the detailed description of these components has been introduced in Sections 3.1.1 and 3.1.2, respectively. The actual presence of the Semantic Translator in the Data Product Pipeline is optional as it depends on the particularities of the data product, and therefore it is defined during the data product onboarding/creation process. In the following subsections, Morph-KGC and the RDF to NGSI-LD Translator are described in detail.

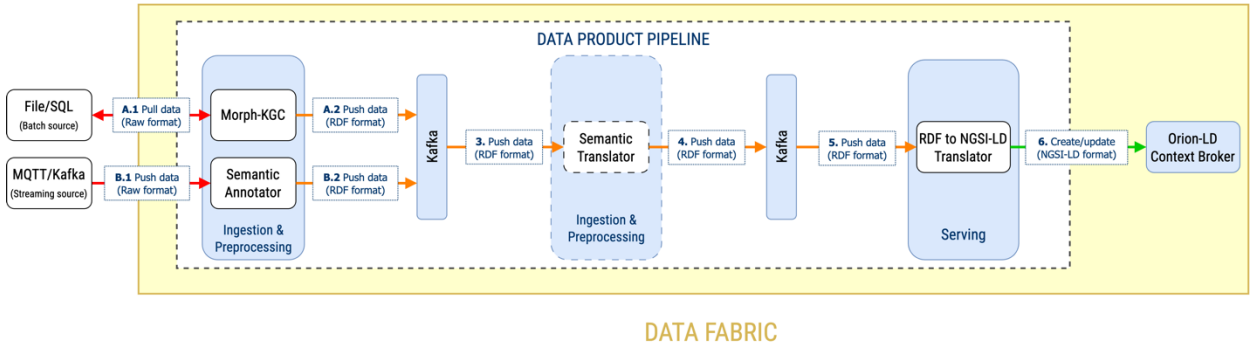


Figure 15. Low-level architecture of the Data Product Pipeline.

3.2.2.1. Ingestion and Preprocessing

3.2.2.1.1. Morph-KGC

Morph-KGC [D1] is an open-source project that implements an engine designed for constructing Resource Description Framework (RDF) knowledge graphs from heterogeneous data sources. It supports ingestion of raw datasets from batch data sources, such as remote files in JSON format or relational databases like MySQL. Morph-KGC builds upon the RML language for declaring the mapping of raw datasets to an ontology or set of ontologies. Based on these mappings, the tool transforms the ingested raw datasets and produces RDF triples.

As part of the development of the Data Fabric, Morph-KGC has received significant open-source contributions. The application now boasts integration with Kafka, allowing for the materialization of the knowledge graph into a Kafka topic.

Furthermore, an open-source enhancement has been implemented to simplify the deployment and management of Morph-KGC using Docker. The application can now be built into a Docker image, allowing for flexibility with optional dependencies specified during the build process.

Finally, Helm Charts have been introduced to streamline the deployment of Morph-KGC in Kubernetes environments. These Helm Charts offer an efficient way to manage the application, and they incorporate the capability to schedule recurring tasks using Cron Jobs. This feature enables users to execute specific tasks at scheduled intervals, enhancing the automation and periodic execution of Morph-KGC processes.

3.2.2.2. Serving

3.2.2.2.1. RDF to NGSI-LD translator

This component implements a generic translator from RDF to NGSI-LD, as depicted in Figure 16. The work takes inspiration from rdflib6 plugins that store RDF data in backends like Neo4j[7]. In this sense, this project

[6] https://github.com/RDFLib/rdflib
[7] https://github.com/neo4j/neo4j

provides a rdflib plugin where an NGSI-LD Context Broker works as the storage backend for RDF data. Additionally, the translator supports the ingestion of streams of RDF data via Kafka.

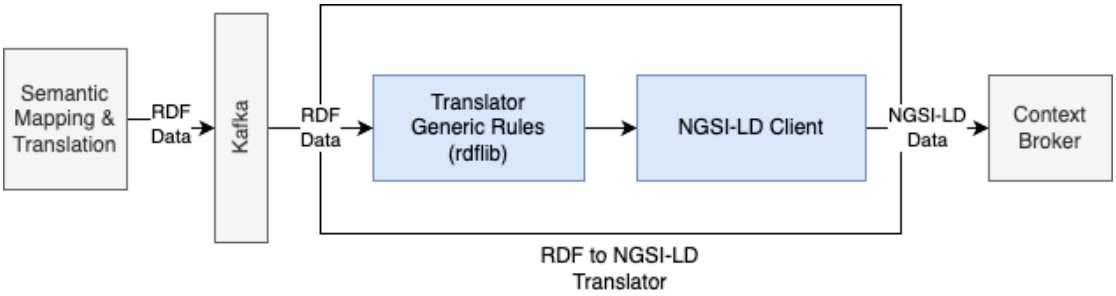


Figure 16. RDF to NGSI-LD Translator.

In its final release, the translator has evolved towards an implementation based on the rdflib library. This powerful library provides functions for parsing and serializing RDF. It does not support RDF-star yet, but the community is planning to extend the library to add support, since RDF-star is about to become a standard.

To transform the RDF data model (triples) into the NGSI-LD data model (property graph), the translator implements the following set of generic rules:

- **Subject**: Maps to an NGSI-LD Entity. The URI of the subject in the input RDF triple is the URI of the output NGSI-LD Entity. It should be noted that this approach does not follow the convention recommended by ETSI CIM, which goes `urn:ngsi-ld:<entity-type>:<identifier>.` The reason for doing this is to provide interoperability between RDF and NGSI-LD.

  Additionally, the translator has been improved with an additional feature required for handling subject mapping in certain scenarios:
  - **Blank node handling.** Blank nodes (or BNodes) are skolemized as described in Section 3.5 of RDF 1.1 Concepts and Abstract Syntax. By generating Skolem IRIs, blank nodes of an RDF graph can be transformed into NGSI-LD Entities and stored in the NGSI-LD Context Broker.

- **Predicate**:
  - **RDF classes**
  - The `a` or `rdf:type` predicates map to the NGSI-LD Entity Type. For example, the RDF triple `<http://example.org/people/Bob> a foaf:Person` translates into an NGSI-LD Entity of `foaf:Person` type, and URI http://example.org/people/Bob.
  - If no `rdf:type` predicate is found for the subject in the RDF graph, then the Entity Type will be set by default to the base RDF class: `http://www.w3.org/2000/01/rdf-schema#Class`. As described in Section 2.2 of RDF Schema 1.1, the `rdfs:Class` represent the class of RDF classes.
  - **RDF Datatype property** maps to an NGSI-LD Property. A special treatment is required when the literal of the predicate uses `xsd:datetime`. In this case the resulting NGSI-LD Property must follow the special format:

```
"myProperty": {
    "type": "Property", "value": {
        "@type": "DateTime",
        "@value": "2018-12-04T12:00:00Z"
    }
}
```

- **RDF Object property** maps to an NGSI-LD Relationship. The target of the Relationship is the URI of the object in the RDF triple.

- **Namespaces**: There is no need to create specific `@context` for translating to NGSI-LD. The resulting NGSI-LD Entity can just use expanded URIs. This approach is easier to maintain as it avoids maintaining `@context` files.

  Optionally, If the ingested RDF data includes a definition of namespaces with prefixes, then this information could be used to generate the `@context` for the translated NGSI-LD Entity. The resulting `@context` can be sent along the NGSI-LD payload or stored elsewhere and reference via Link header. The selected approach will depend on the use case and the developer's implementation.

In addition to the library that translates RDF into NGSI-LD based on the generic transformation rules, the component also includes and NGSI-LD client that sends the resulting creation or update of NGSI-LD Entities to the specified Context Broker.

3.2.3. Data Product Manager

The Data Product Manager component takes the form of a containerized REST API server with orchestration capabilities in the backend. The Data Product Manager has been developed in Python, leveraging the FastAPI[8] and Uvicorn[9] libraries.

The REST API provides methods for managing the life cycle of data products as depicted below:

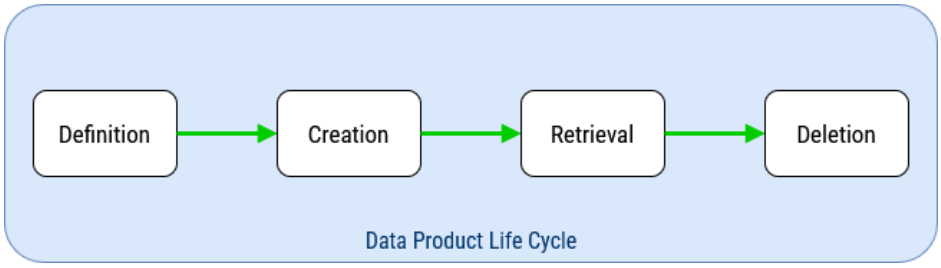


Figure 17. Data Product Life Cycle.

3.2.3.1. Data Product Creation

Allows the onboarding of new data products. Building upon the proposed definition of a data product, this method expects the following metadata and artifacts:

- **Data source configuration**. Indicates the type of data source along with connection details such as source URL (e.g., JDBC URL in relational databases) and access credentials (e.g., username/password, certificate). Additionally, the following information might be provided depending on the type of data source:
  - **Data source freshness**. Only supported for data sources of batch type. Determines how frequently Data Fabric collects raw data from the target data source.
- **Mapping**. Data mapping is mandatory, based on declarative mapping rules defined with RML [D2] or YARRRML [D3]). Artifact that describes the mappings to transform the ingested raw data into a graph structure and semantically annotate the data based on referenced ontologies.
- **Translation**. If ontology translations are needed, the onboarding process allows uploading artifacts (files) that will trigger the utilization of the Semantic Translator in the Data Product Pipeline. According to the description of this component in Section 3.1.2, these optional files (alignments) enable the translation:

[8] https://fastapi.tiangolo.com/
[9] https://www.uvicorn.org/

- From source ontology to central ontology.
- From central ontology to target ontology.

Both files can be provided at the same time, or only one of them. If no files are provided, no translation is required and, therefore, the Semantic Translator will not be used.

- **Data Governance**. Metadata containing identifiers to entities required for governing the new data product from the Data Catalogue. These identifiers are used by the respective entities in the knowledge graph:
  - Data product owner.
  - Business glossary terms.
  - Tags/keywords.

The Data Product Manager supports onboarding data products from batch data sources (relational databases and files), as well as from streaming data sources (Kafka and MQTT). Figure 18 to Figure 21 include snapshots of the documentation of the data product onboarding REST API method for all data sources.

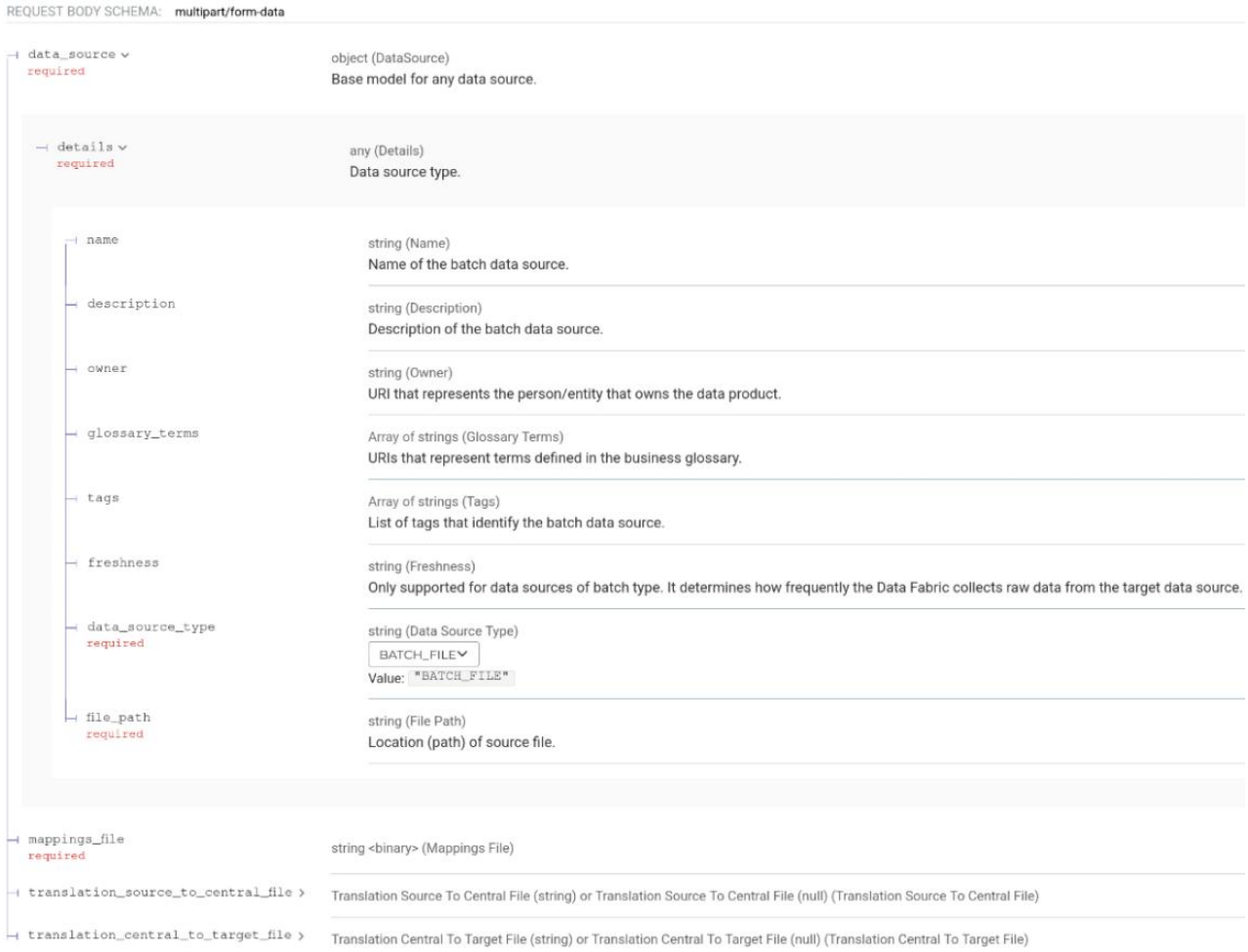


Figure 18. Data product onboarding REST API - Batch type - Files.

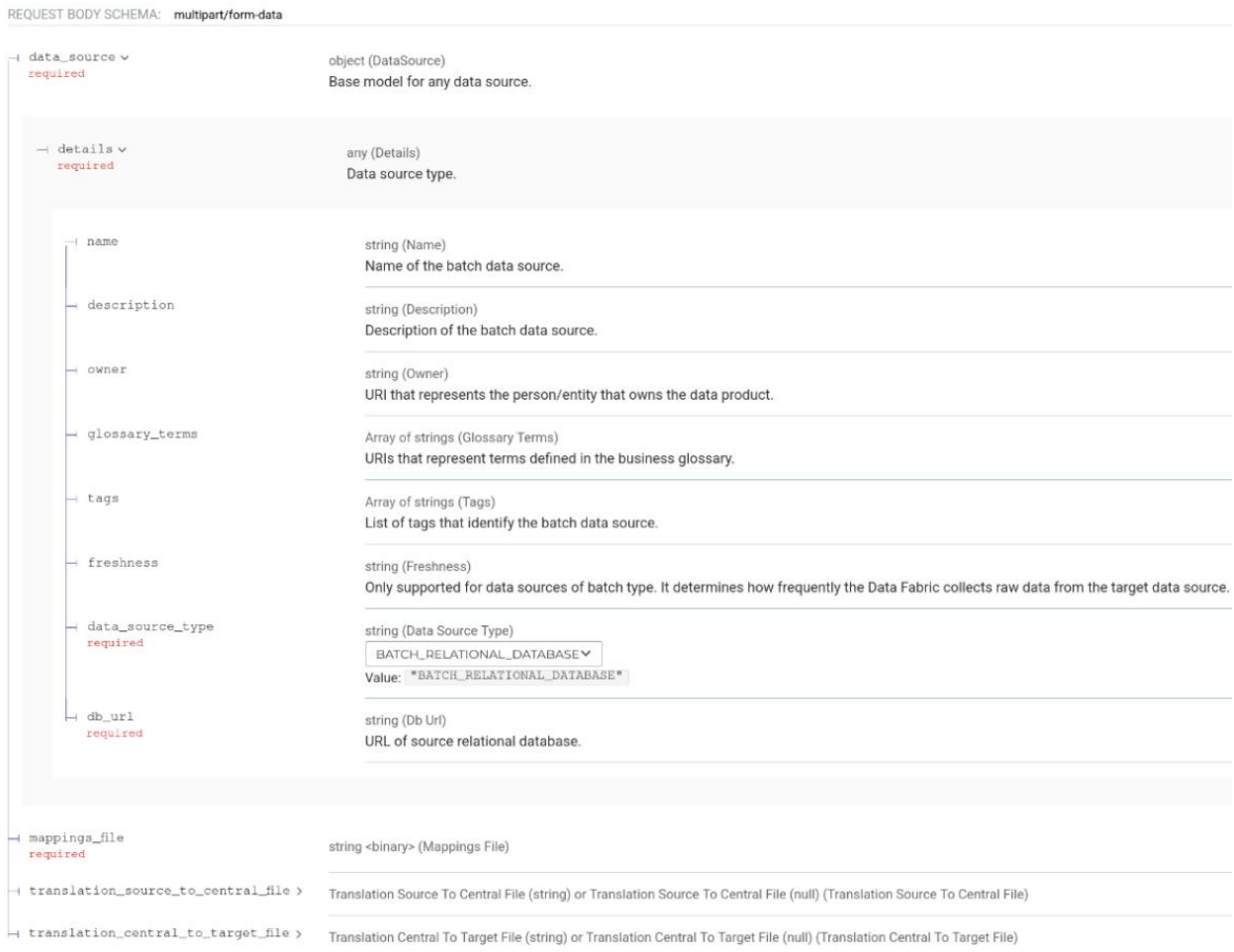


Figure 19. Data product onboarding REST API - Batch type - Relational databases.

REQUEST BODY SCHEMA: multipart/form-data

data_source ⌄ required	object (DataSource) Base model for any data source.
details ⌄ required	any (Details) Data source type.
name	string (Name) Name of the streaming data source.
description	string (Description) Description of the streaming data source.
owner	string (Owner) URI that represents the person/entity that owns the data product.
glossary_terms	Array of strings (Glossary Terms) URIs that represent terms defined in the business glossary.
tags	Array of strings (Tags) List of tags that identify the streaming data source.
input_format › required	"JSON" (string) or "XML" (string) or "CSV" (string) (Input Format) Input data format of the streaming data source.
input_topic required	string (Input Topic) Name of the input topic.
data_source_type required	string (Data Source Type) STREAMING_KAFKA Value: "STREAMING_KAFKA"
host required	string (Host) Hostname, FQDN or IP address where the Kafka broker is reachable.
port required	integer (Port) Port number where the Kafka broker is reachable.
group_id	string (Group Id) Kafka group ID.
mappings_file required	string <binary> (Mappings File)
translation_source_to_central_file ›	Translation Source To Central File (string) or Translation Source To Central File (null) (Translation Source To Central File)
translation_central_to_target_file ›	Translation Central To Target File (string) or Translation Central To Target File (null) (Translation Central To Target File)

Figure 20. Data product onboarding REST API - Streaming type - Kafka sources.

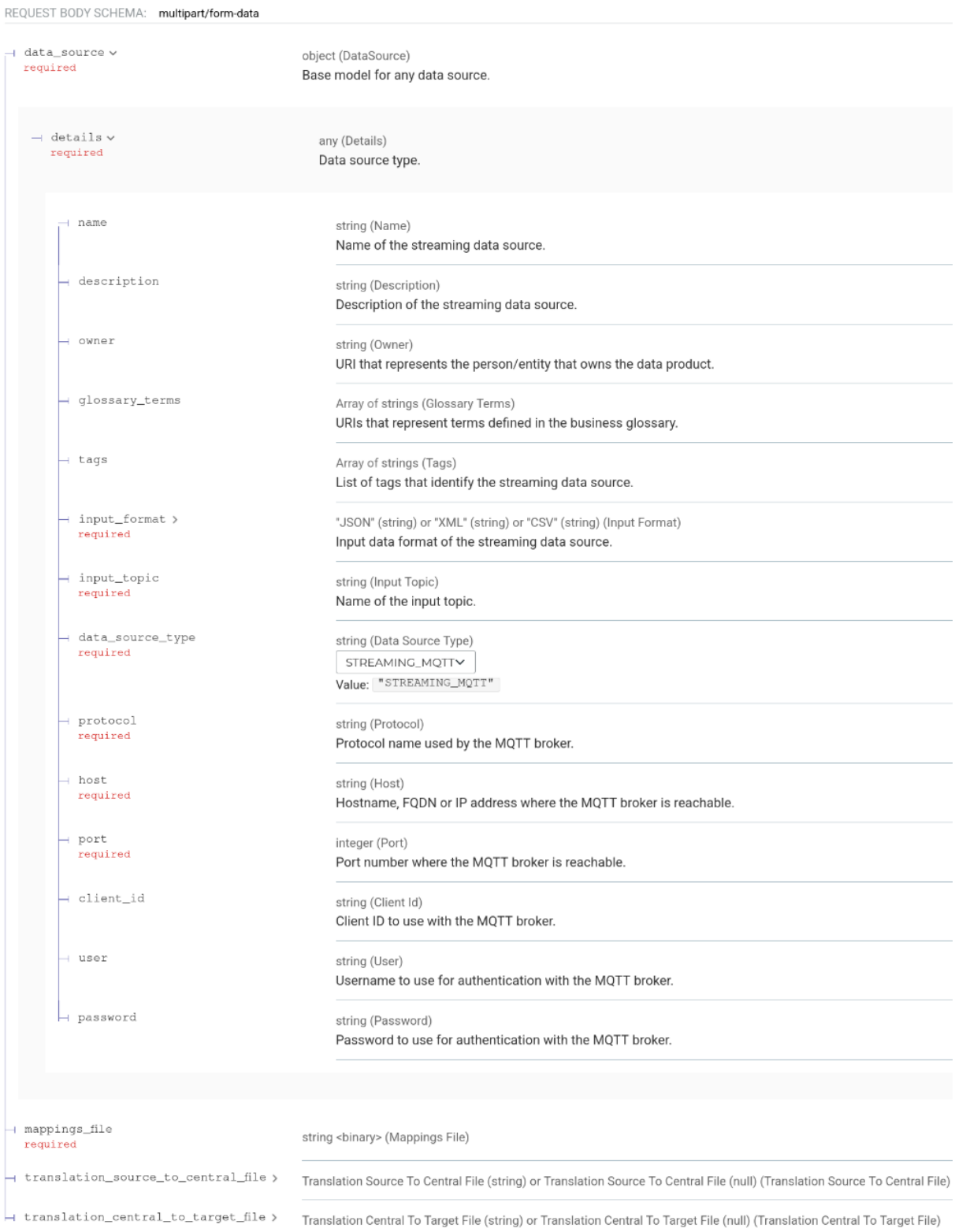


Figure 21. Data product onboarding REST API - Streaming type - MQTT sources.

To manage the life cycle of data products, this component uses a local MongoDB[10] collection for storing data product details/metadata.

[10] https://www.mongodb.com/

Upon creation of a new data product, the Data Product Manager returns a JSON object containing all its details, including its identifier, that enables its retrieval and deletion at any time. Figure 22 contains an example of this response.


```
{
  "message": "Data product onboarded successfully.",
  "data_product": {
    "_id": "0d385e70-4035-43ec-b779-6ef853c6cf00",
    "name": "ldap",
    "description": "LDAP Test Data Product",
    "owner": "continuumadministrator2",
    "glossary_terms": [
      "example_term_1",
      "example_term_2"
    ],
    "tags": [
      "ldap",
      "batch",
      "example"
    ],
    "data_source_type": "BATCH_FILE",
    "translation": {
      "defined": "no"
    },
    "details": {
      "output_kafka_topic": "knowledge-graphs",
      "file_path": "http://data-fabric-ldap-collector.default.svc.cluster.local:63300/ldap.json",
      "freshness": {
        "enabled": "True",
        "schedule": "0 0 * * *"
      }
    },
    "creationTimestamp": "2025-01-15T11:29:04Z"
  }
}
```


Figure 22. Data product onboarding REST API - Successful JSON response.

Taking the information provided by data product owners through the REST API, the Data Product Manager deploys and configures a new data pipeline for the onboarded data product. In this regard, all components of the Data Product Pipeline are containerized and deployed as Helm Charts on Kubernetes. To orchestrate the construction of the pipeline, the Data Product Manager leverages the Helm Controller component from the FluxCD project[11]. The Helm Controller implements a Kubernetes operator that defines Helm charts and Helm releases as new custom resources in Kubernetes. This allows to manage the life cycle of Helm releases through the K8s API.

This release also provides support for data governance features in coordination with the Data Catalog Service, described in Section 3.2.4. Regarding access control policies, these are applied and enforced by the Data Security Service, described in Section 3.2.5, along with the rest of aerOS security components.

Table 3 includes the definition of the associated REST API method.

Table 3. Data Product Manager – REST API method definition – onboard data product.

Endpoint	/dataProducts
HTTP Method	POST
Header	**Value**
accept	application/json
Content-Type	multipart/form-data

[11] https://fluxcd.io/

Description	Onboards a new data product.

3.2.3.2. Data Product Retrieval

Allows to get the information of the data products currently registered in the data catalogue. The retrieval can be performed for all data products (a JSON list with details of all of them is returned), or for a specific one, providing its identifier. Table 4 and Table 5 include the definitions of the associated REST API methods.

Table 4. Data Product Manager - REST API method definition - retrieve all data products.

Endpoint	/dataProducts
HTTP Method	GET
Header	Value
accept	application/json
Description	Returns information of all current data products.

Table 5. Data Product Manager - REST API method definition - retrieve specific data product.

Endpoint	/dataProducts/{data_product_id}
HTTP Method	GET
Header	Value
accept	application/json
Parameter	Description
data_product_id	Data Product identifier
Description	Returns information of the data product whose identifier is passed as parameter.

3.2.3.3. Data Product Deletion

Allows deleting current, active data products. As with retrieval, deletion can be done for all data products or for a specific one, providing its identifier. Table 6 and Table 7 include the definitions of the associated REST API methods.

Table 6. Data Product Manager - REST API method definition - delete all data products.

Endpoint	/dataProducts
HTTP Method	DELETE
Description	Deletes all current data products.

Table 7. Data Product Manager - REST API method definition - delete specific data product.

Endpoint	/dataProducts/{data_product_id}
HTTP Method	DELETE
Parameter	Description
data_product_id	Data Product identifier
Description	Deletes the data product whose identifier is passed as parameter.

3.2.3.4. Technologies and standards

Table 8. Data Product Pipeline technologies.

Technology/ Standard	Description	Component
FastAPI	Web framework for building APIs with Python	Data Product Manager
JSON	JavaScript Object Notation	Morph-KGC
Kafka	Open-source distributed event streaming platform	Morph-KGC, RDF to NGSI-LD Translator
MongoDB	Open-source, document-oriented, NoSQL database system implementation	Data Product Manager
NGSI-LD	Next Generation Service Interface with Linked Data	RDF to NGSI-LD Translator
OpenAPI	Formal standard for describing HTTP APIs	Data Product Manager
RDF	Resource Description Framework	Morph-KGC, RDF to NGSI-LD Translator
RML	RDF Mapping Language	Morph-KGC
SQL	Structured Query Language	Morph-KGC
Uvicorn	Web server implementation for Python	Data Product Manager
YARRRML	YAML subset for representing human-readable RML mapping rules	Morph-KGC

3.2.4. Data catalogue

3.2.4.1. LDAP Collector

The intermediate release of the aerOS Data Catalogue, described in deliverable D4.2, brought a new component called LDAP (*Lightweight Directory Access Protocol*) Connector. As depicted in Figure 23, the main idea that lies behind this component is allowing the collection of data from LDAP-compliant databases, such as OpenLDAP, and their inclusion into the knowledge graph as NGSI-LD data that can persist in the Orion-LD Context Broker.

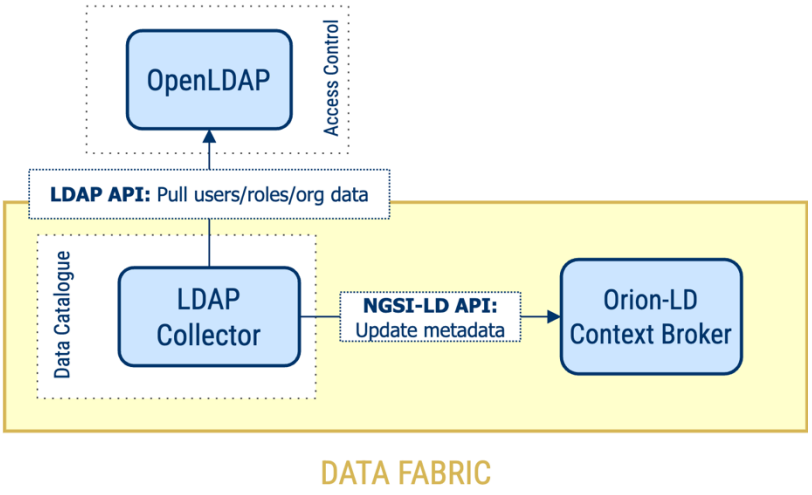


Figure 23. Data catalog connector for LDAP.

In this final release, this component has been improved, changing its name for LDAP Collector. It leverages the architecture and framework of the Data Fabric and the Data Product Pipeline by enabling the inclusion of LDAP data as a batch data product, as depicted in Figure 24. In this case, the freshness property of this kind of data allows for periodic synchronization and ensures that mapped directory data is up to date in the knowledge graph. This is important because access control policies ultimately rely on directory data, particularly users and their roles (that define permissions).

The LDAP Collector exposes a REST API that allows retrieving data (users, groups, roles and organizations) as a JSON file/dictionary object. Upon calling the collector, it connects to the configured server and fetches the data, transforming and aggregating them into a JSON dictionary that is returned in response.

According to the framework and architecture provided by the Data Product Pipeline, the JSON file is periodically requested by Morph-KGC which, using the appropriate mappings, generates RDF triples and writes them to Kafka. These triples are then read by the RDF to NGSI-LD translator, generating NGSI-LD data that are uploaded to the Orion-LD Context Broker, that ultimately ensures their persistence. A sequence diagram that represents the working principle of the LDAP Collector and the interactions between the LDAP server, the LDAP Collector and Morph-KGC is also depicted in Figure 25.

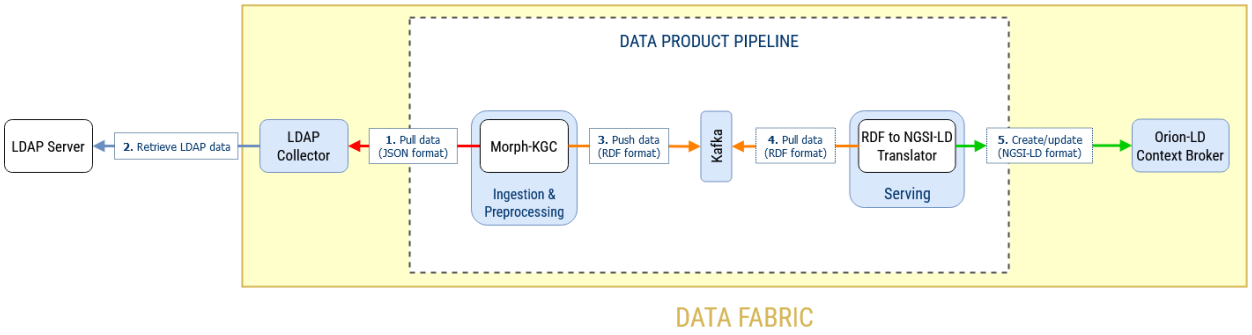


Figure 24. Data Product Pipeline for LDAP.

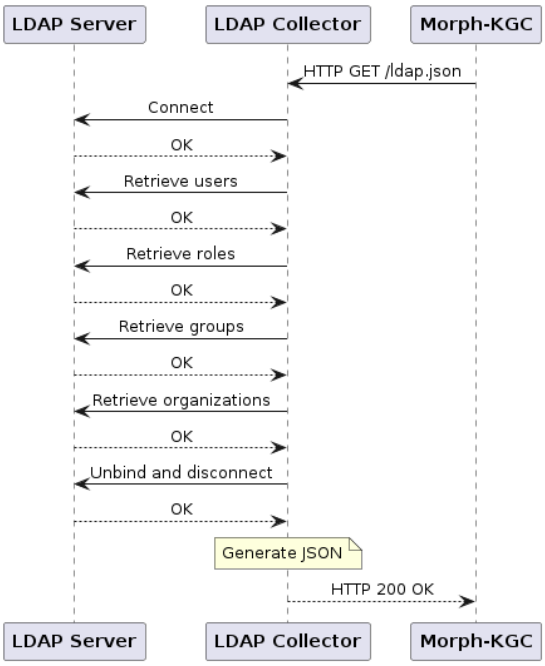


Figure 25. Sequence diagram - Data Product Pipeline for LDAP.

The LDAP Collector has been implemented as a containerized Python application that uses the FastAPI, Uvicorn and ldap3[12] libraries, and is compatible with any LDAP database/server implementation.

The mappings of LDAP data to concepts of standard ontologies (FOAF [D4] and ORG [D5]) and the aerOS continuum ontology have also been completed in this release. These mappings are grouped into the following entities: **User**, **Role**, **Organization** and **Membership**. **User** and **Role** entities match to the homonymous classes present in the LDAP database. The **Organization** entity holds a particular meaning in the use case of aerOS: since there will always be one single organization, it matches for groups defined in the database, rather than the homonymous class in LDAP. The **Membership** entity acts as a linking concept between users, the roles they are assigned and the groups they belong to. The YARRRML[13] (subset of YAML for representing RML mapping rules in a human-readable format) mapping definitions are included in Figure 26 to Figure 29. These include sources and how subjects and properties-objects are generated.

```
users:
  sources:
    - ['http://ldap-collector:63300/ldap.json~jsonpath', '$.users[*]']
  s: urn:User:$(attributes.uid)
  po:
    - [a, aeros:User]
    - [foaf:firstName, $(attributes.givenName), xsd:string]
    - [foaf:lastName, $(attributes.sn), xsd:string]
    - [schema:email, $(attributes.mail), xsd:string]
    - [aeros:username, $(attributes.uid), xsd:string]
```

Figure 26. Mappings for LDAP users.

```
roles:
  sources:
    - ['http://ldap-collector:63300/ldap.json~jsonpath', '$.roles[*]']
  s: urn:Role:$(attributes.cn)
  po:
    - [a, org:Role]
```

Figure 27. Mappings for LDAP roles.

```
# Specific use case: LDAP groups are considered organizations.
organizations:
  sources:
    - ['http://ldap-collector:63300/ldap.json~jsonpath', '$.groups[*]']
  s: urn:Organization:$(attributes.cn)
  po:
    - [a, org:Organization]
    - [dct:identifier, $(attributes.cn), xsd:string]
```

Figure 28. Mappings for LDAP groups (organizations).

[12] https://ldap3.readthedocs.io/en/latest/
[13] https://rml.io/yarrrml/

```
membership:
  sources:
    - ['http://ldap-collector:63300/ldap.json~jsonpath', '$.memberships[*]']
  s: urn:Membership:User:$(memberUid):Role:$(role_cn)
  po:
    - [a, org:Membership]
    - p: org:member
      o:
        - mapping: users
          condition:
            function: equal
            parameters:
              - [str1, $(memberUid), s]
              - [str2, $(attributes.uid), o]
    - p: org:role
      o:
        - mapping: roles
          condition:
            function: equal
            parameters:
              - [str1, $(role_cn), s]
              - [str2, $(attributes.cn), o]
    - p: org:organization
      o:
        - mapping: organizations
          condition:
            function: equal
            parameters:
              - [str1, $(group_cn), s]
              - [str2, $(attributes.cn), o]
```

Figure 29. Mappings for memberships (link between users, roles and organizations/groups).

This final release of the aerOS Data Fabric has introduced the Data Catalog Service as a new component of the Data Catalog building block. The Data Catalog Service provides a REST API for registering new data products that have been made available in the Data Fabric by means of the Data Product Pipeline.

This service can be used only by aerOS users with the Data Product Owner to register their data products in the Data Catalog of the Data Fabric instance. To this end, the access control setup based on KrakenD and Keycloak is leveraged to enforce authorization for this kind of role.

For the registration of a data product, the owner must provide the following metadata to the API:

- Name of the data product
- Description of the data product
- Owner (aerOS username) of the data product
- Free-text keywords that identify the data product
- Glossary terms formally defined in a business glossary that identify the data product

Based on this input, the Data Catalog Service processes these metadata and registers the data product in the Data Catalog, as depicted in Figure 30. In this process, the Data Catalog Service transforms the metadata into NGSI-LD format based on the aerOS Data Catalog Ontology defined in Section 3.1.3.3, and stores these metadata in the Orion-LD Context Broker of the Data Fabric instance. The result is a Data Catalog that is to the Data Fabric instance. As noted previously, external applications like the Management Portal can construct a global aerOS Data Catalog by federating the local Data Catalog metadata among Context Brokers.

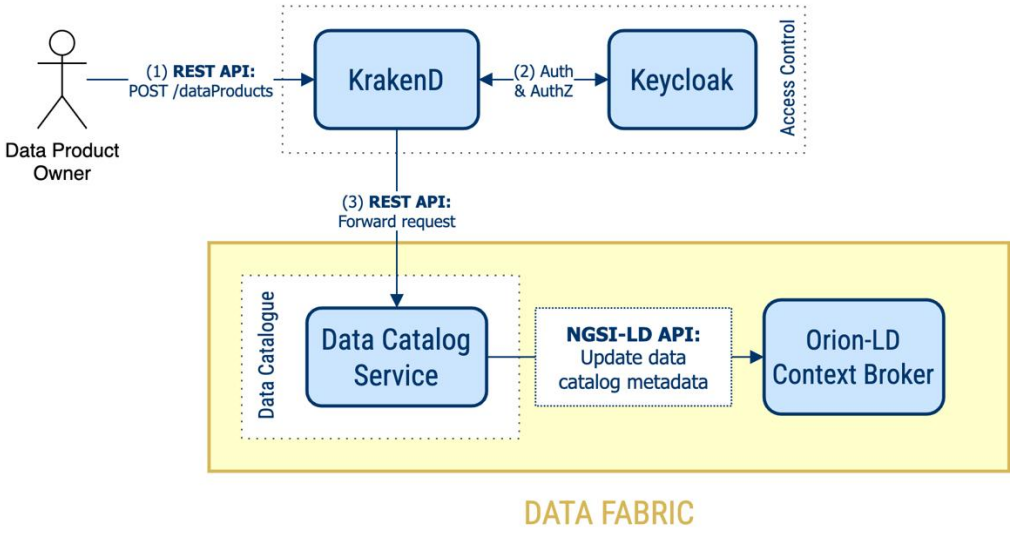


Figure 30. Architecture of Data Catalog Service.

Following a similar approach as in the Data Product Manager, this service has also been implemented as a containerized microservice based on the FastAPI framework in Python. The implemented REST API can be accessed by sending a `POST` request to the `/dataProducts` endpoint. Such API is documented in more detail in Figure 31 based on its OpenAPI specification.

Register Data Product

Registration of a data product in the data catalog.

REQUEST BODY SCHEMA: application/json

name required	string (Name) Name of the data product. This will uniquely identify the data product in the data catalog.
description required	string (Description) Description of the data product.
owner required	string (Owner) aerOS username that identifies the owner of the data product.
keywords >	Array of Keywords (strings) or Keywords (null) (Keywords) (Optional) List of custom keywords/tags that identify the data product.
glossary_terms required	Array of strings (Glossary Terms) List of URIs identiftying concepts associated with the data product. These concepts must be captured in existing ontologies.

Figure 31. Documentation of Register Data Product API.

3.2.4.2. Technologies and standards

Table 9. Data Catalog technologies.

Technology/ Standard	Description	Component
FastAPI	Web framework for building APIs with Python	LDAP Collector, Data Catalog Service

JSON	JavaScript Object Notation	LDAP Collector, Morph-KGC
Kafka	Open-source distributed event streaming platform	Morph-KGC
LDAP	Lightweight Directory Access Protocol	LDAP Collector
NGSI-LD	Next Generation Service Interface with Linked Data	Data Catalog Service, LDAP Collector
OpenAPI	Formal standard for describing HTTP APIs	LDAP Collector, Data Catalog Service
RDF	Resource Description Framework	Morph-KGC, Data Catalog Service
RML	RDF Mapping Language	Morph-KGC
Uvicorn	Web server implementation for Python	LDAP Collector, Data Catalog Service
YARRRML	YAML subset for representing human-readable RML mapping rules	Morph-KGC

3.2.5. Data security

The security model in the Data Fabric has undergone significant enhancements, incorporating robust authentication and authorization mechanisms across multiple components through integrations with Keycloak[14], KrakenD[15] (T3.4), and new technologies such as Apache APISIX[16] and Open Policy Agent (OPA)[17].

Apache APISIX serves as a high-performance, cloud-native API gateway that manages API traffic. It offers features such as load balancing or dynamic upstream routing. By integrating Apache APISIX, the Data Fabric centralizes and streamlines API traffic management, ensuring secure and efficient communication between components.

OPA is an open-source, general-purpose policy engine that decouples policy decision-making from policy enforcement. Based on the "Policy as Code" paradigm, OPA uses the Rego[18] policy language to define and evaluate fine-grained policies. This allows the Data Fabric to enforce detailed access controls at the data level. For instance, access for a user with the "maintainer" username can be restricted to querying entities belonging to "Luxury Stores," as shown in the following policy example:

```
default allow = false

is_maintainer if input.claims.preferred_username == "maintainer"
is_get if input.request.method == "GET"
is_entities if input.request.path == "/ngsi-ld/v1/entities"
is_Store if input.request.query.q == "storeName==\"Luxury Store\""

allow if {
        is_get
        is_entities
        claims.preferred_username == "maintainer"
        is_Store
}
```

Figure 32. Policy written in Rego.

[14] https://www.keycloak.org/
[15] https://www.krakend.io/
[16] https://apisix.apache.org/
[17] https://www.openpolicyagent.org/
[18] https://www.openpolicyagent.org/docs/latest/policy-language/

In addition, the integrity and authenticity of the incoming access tokens is verified by checking their signatures against the public key of our Keycloak. This ensures that the tokens come from a trusted source and have not been tampered with, as shown in the image below:

```
claims := payload if {
        # Verify the signature on the Bearer token. In this example the
        # hardcoded into the policy however it could also be loaded via
        # an environment variable. Environment variables can be accessed
        # the `opa.runtime()` built-in function.
        io.jwt.verify_rs256(bearer_token, "-----BEGIN PUBLIC KEY-----\nM

        # This statement invokes the built-in function `io.jwt.decode` p
        # parsed bearer_token as a parameter. The `io.jwt.decode` functi
        # array:
        #
        #       [header, payload, signature]
        #
        # In Rego, you can pattern match values using the `=` and `:=` o
        # example pattern matches on the result to obtain the JWT payloa
        [_, payload, _] := io.jwt.decode(bearer_token)
}
```

Figure 33. Analysing the signature of the access token in OPA.

The Orion-LD Context Broker now incorporates Apache APISIX as the API gateway and OPA. This integration ensures that:

- Only authorized requests reach the Context Broker.
- Dynamic, fine-grained access control policies are enforced.
- Policies are centrally managed, simplifying updates and maintenance.

The workflow for data consumption is depicted in Figure 34, highlighting the authorization process for data consumers of the Context Broker.

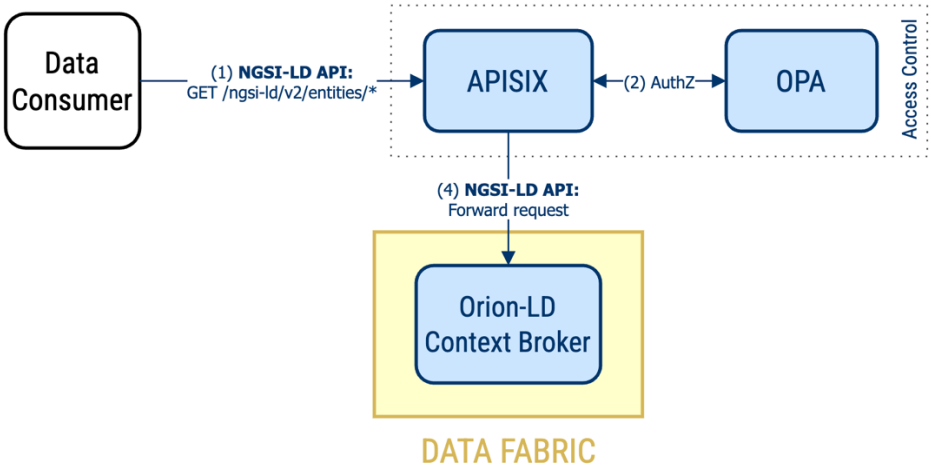


Figure 34. Authorization workflow for data consumers of the Context Broker.

The Data Product Manager continues to enforce role-based access control (RBAC) for onboarding new data products. As in previous implementations, only users with the "admin" role are authorized to make POST requests to the `/onboard` endpoint of the Data Product Manager's REST API. This ensures that only designated individuals can introduce new data products into the system. The authorization workflow for this process is illustrated in Figure 35.

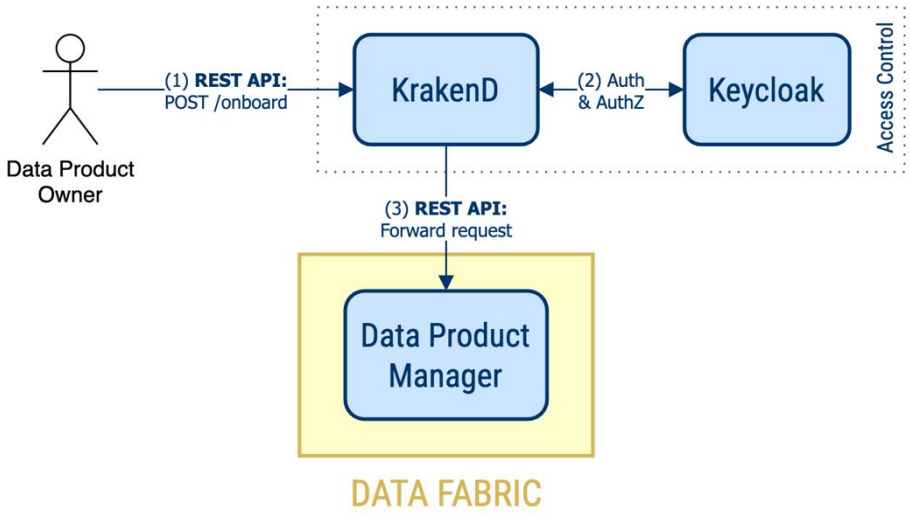


Figure 35. Authorization workflow for data product owners in the Data Product Manager.

3.2.5.1. Data Security Service

To further enhance policy management and enforcement, the Data Security Service has been introduced. The service empowers data product owners to define custom access policies tailored to their specific needs. This FastAPI-based service interfaces with the OPA client, providing comprehensive policy management capabilities, including:

- **Register Policies**: Allows users to upload `.rego` files via a POST request.
- **Retrieve and Download Policies**: Provides the Rego content of a specific policy, with an option to download it as a `.rego` file.
- **List Policies**: Displays all registered policies.
- **Update Policies**: Enables users to modify existing policies without deleting them.
- **Delete Policies**: Supports the removal of policies via a DELETE request.

The Data Security Service offers the following endpoints for policy management:

- **List policies**: GET /policies
- **Get policy content**: GET /policies/{policy_name}/?as_file={true|false}
- **Register policy**: POST /policies/{policy_name}
- **Update policy**: PUT /policies/{policy_name}
- **Delete policy**: DELETE /{policy_name}

This service allows for fine-grained control over data access policies, enabling data product owners to define custom access rules based on various criteria such as specific aerOS users, roles, or organizational groups. This functionality enhances the data product creation process by allowing the establishment of a default policy when a new data product is created. Subsequently, data product owners can refine and adjust these policies over time, all through this centralized service.

With these advancements, the Data Fabric achieves a robust, flexible, and scalable security model, ensuring compliance with organizational and regulatory requirements while empowering users with granular control over data access.

3.2.5.2. Technologies and standards

Table 10. Data Security technologies.

Technology/ Standard	Description	Component
Apache APISIX	Open-source API Gateway	Access Control
Open Policy Agent (OPA)	Policy-based control for cloud native environments	Access Control, Data Security Service
Keycloak	An open-source identity and access management solution	Access Control
KrakenD	Open-source implementation of an API GW	Performs the Primary Entry Point (PEP) role in the access control for the Data Product Manager
Rego	A high-level declarative language used to express policies	Access Control, Data Security Service
OAuth 2.0	An authorization framework that enables applications to obtain limited access to user accounts	Access Control
FastAPI	Web framework for building APIs with Python	Data Security Service
OpenAPI	Formal standard for describing HTTP APIs	Data Security Service
Uvicorn	Web server implementation for Python	Data Security Service

3.3. Decentralized frugal AI

aerOS Meta-OS must allow the execution of distributed AI tasks over the continuum considering also limited resource availability that is possible close to the edge. Decentralized AI mechanisms are to be supported for internal AI (internal aerOS mechanisms) as well as for external AI (the deployment of specific AI services for stakeholders over the continuum). Moreover, frugality and explainability of AI-based functionalities have been also investigated as auxiliary but relevant elements of aerOS. Supplementary research

Decentralized AI service deployment with network representation is included in Appendix A.

3.3.1. Decentralized AI workflows

AI workflows are a combination of workloads, put together in a way to achieve a specific functionality of AI. In that sense, decentralized AI workflows in aerOS can be described as a concept of AI tasks that can be divided into sub-tasks and delegated for execution to selected IEs. Hence, aerOS decentralized AI workflow covers the use case of federated training of ML models (Federated Learning, FL) using continuum resources.

The aerOS AI tasks/workflows will be deployed as a set of interacting services, namely AI Task Controller and AI Local Executors. One the one hand, **AI Task Controller** is a set of logically related service components that control the execution of a decentralized AI workflow in terms of synchronization of partial results. AI workflow can be decomposed into workloads (or sub-tasks) that e.g., produce results that need to be aggregated or prepare data to be consumed by the model. On the other hand, **AI Local Executor** is an aerOS service that executes workloads (or AI sub-tasks) i.e., to execute a granular “step in the workflow”.

The service components of the AI task controller and AI local executors are illustrated and described below:

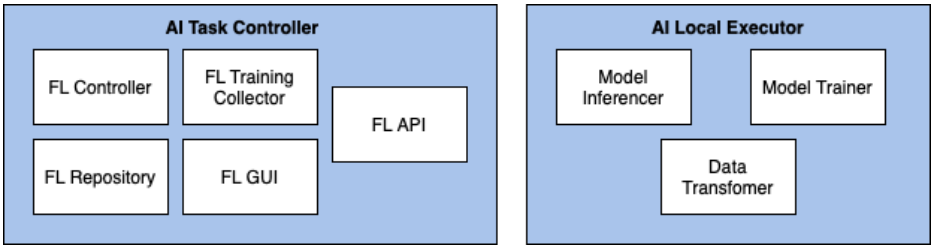


Figure 36. Internal components of AI Task Controller and AI Local Executor services.

Table 11. Components of AI Task Controller.

Component	Description
FL Controller	Responsible for accepting a task description, initializing execution of workload using deployed services, managing, and monitoring AI task lifecycle by means of a simultaneous two-way communication channel with every AI Local Executor through TCP connection for monitoring how many of them are available and running.
FL Training Collector	The FL training process involves several independent parties that collaborate to provide an enhanced ML model. In this process, the different local updates suggestions should be aggregated. This is tackled by the FL Training Collector, which is also in charge of sending the results of the training along with the updated model weights to FL Repository for storage.
FL Repository	One of the key application aspects of FL is making it persistent and configurable. The FL repository stores (and delivers upon request/need) the aggregation algorithms, ML algorithms, and the resulted ML models from FL training.
FL GUI	Provides a Graphical UI that allow to set up and track an AI Task workflow, including the initial configuration (e.g., minimum number of FL Local Executors, number of FL training rounds, the initial shared AI algorithm, encryption mechanism, aggregation strategy, or the evaluation metric).
FL API	Offers a REST API to allow communication and interaction between the FL GUI and FL Controller.

Table 12. Components of AI Local Execution.

Component	Description
Model Inferencer	Designed for fast and lightweight communication, Local Model Inferencer provides predictions of a selected model.
Model Trainer	Offers the functionalities of an FL client, including the local training and evaluation of models from two popular ML libraries.
Data Transformer	Supports data preprocessing, data loading and training methods using serializable modules.

Figure 37 presents a sequence diagram of federated training task execution using aux AI services. The execution is initiated by the AI Task Controller's that receives and distributes task configuration to other services from the GUI. After that, both the FL Training Collector service component from AI Task Controller, and AI Local Executors set up a communication channel and, in rounds, perform training and aggregation of results. The final validated model (according to predefined performance criterion) is stored in the FL Repository of the AI Task Controller, which is also accessible through the Task Controller GUI.

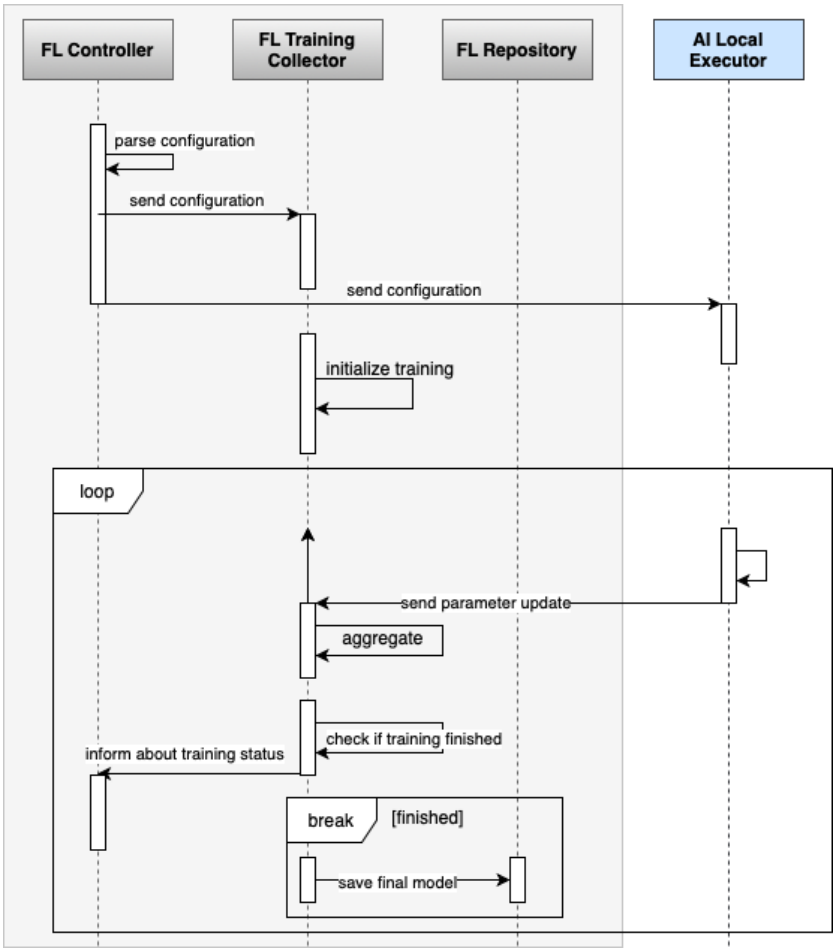


Figure 37. Workflow of FL training execution.

3.3.1.1. Technologies and standards

The deployment process of aerOS decentralized AI workflows is split into two parts, as shown in Figure 38. Whereas standalone AI Task Controllers must be deployed on aerOS continuum using helm charts, AI Local Executor services might potentially require some specific configurations depending on the tasks that will be run on them. Thus, AI Local Executor placement on an aerOS continuum is controlled by intention blueprints established in the aerOS Management Portal and orchestration mechanisms.

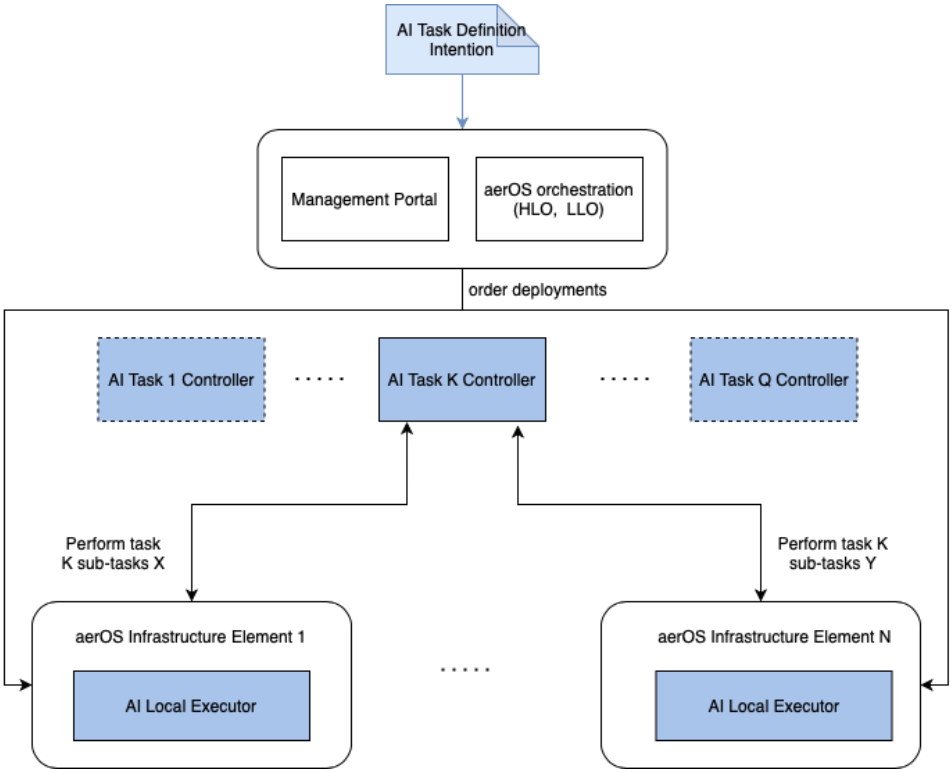


Figure 38. aerOS decentralized AI workflow deployment.

Table 13. AI workflows technologies.

Technology/ Standard	Description	Component
Python	Python is an interpreted high-level general-purpose programming language with a set of libraries. Very popular for data analysis and ML applications.	All
FastAPI	A web micro framework written in Python, known for being both robust and high performing.	All
Flower	A FL framework designed to work with many clients. It is both compatible with a variety of ML frameworks and supports a wide range of devices.	Model Trainer, FL Training Collector
FedML	Research library and benchmark for Federated ML containing federated algorithms and optimizers.	FL Training Collector
TensorFlow Lite	A free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.	Model Inferencer, Model Trainer
pyTorch	An open-source machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab (FAIR).	Model Trainer
MongoDB	MongoDB is a source-available cross-platform document-oriented database program.	FL Repository

VUE.JS	An industry standard with a comprehensive set of tools for creating web user interfaces, a d built with full Typescript support.	FL GUI

The main pages of the customized Graphical User Interface web application embedded within the AI Task Controller are pictured below:

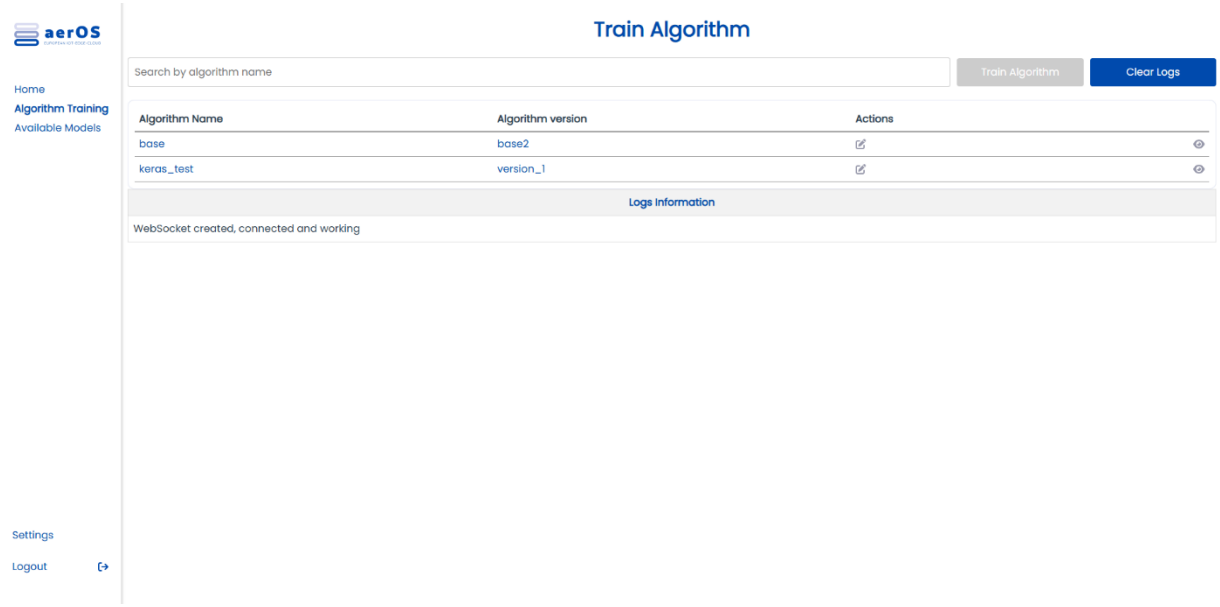


Figure 39. FL GUI home page in which a new decentralized AI workflow can be set up on top of the AI Task Controller.

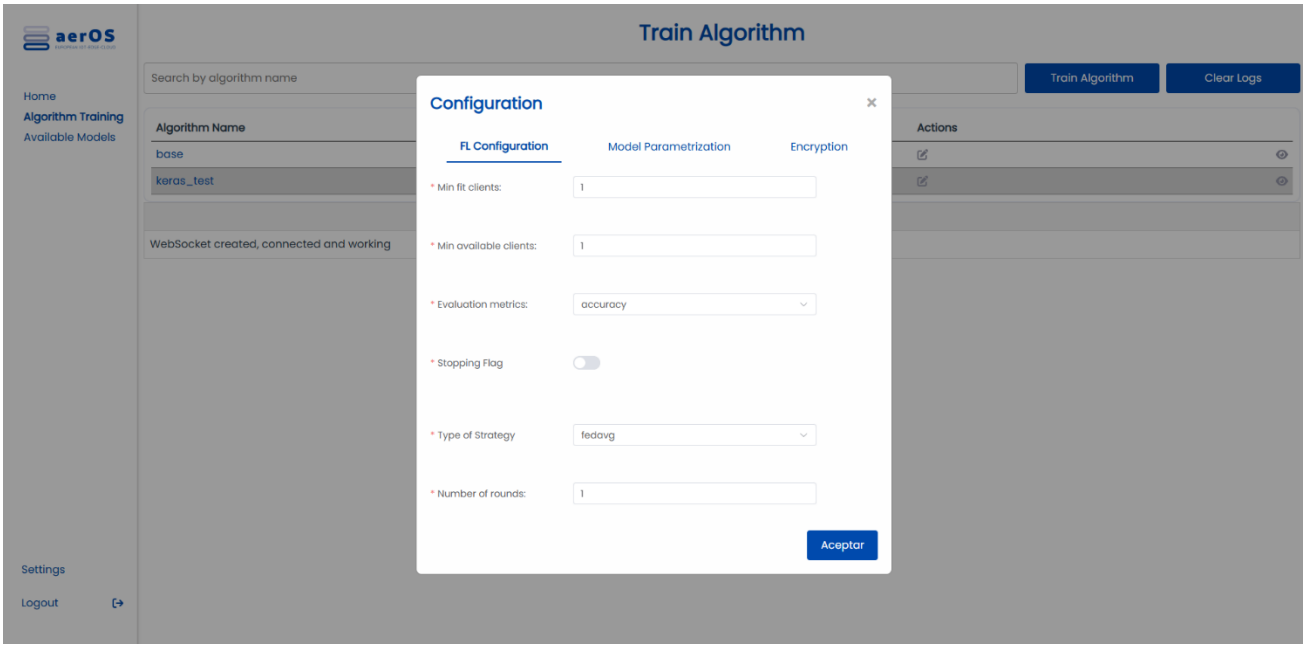


Figure 40. FL GUI configuration for the decentralized AI workflows.

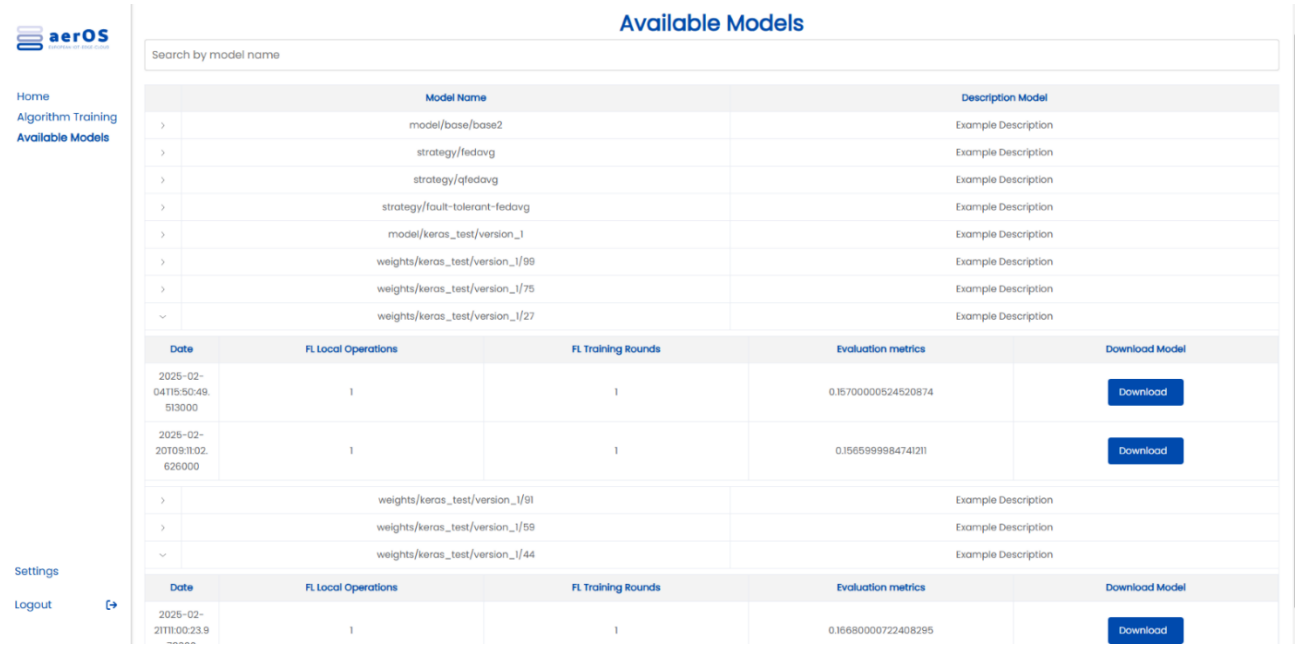


Figure 41. ML models resulted from aerOS decentralized AI workflows, stored in the FL Repository.

3.3.2. Frugal AI – AI Model Reduction

Frugal solutions allow deployment and execution in a resource-restricted environment in terms of memory, processing power, network bandwidth, but also availability of data for model training. Frugal applications can be deemed as the ones most suitable to be deployed close to the edge, instead of a centralized deployment. Consequently, frugality can be treated as an "add-on" to decentralized AI.

In the scope of T4.3 the following techniques have been implemented and tested to validate their applicability in the edge deployment scenarios.

Pruning in neural networks is a technique used to reduce the size of a model by removing parts of the network that contribute little to its output. The main goal of pruning is to improve the efficiency of neural networks without significantly sacrificing accuracy. This technique can be essential for deploying models on devices with limited computational resources, such as smartphones or embedded systems.

There are various benefits of using the pruning technique. It reduces the number of parameters in the model, which decreases its storage requirements. Next, with fewer calculations, the pruned network can offer faster inference times, making it more suitable for real-time applications. Smaller models require fewer computational resources, which can lead to lower power consumption. Lastly, pruning can help reduce overfitting by removing unnecessary parameters, leading to models that generalize better on unseen data.

There are two main approaches to pruning. The first one is unstructured pruning, which focuses on removing individual weights or connections within the network based on specific criteria, typically their magnitude. In this approach, weights deemed less important (e.g., those with values close to zero) are set to zero, eliminating their influence in the network. The second approach is structured pruning, which involves removing entire units or sets of connections, such as neurons, channels, or even layers. This type of pruning is guided by the structure of the neural network itself, aiming to simplify the architecture in a way that maintains its integrity while reducing complexity.

One must note that unstructured pruning's effectiveness in accelerating inference is highly dependent on the availability of specialized hardware or software that can exploit the sparsity of the model. Structured pruning, however, typically results in models that can be more readily accelerated by standard hardware because the pruned model retains a dense structure. Both methods aim to preserve the model's accuracy as much as possible. However, structured pruning may sometimes lead to a more significant drop in performance than unstructured pruning due to its coarse-grained nature.

Both structured and unstructured pruning have been implemented. The methods have been tested on neural networks involving recurrent neural networks (RNNs) and convolutional neural networks (CNNs). The results for structured pruning are very promising as they lead to a significant increase in inference speed and model size reduction. Nonetheless, more work must be done to explore the aspects of generalizability of the implemented methods and ways of applying them in aerOS as a service for users.

The following method that has been deeply explored is **quantization**, a technique used to reduce the precision of the numbers representing the weights and activations within a model. Quantization works by mapping a large range of values to a smaller one, often through rounding operations. This process is vital for deploying deep learning models on resource-constrained devices such as mobile phones, embedded systems, and IoT devices, as it can significantly reduce the model's memory footprint and speed up inference while maintaining acceptable levels of accuracy. The main challenge in quantization is maintaining the model's accuracy with reduced numerical precision, which requires careful selection of the quantization scheme and possibly adjustments to the model or training procedure.

Quantization techniques have been applied in a practical setting. These include static quantization, dynamic quantization, and their combination. The test results are reassuring, proving that this method can be applied to different neural network architectures, speeding up the inference and reducing the model size.

Knowledge distillation is a model compression technique that facilitates the transfer of knowledge from a larger, more complex model to a smaller, more efficient model. In this approach, a pre-trained model with high capacity, often referred to as the "teacher", serves as a guide for training a "student" model that has a reduced number of parameters and computational requirements. The student model is trained not solely on the original dataset labels but also on the soft predictions provided by the teacher, which encapsulate richer information about the underlying data distribution. This process enables the student model to approximate the teacher's performance despite its simpler architecture, making it well suited for deployment in environments where computational resources are limited. By capturing the nuances of the teacher's behaviour, knowledge distillation helps maintain a high level of accuracy and generalization while significantly reducing model size and inference time. This technique has been applied during the experiments. The results proved that the method impact positively the quality of smaller, student networks.

3.3.2.1. Experimental results

In order to validate the behaviour of mentioned methods, experiments were conducted using the dataset with required characteristics (correspondence to dataset that can be processed in edge computing scenarios). The dataset used for the experiments should be structured as datasets that can be processed in potential aerOS deployment (e.g. monitoring readings with the aim of anomaly detection). Requirements:

- Time-series data – in edge environments data are often read from sensors or come from continuous observations.
- Spikes – unusual spikes (or drops) can signal system issues.
- Noise – the data coming from the observations usually contains noise.

For the experiments the PTB-XL[19] dataset was chosen as a good representative of potential datasets that can be used in edge computing-based systems.

The tests were run against CNN, RNN, and ResNet architectures. In the experiments, pruning, knowledge distillation, and quantization have proven effective in reducing model size and computational requirements while maintaining satisfactory performance. However, the results were not consistent in every case. E.g., quantization led to a slowdown in case of small convolutional models, quantization in case of RNN was rather a poor choice on the tested processor (Intel i9) as the floating-point version of the model had access to faster kernels for the LSTM layer, too aggressive pruning could lead to a poor fit of the models. And most importantly the results are use-case and hardware specific. The Table 14 presents a concise summary of the results obtained during the experiments. Columns with arrow pointing up represent the relative increase, and these with the arrow down the relative represent decrease. The relative results are compared against the base model behaviours.

[19] https://www.kaggle.com/datasets/bjoernjostein/ptbxl-electrocardiography-database

Table 14. Extract from AI model reduction experiment.

Model	Parameters	$\text{Speed}_q \uparrow$	$\text{Size}_q \downarrow$	$\text{AUROC}_q \downarrow$	$\text{Speed}_p \uparrow$	$\text{Size}_p \downarrow$	$\text{AUROC}_p \downarrow$	Parameters_p
CNN	37K	x1.15	x3.07	x0.99	x1.61	x13.80	x0.93	1.66%
RNN	23K	x1.16	x3.00	x0.99	x1.18	x14.39	x0.95	3.91%
ResNet1D	500M	x1.36	x3.68	x0.99	x12.97	x37.23	x0.98	1.23%

- q – quantization (float32 -> int8)
- p – pruning
- Speed – how much the inference speeds up
- Size – reduction of disk size
- AUROC – reduction in value

During the experiments and further research, it was established that each of the methods requires a considerable customization for each model (considering its domain, i.e., data or task type) and the target hardware capabilities along with software available during the inference time. Any simplification of the customization could be deemed as unacceptable, making the potential service offering generic methods unusable. Later, neural architecture search (NAS) was considered as another option for the service abstracting the algorithmic task of compressing the model from the use case. Yet, the runtime of the algorithms, even for simple models, is too long or resource expensive, and again the needed customization in a practical, not theoretical, scenario is difficult to achieve. Thus, the implementation of the model reduction as a service was declared as unachievable. The outcome of the experimentation is a set of observations and guidelines that can be useful when considering techniques to be used for a specific use case. The application of frugality techniques is recommended to be performed “inside” the AI implementation for a given use case with careful verification which methods give satisfying results.

3.3.2.2. Technologies and standards

Table 15. Technologies considered for AI Model Reduction.

Technology/ Standard	Description	Component
PyTorch	PyTorch is an open-source machine learning library based on the Torch library, widely used for applications such as computer vision and natural language processing. Initially, it was developed by Facebook's AI Research lab. The library was used to define the model architectures and create the training environments. It was used both for implementing pruning and quantization methods.	Pruning, Quantization
Python	Python is a high-level, interpreted programming language known for its clear syntax, readability, and versatility. The programming language was used to implement the experiments.	Pruning, Quantization
Neural Compressor	The Neural Compressor is a tool provided by Intel that focuses on compressing and optimizing deep learning models to improve their performance and efficiency, especially on Intel hardware. It was used for quantization purposes.	Quantization
ONNX	ONNX, which stands for Open Neural Network Exchange, is an open-source format for AI models. It provides a platform-agnostic way of representing deep learning models, enabling them to be used across different frameworks, tools, and hardware without being tied to one ecosystem. In addition to its core functionality, the ONNX ecosystem includes tools for optimizing models and ONNX	Pruning, Quantization

	Runtime, a performance-focused engine for running ONNX models. ONNX was used to quantize models and to run them.	
NNI	NNI, which stands for Neural Network Intelligence, is an open-source AutoML toolkit developed by Microsoft. It aims to help users automate the machine learning lifecycle, including feature engineering, neural architecture search, model compression, and hyper-parameter tuning. It was used for pruning techniques.	Pruning

3.3.3. Explainability support - AI Explainability Service

In aerOS, AI is used internally to support intelligent decision making when managing the continuum and can be used externally to enable running of arbitrary AI using aerOS infrastructure. In both cases, the need may arise to explain and/or interpret predictions made by ML models. To this aim, an approach based on a dedicated service is proposed to enable "plug-in" of explainability/interpretability step. The AI Explainability Service handles predefined cases like the interpretability of HLO allocator decisions. However, it will also provide methods that can be used for a more comprehensive number of use cases.

An AI-related service must meet some requirements to use the AI Explainability Service.

The first requirement is to prepare a small representative dataset (the ground dataset). The bigger it is, the more reliable the output of the AI Explainability Service. A rule of thumb is to prepare something a size of 100 data samples. These could be the training data. However, the exact size may depend on the service's data availability, the algorithm's (i.e., AI model) complexity, and other aspects.

Regarding the representative aspect of the data, one can understand it as being typical or average data encountered by the model. The exact way of preparation of the dataset is something to be discovered by a service maintainer to verify that the explanations returned by the AI Explainability Service "make sense" in the specific domain. The Explainability would reuse the dataset for different explanations until a maintainer observes a degradation of the results. For instance, external factors like a data drift phenomenon on the service side could cause that.

Next, the AI Explainability Service, to explain a prediction requires the input and the output. The explainer's internal algorithm requires the original input to perform calculations, which are needed to provide the mathematical explanations of the prediction. The explainer uses the original output of the prediction for visualization purposes.

The following requirement is access to the explained model. This element is a crucial aspect of the AI Explainability Service. A service maintainer decides what part of the model one wants to explain. Let us take, for example, a simple logistic regression. Then, the AI Explainability Service needs access to the whole model to run experiments on it internally. However, suppose that a service uses a more sophisticated algorithm, for instance, some reinforcement learning approach. Suppose a maintainer wants to explain the behaviour of some part of it (i.e., the policy network). In that case, the maintainer must be able to extract this part of it and make it usable by the AI Explainability Service.

The explainer assumes that a model behaves as a function that, for an input, returns an output. Although this may sound simple and obvious, the practical aspect of this realization is much more difficult. Users may want to use various frameworks and programming languages to create their AI models. Furthermore, the AI Explainability Service should focus on providing explanations and not handling various inference runtimes for different users. To this end, the Explainability Service would use a specific interface to which users must adhere while exporting their models. This can be realized by, for instance, Function-as-a-Service (FaaS). However, one must note that different approaches with advantages and drawbacks exist. This aspect is still to be verified in the aerOS ecosystem.

Various SoTA methods for explainability in AI have been analysed. The most promising ones (popular, applicable to a wide variety of cases) are based on calculating Shapley values to provide users with easy-to-understand explanations that are mathematically provable. In this area, algorithms like Kernel SHAP and Deep SHAP (an enhanced version of DeepLIFT) were tested in practical settings. The explanations were generated

for a reinforcement learning algorithm that allocates a set of interconnected tasks to a set of computing nodes to reduce parameters like overall energy consumption. Apart from connecting the reinforcement learning algorithm with an explainer, the visualization of the results was prepared. So far, the results are auspicious, especially in giving a user a way to interpret the decisions made while allocating the tasks.

Figure 42 shows an importance graph that visualizes the relative significance of different features in influencing a model's predictions. It helps identify which variables have the most impact on the model's output, aiding in feature selection and interpretation.

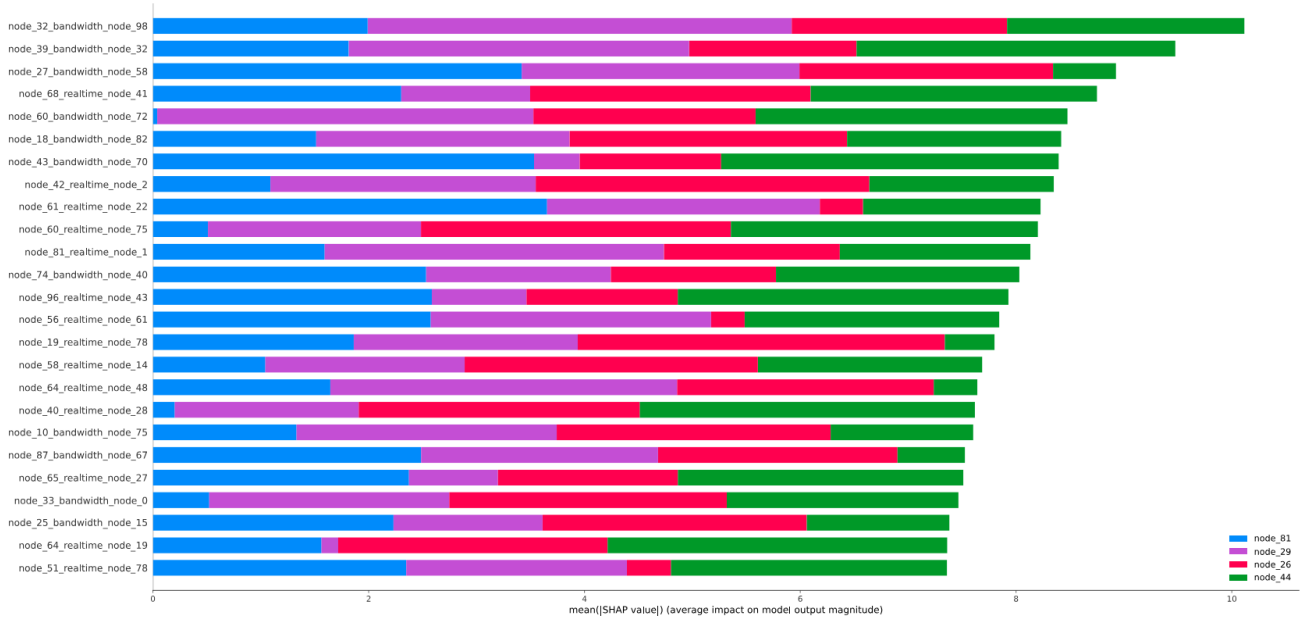


Figure 42. Example of the output for explanation of a single decision made by the HLO, accessible in Embedded Analytics Tool.

To create an explainer the following steps are required:

1. Prepare a set of example data that is representative of the inputs for the algorithm during inference, this is task specific. About 100-1000 examples should be enough.
2. Create a handler that has access to the raw algorithm to be explained, i.e., the handler should be able to call the algorithm directly and treat it as a simple function of the inputs, f(x). The handler should use the Shap Python library to generate the explanations. The output is use-case specific. The example data should be accessible to the handler. For an example, see the explainer for the HLO.
3. Deploy the handler using Embedded Analytics Tool.

3.3.3.1. Technologies and standards

Table 16. Technologies for AI Explainability Service.

Technology/ Standard	Description	Component
Shap	SHAP (SHapley Additive exPlanations) is a Python library for interpreting machine learning models by assigning feature importance scores using Shapley values from cooperative game theory.	Explainer
Python	Python is a high-level, interpreted programming language known for its clear syntax, readability, and versatility. The programming language would be used to implement the solution.	Explainer, API
FastAPI	FastAPI is a modern, high-performance, web framework for building APIs with Python. The framework would be used to create a HTTP interface to the service.	API

Kafka	Apache Kafka is an open-source stream-processing software platform developed by the Apache Software Foundation, written in Scala and Java. Designed to provide a unified, high-throughput, low-latency platform for handling real-time data feeds, Kafka is fundamentally a distributed event streaming platform capable of publishing, subscribing to, storing, and processing streams of records in real time. The message queue would be used for internal communication inside the service, as well as the external communication to store the results.	Message Queue
Celery	Celery is an asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation but supports scheduling as well. The execution units, called tasks, are executed concurrently on one or more worker servers. Celery is used for executing and managing tasks in a distributed fashion, allowing developers to scale their applications easily and process vast amounts of tasks quickly and efficiently. It would be used to schedule the internal computations of the explainer.	Explainer
MongoDB	MongoDB is an open-source, document-oriented NoSQL database designed for ease of development and scaling. It is one of the most popular databases for modern apps, particularly known for its flexible schema, scalability, and performance. The database would store the configuration of the service, as well as some intermediate results.	Database

3.4. Embedded multiplane analytics

The Embedded Analytics Tool (EAT) is a collection of containers with established interfaces to produce a platform for the creation, distribution and deployment of analytical functions to provide insights and advanced decision making to the aerOS Meta-OS. The purpose of EAT is to allow for the utilisation of additional/specialised operations for predefined use cases such as those described in the project pilot demonstrations. Additionally, the flexible and modular nature of "Function-as-a-Service" while being part of the cluster allows for EAT to be called on to fill operational gaps or extensions for future adaptations or use cases. The EAT also provides visualisation features through Grafana for user friendly dashboard creation, and web page hosting for technical experts who require full customisable visualisation capabilities.

3.4.1. Architecture

The EAT is designed as a collection of open-source containers that have been adapted and coordinated to provide complete embedded multiplane analytics for the aerOS system accessible through user friendly dashboards.

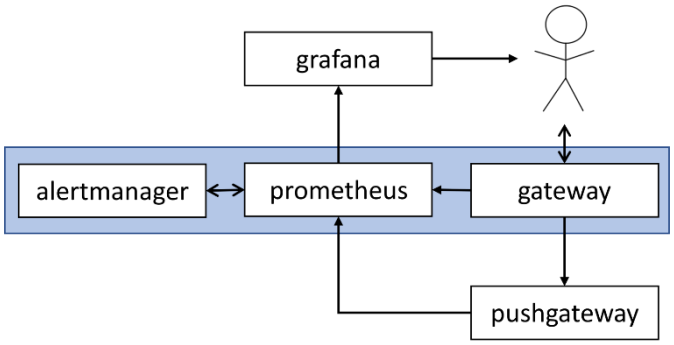


Figure 43. The Embedded Analytics Tool Architecture.

The Embedded Analytics Tool is divided into subcomponents for easier management. The architecture of the tool, as shown in Figure 43, is referenced in Table 17 which provide detailed descriptions of each subcomponent with the interfaces and communication between components. The subcomponents, highlighted within the blue box, consist of images included in the OpenFaaS collection. From a user perspective, the key subcomponent is

the gateway. Through this component, users can access the dashboard interface, view deployed functions, and manually trigger those functions via webpage buttons. Through this gateway, new deployed functions are created as containers in the cluster running docker images based on our aerOS template. All subcomponents are sourced from the open-source community as independent container images, with communication configured using helm charts. These images are customized for aerOS and deployed on Kubernetes clusters. The interfaces between the subcomponents are predefined and static, allowing them to be configured during the helm installation process. However, the functions themselves are dynamic, enabling real-time deployment, removal, or modification, and the ability to establish new interfaces within a function. These interfaces are crucial for processes like data retrieval from the Data Fabric or triggering actions through the High-Level Orchestrator (HLO). The EAT platform has been finalised and delivered via the project GitLab.

Table 17. Components for Embedded Analytics Tool.

Component	Description	Interactions
gateway	The gateway hosts the deployed functions on the aerOS Embedded Analytics Tool. It exposes the functions through an HTTP API. The gateway also hosts a dashboard GUI.	The gateway provides connectivity for the deployed functions. Technical users will establish a HTTP connection to invoke functions and retrieve the results Non-technical users can utilise a GUI to invoke functions
prometheus	Prometheus provides monitoring of the aerOS Embedded Analytics Tool.	Prometheus monitors the gateway and pushgateway components through a scraping process on a regular interval. Metrics may be exposed through the Prometheus dashboard.
alertmanager	Alertmanager is a feature component of Prometheus. This allows for the creation of alert criteria around metrics gathered from the gateway and pushgateway component.	Alertmanager interacts directly with the Prometheus component. Alerts can be configured and communicated to external components when they are triggered.
pushgateway	Pushgateway is a feature component of Prometheus. This allows for the exposing of in-function metrics to Prometheus. As Prometheus scrapes metrics at an interval it is possible that several functions may execute and close within that time frame. Pushgateway hosts this information beyond the lifecycle of the function so it may be gathered.	Pushgateway communicates directly with executing functions on the aerOS Embedded Analytics Tool exposing their in-function metrics to Prometheus.
grafana	Grafana provides visualization features to the aerOS Embedded Analytics Tool.	Grafana connects directly to Prometheus as a data source. Exposing all metrics to visualization by Grafana. The Grafana dashboard may be used to visualise metrics from the aerOS Embedded Analytics Tool and previously executed functions.

3.4.2. Template

The flexibility of Function-as-a-Service (FaaS) approaches also brings added complexity in terms of resource management, function design, and communication. Resource management challenges can be mitigated by configuring the clusters and utilizing monitoring tools available through aerOS. To address the issues related to function design and communication, templating is employed.

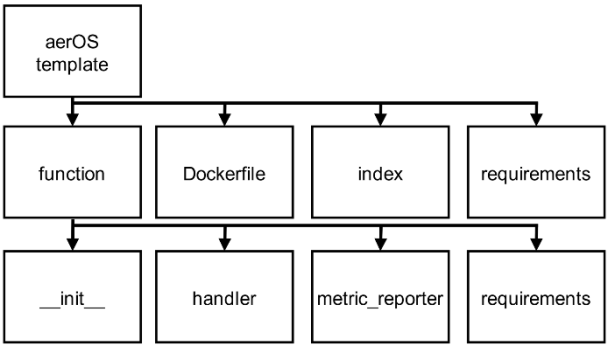


Figure 44. aerOS Template Structure.

Functions for the aerOS system are created using the aerOS template, which offers a standardized structure that abstracts and simplifies common functionality. The function template provides content for 4 scripts which the user has generated at the creation of a new function. The initialiser (__init__) makes connection with the Grafana component, sending the predefined dashboard to be loaded and data sources are created for visualisation. The metric reporter (metric_reporter) provides templated code for the user to export data from inside the function to Prometheus where it can in turn be visualised through the Grafana component. The requirements file (requirements) is a list of libraries to be installed to the function image through the python pip application. Finally, the handler script (handler) is the logic of the function. The handler script is executed once the function is invoked and typically follows three stages of execution: 1) Data Retrieval, 2) Processing, and 3) Response. Data Retrieval provides generalized code to fetch information from data sources, such as the Data Fabric. Processing defines the core logic of the function, where specific operations are executed. Response handles the aerOS-approved interfaces, allowing functions to trigger additional actions, such as invoking other functions or interacting with other aerOS components.

3.4.3. Functions Implementation

The Embedded Analytics Tool comes pre-packaged with three distinct functions: 1) Stratified Sampling, 2) Anomaly Detection, and 3) Data Drift. Each of these functions are created using the faas-cli application and are implemented in Python using a range of data science libraries.

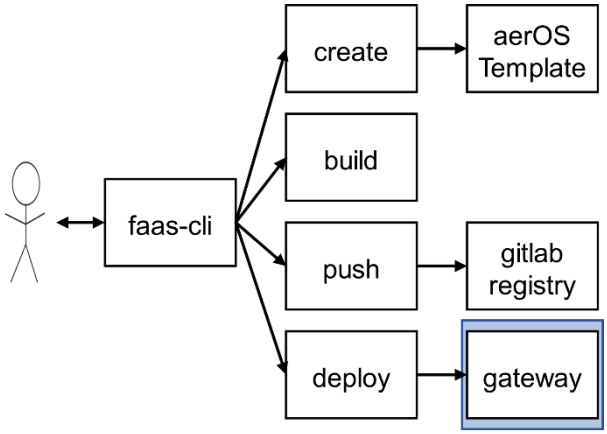


Figure 45. faas-cli application operations.

The faas-cli application utilises our aerOS template to create a new function which can then be built into a Docker image. These images are pushed to the project's Gitlab to be accessible to all aerOS partners. These images can then be deployed onto the EAT gateway from the registry ensuring users always access the latest version of EAT functions. A collection of predefined functions is provided on the Gitlab registry allowing users immediate access to EAT features. Stratified Sampling is a flexible function that generates a data sample based on parameters specified by the user. These parameters include filters for data retrieval and an option to produce

either a proportional or disproportional sample. Anomaly Detection uses a similar data retrieval process but applies different logic to identify outliers in the data sample. This function can be enhanced through training to better suit specific use cases. Data Drift involves a more complex data retrieval process, as it compares current data to historical data to detect variance over time. Depending on the type of drift, some algorithms are better suited to identifying this variance. Our approach uses a distribution-based algorithm to highlight data drift between historical and current data.

3.4.4. Technologies and standards

Table 18. Technologies for Embedded Analytics Tool.

Technology/ Standard	Description	Component
OpenFaaS	OpenFaaS is an open-source Function-as-a-Service platform which utilises containerised docker images as functions to be deployed, executed and managed.	EAT
Prometheus	Prometheus is a popular open-source monitoring component in industry used to expose metrics of internal components and processes.	EAT
Grafana	Grafana is an open-source interactive visualization tool. When linked to a data source Grafana provides diagrams and tables to represent the data.	EAT

3.5. Trustworthiness and decentralized trust management

Trustworthiness and secure communication among the IEs are key features of the aerOS ecosystem. Tasks 4.5 and 3.4 collectively comprise the security aspects of the aerOS architecture. This architecture ensures a robust and trustworthy environment by adopting an effective and comprehensive approach to cybersecurity, utilizing complementary security-enabling components and multi-layered solutions.

The task aims to develop two components: the aerOS Trust Manager, responsible for the dynamic assessments of the IEs, and the IOTA, ensuring decentralization, data integrity, scalability, and efficiency in data exchange. This section presents a comprehensive summary of the final progress achieved in Task 4.5, emphasizing the key advancements and refinements made since the submission of deliverable D4.2.

3.5.1. Trustworthiness of IEs in the continuum

This document outlines the updates introduced to the Trust Manager since deliverable D4.2, focusing on the calculation of the trust score for Infrastructure Elements (IEs). Key changes include the deprecation of the Trust Agent and the introduction of new sub-scores, as well as adjustments to the overall calculation process. After discussions with the partners of the task, it was decided to enhance the trust score to make it more concrete and improve the aerOS ecosystem's trustworthiness. The previously used Trust Agent module, which played a role similar to the Self-Awareness module, has been deprecated to achieve this. This change was made to streamline functionality, as both modules retrieved overlapping system information from IEs (e.g., CPU usage, memory usage) and stored it in the Orion-LD Context Broker. This data now directly contributes to the calculation of the Reliability sub-score, which forms a part of the total trust score.

In addition, to provide a more comprehensive and granular evaluation of trustworthiness in terms of security and reliability, the trust score has been divided into three sub-scores. These sub-scores introduce new terms and frameworks for assessment, ensuring a transparent, effective, and stronger measure of trustworthiness within the aerOS ecosystem. This division allows for more precise identification of strengths and areas for improvement, further enhancing the system’s overall reliability and security. The three sub-scores are:

Reliability Sub-score (SBrel)

The Reliability Sub-score evaluates the performance and stability of Infrastructure Elements (IEs) based on key system metrics such as CPU and memory usage. The TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) algorithm is used to calculate this score, as it is well-suited for multi-criteria decision-making. TOPSIS ranks IEs by comparing their performance to an ideal solution, ensuring that the evaluation is both systematic and objective.

- **Process:**
    1. The Self-Awareness module collects various system data (e.g., CPU and memory usage) based on predefined configurations.
    2. These data points are updated periodically in the Orion-LD Context Broker.
    3. The Trust Manager retrieves this information and calculates the reliability score using the TOPSIS algorithm for each attribute.
    4. The resulting score represents the reliability of the IE.

Security Sub-score (SBsec):

This sub-score represents the system's short-term state regarding exposure to cyber threats, specifically the occurrence of attacks. The Self-Security module is configured to operate in push mode, transmitting real-time alerts to a designated endpoint in the Trust Manager. These alerts are triggered whenever Self-security generates a notification, enabling immediate processing and integration into the trust evaluation framework.

- **Process:**
    1. The Self-Security module includes an API designed to send real-time alerts to the trust manager, each of which has a severity priority ranging from 1 (least severe) to 5 (most severe).
    2. The Trust Manager calculates the average priority value of these alerts and normalizes it to a scale of 0 to 1.
    3. The normalized value becomes the security sub-score.

Reputation Sub-score (SBrep)

The Reputation Sub-Score represents an overall measure of confidence in the robustness and reliability of a node within the system. It evaluates the node's historical performance by analyzing its exposure to cyber threats and its ability to maintain consistent functionality without frequent failures. This score is designed to reflect how well the node can avoid excessive risk and recover from potential issues, providing a long-term perspective on its trustworthiness. To calculate the Reputation Sub-Score, the Trust Manager interacts again with the Self-Security API. As aforementioned, the Self-Security API supplies input that helps assess the node’s resilience against cyber threats over time, but in this sub-score the timeframe is longer.

- **Process:**
    1. The Self-Security module provides alerts generated over a one-week period via an API.
    2. The Trust Manager retrieves these alerts, calculates the average priority value, and normalizes it to 0 to 1.
    3. This value forms the reputation sub-score.

Penalty Mechanism

A key factor in the trust score calculation is the enforcement of a penalty to account for critical events that impact Infrastructure Elements (IEs). These events are identified through self-healing alerts over a one-week period and are incorporated into the trust score to reflect the system's operational reliability.

The penalty is determined using the formula:

Total Penalty = **Health Penalty** × **Number of Alerts**.

In this function, the Health Penalty represents a predefined cost assigned to each alert, and the Number of Alerts corresponds to the total alerts generated during the evaluation period. This mechanism ensures that recurring or

critical issues are effectively accounted for in the trust evaluation, encouraging proactive maintenance and timely resolution of incidents.

Weight Mechanism

To introduce flexibility into the trust score calculation, a weighting system has been designed to allow the Trust Manager to prioritize specific aspects of trustworthiness based on the system's operational goals. By adjusting the weights assigned to different components, such as reliability and security, the system can adapt to varying priorities and use cases.
A key constraint of the weighting system is that the sum of all weights must equal 1, expressed as **Wrep + Wsec + Wrel = 1**, where **Wrep** represents the weight for the reputation score, **Wsec** the weight for the security score, and **Wrel** the weight for the reliability score.
This approach enables tailored trust evaluations for different environments. For instance, in reliability-focused environments, a higher weight can be assigned to **Wrel**, emphasizing performance and stability metrics. Conversely, in security-critical environments, increasing **Wsec** allows for greater emphasis on mitigating security risks. This flexibility ensures that the trust score remains relevant and aligned with the specific goals and challenges of the operational context.

Total Trust Score Calculation

The total trust score (TS) provides a comprehensive evaluation of the trustworthiness of Infrastructure Elements (IEs) by combining multiple sub-scores and applying a penalty for critical events. This score ensures a balanced assessment of reliability, security, and reputation, tailored to the specific priorities of the system.

The total trust score is computed using the following formula:

$$TS=(Wrep\times SBrep) + (Wsec\times SBsec)+(Wrel\times SBrel) - Penalty$$

Where:

- **Wrep, Wsec, Wrep**: Weights assigned to the Reputation, Security, and Reliability sub-scores, respectively.
- **SBrep, SBsec, SBrel**: The Reputation, Security, and Reliability sub-scores.
- **Penalty:** A deduction based on the frequency of critical self-healing alerts, reflecting the stability of the IE.

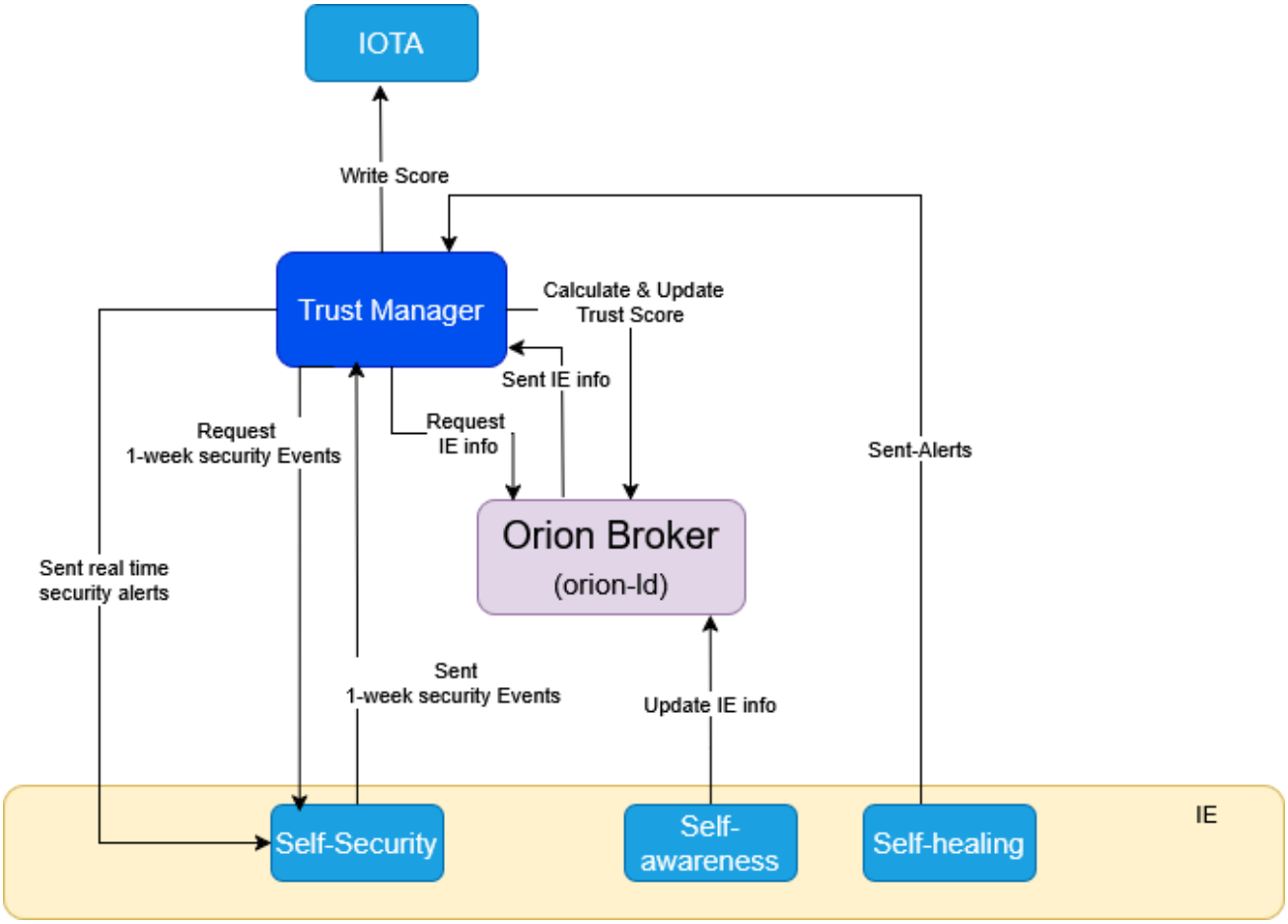


Figure 46. Trust Manager architecture and overall workflow for trust score calculation.

The Figure 46 illustrates the workflow for calculating the trustworthiness of an Infrastructure Element (IE) within the aerOS ecosystem. Only the total trust score is stored in Orion-LD and IOTA for broader access and integration. This approach ensures that the most relevant and actionable data is readily available while minimizing storage requirements. By restricting Orion-LD and IOTA storage to the total trust score, the system significantly reduces unnecessary data transmission, enhancing scalability and efficiency. All sub-scores, including reliability, security, reputation, and health alerts, are stored locally within the Trust Manager.

These data points are maintained as JSON files, which are effectively managed by the Trust Manager. This lightweight yet comprehensive storage solution provides detailed insights and maintains a historical record for in-depth analysis and retrospective evaluations. The Trust Manager prioritizes dynamic responsiveness by integrating event-triggered recalculations with fixed interval updates, ensuring trust scores remain adaptive and accurate in a rapidly changing environment. Real-time, event-driven updates allow the system to swiftly respond to critical changes, while periodic scheduled updates maintain a consistent baseline for recalculations.

3.5.1.1. Technologies and standards

Table 19. Technologies for trustworthiness.

Technology/ Standard	Description	Justification
Python	A high-level programming language.	Python is widely used due to its ease of use, extensive libraries, and strong community support.
TOPSIS (Technique for Order of	A multi-criteria decision-making (MCDM) method used to rank alternatives	TOPSIS is effective in evaluating multiple criteria, making it suitable for computing trust subscores by considering various weighted factors

Preference by Similarity to Ideal Solution)	based on their similarity to an ideal solution.	

3.5.2. Trustful decentralized exchange: IOTA

IOTA provides a shared network between all nodes called the Tangle, which allows for secure and trustworthy feeless data transactions between different nodes. This is the root of the IOTA deployment in aerOS, where the Hornet Nodes are the multiple IEs found in the shared continuum and the Tangle is the data structure that contains all the information necessary to track messages and ensure traceability of the payloads distributed across the network. When one program issues a message to a node, said node verifies the message and sends it via the gossip protocol through the network to its 'neighbours', other nodes in the Tangle network connected to the first one. All other nodes in the network see the message and retrieve the same status of the network. The data shared here is only meant to be the most important of data regarding the state of the network. This data could be, for example, the IP address of elements in the continuum, the domain addresses, etc. With this in mind, the deployment of IOTA's nodes is done accordingly with both the domain and continuum requirements.

A deployment of a private IOTA Tangle network has been done in the CF and NCSRD environments, with the CF Kubernetes cluster acting as the main cluster and the NCSRD cluster acting as the secondary cluster. Successful tests with data transfer between the nodes have been performed, with the results being shown in the DLT status and the capability of data to support multiple formats. Afterwards the deployment was done spreading the nodes across multiple test domains that would emulate a real use case situation, with all the parts of the network involved. A diagram of the setup can be seen below, with all icons taken from IOTA:

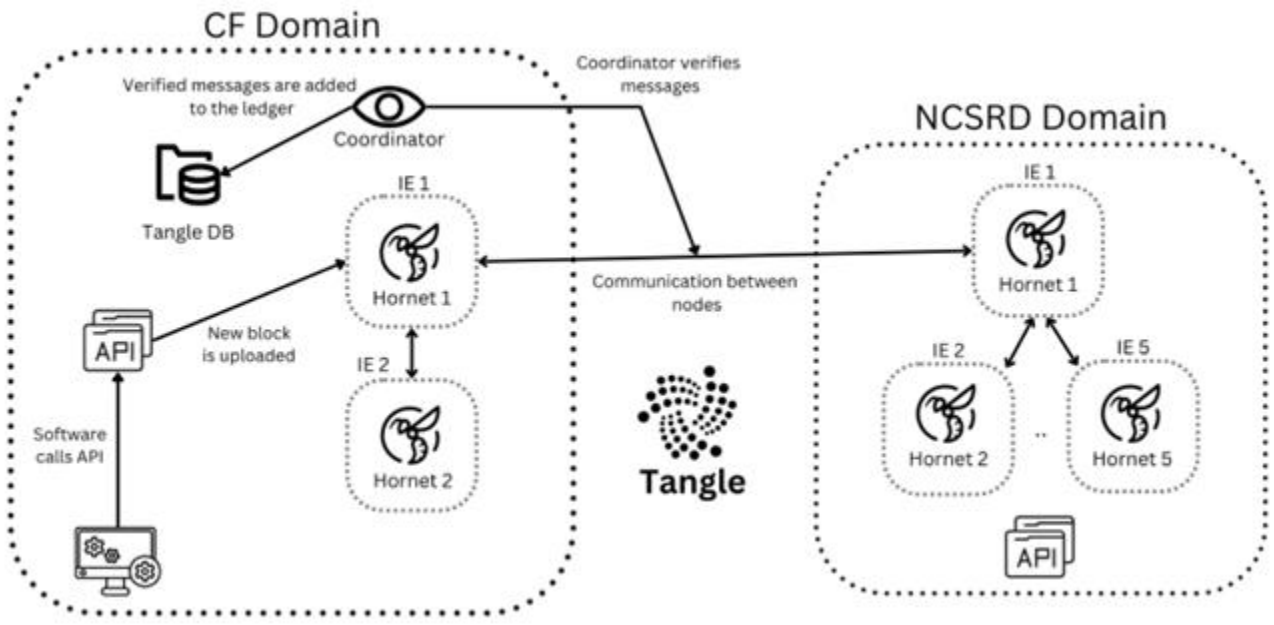


Figure 47. aerOS testing IOTA Tangle.

A few elements have been changed or upgraded throughout the project. The entire installation was automated as much as possible via a Helm chart. A Kubernetes automatic peering element has been developed, where the main node of each cluster automatically links with all other hornet nodes of the cluster.

This way the user only needs to link the different domains between one another. Another development was the creation of an API that simplifies the process of interaction with the hornet nodes of the cluster. This allows the users to upload to the Tangle in an easier way and keeps a log of the successful uploads. The API works in conjunction with the dashboard to provide all the necessary utilities to use IOTA and the Tangle.

3.5.2.1. Technologies and standards

Table 20. Technologies for trustful decentralized exchange.

Technology/ Standard	Description	Component
IOTA	A decentralized, feeless distributed ledger designed for the Internet of Things (IoT) and secure data transfer.	All components related with IOTA
IOTA Tangle	A directed acyclic graph (DAG) used instead of blockchain, where each transaction confirms two others, enabling scalability and efficiency.	All components related with the data transfers, alongside the Tangle DB
IOTA Hornet	A lightweight, high-performance IOTA node software that helps maintain and validate the Tangle network.	The individual nodes installed in all IEs
IOTA Coordinator	A temporary mechanism that ensures security and finality in the IOTA network until full decentralization is achieved.	The single “special component” installed alongside the Tangle DB

3.6. Management services and aerOS management portal

3.6.1. aerOS Management Portal

The block schema of the aerOS Management Portal was slightly modified for the first MVP when it comes to the interactions between the Backend and the aerOS Basic Services, but this schema has not changed since that point, so this description, which is depicted in Figure 47, is still valid.

The entrypoint balancer it is only needed for distributing the aerOS orchestration requests among all the available domains, but because of its stateless nature it can be used to distribute more requests if needed, so it has been decided to keep this part in the schema. Following this explanation, Data Fabric and aerOS Federation are distributed by definition, hence it is not necessary to add another distribution layer. When it comes to the users and roles management, the aerOS AAA tools to manage them (LDAP and Keycloak) will always be deployed in the entrypoint domain so distribution is not needed.

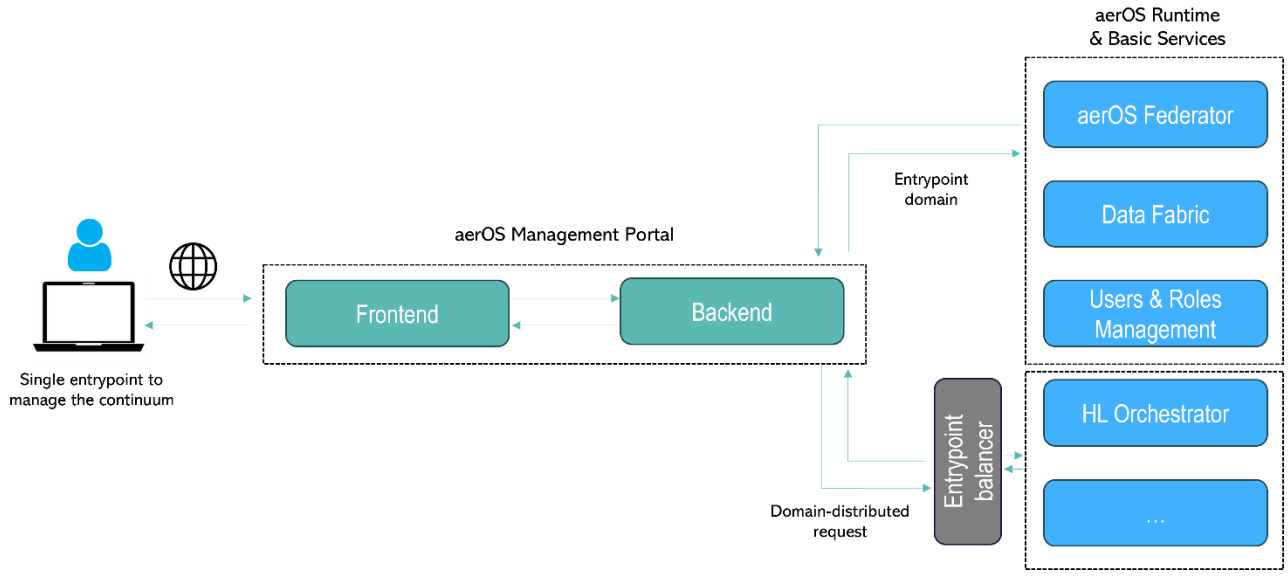


Figure 48. aerOS Management Portal architecture.

3.6.1.1. Frontend

Before developing the code for the Frontend part of the Management Portal, an analysis phase was conducted to define the requirements needed for the Management Portal. Once the requirements were discussed and finally set, these requirements have been used by the UX/UI team as the starting point for drawing the first mock-up designs of the Management Portal using Figma, which is an online collaborative tool for designing user interfaces. Once the mock-ups were approved by the partners and internal team, frontend development began. The management portal is structured as a single-page component-based application built with the popular Vue.js framework (version 3), an industry standard with a comprehensive set of tools for creating web user interfaces. The Vue components has been built with full Typescript support and the application uses Pinia as a store manager, to allow the sharing of states between all the web application components. It is important to highlight that the portal only shows information and allows to perform actions in accordance with the role that has been assigned to the logged user in the aerOS AAA framework (see section 4.4.1.1 of D3.3 for more information about the defined roles in aerOS). For instance, only users with the proper rights can initiate an orchestration request (e.g., a vertical deployer) or check the status of the IEs that belong to his linked domains (e.g., continuum administrator). Therefore, the aspect of the portal changes according to the user roles and permissions.

The IoT-Edge-Cloud continuum presents a heterogeneous range of computing resources, so aerOS as a Meta-OS must provide some supporting tools for depicting this computing resources adapted to the aerOS architecture concepts (Domains, IEs, LLOs, etc). For that reason, after some technical analysis, an interactive network graph has been selected to try to overcome the challenge of drawing the computing continuum. Specifically, the v-network-graph, a lightweight Vue.js fully compatible library, has been used.

Finally, this is the list of the developed user interfaces along with some description:

- **Welcome page and navigation menu**

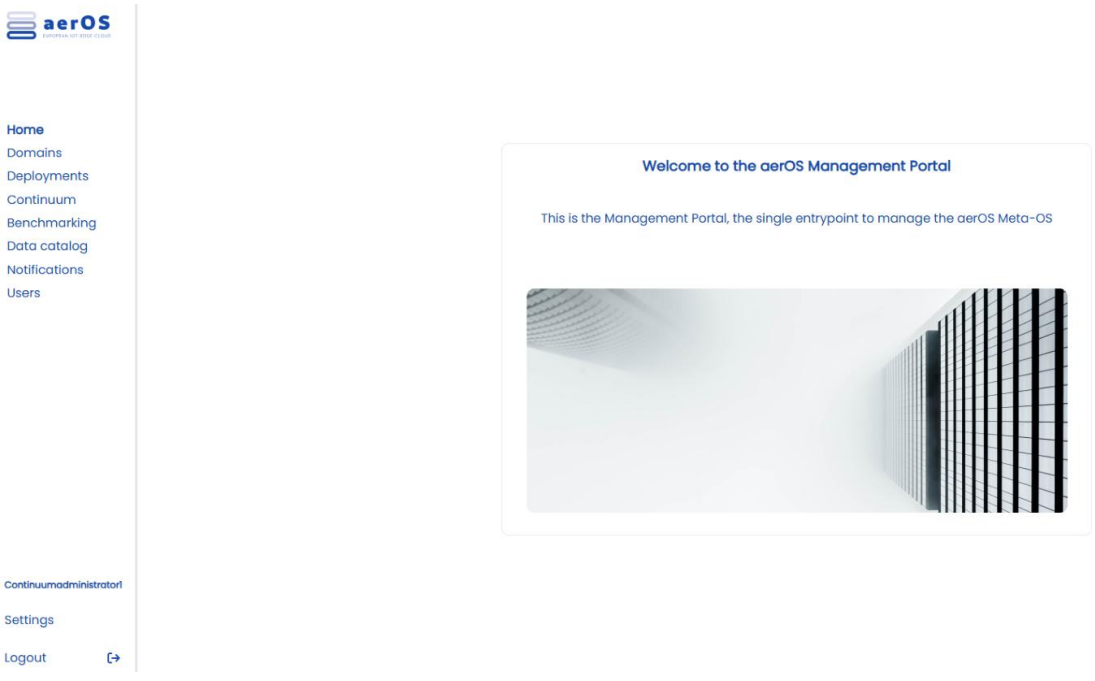


Figure 49. aerOS Management Portal welcome page and navigation menu.

First, users must complete the logging process which is based on the Authorization Code Flow with Proof Key for Code Exchange (PKCE) of the OAuth2.0 protocol implemented by Keycloak. Therefore, when users access to the portal without having been previously logged in, they are redirected to the Keycloak login page to introduce their credentials. After a successful credential's validation, users are redirected to the Management Portal welcome page. This introductory page consists of a central descriptive part and the side navigation menu. When it comes to the navigation menu, the user can choose from several options, the current list follows below:

- **Home**: by clicking this item the user will be redirect to the home page of the Portal.
- **Domains**: clicking on the *Domains* option will direct the user to the section related to his/her domains within the aerOS continuum. In this section the user can go into the details of a selected domain such as the list of the IE that build the domain, along with their metrics collected in real-time.
- **Deployments**: by clicking on the *Deployments* option the user will be redirected to the wizard that list the deployed services and allows to request a service orchestration in the aerOS continuum.
- **Continuum**: in this page, users will be able to see, in a simple and intuitive way, their own relative Infrastructure Elements that comprise their domains: the aerOS continuum indeed.
- **Benchmarking**: a page to display benchmarking results of the IEs, compare them and to run a new benchmark.
- **Data catalog**: this page displays all the registered Data Products in the aerOS Data Fabric and allows to register a new one.
- **Notifications**: this section has been created to inform users that some events have been triggered (the list of events and the development will be performed in the next phase). Moreover, the number of unread notifications is displayed near the notification's entry of the menu. This count is reset each time the user enters to this page.
- **Users**: this view is composed of a list of the registered users in the aerOS Meta-OS with some key information such as their role or status. In addition, new users can be added to the system and existing users can be edited (mainly to change their assigned role) or even deleted.
- **Settings**: a menu to configure some general settings (dark mode, language, font size, …) of the portal.

- **Domains**

Figure 50. aerOS Management Portal domain view.

In the Domains section, the aerOS user accesses a table with a list of federated domains along with a first glance of their characteristics. By clicking on the *view* call-to-action, the user will land on the detail page of the selected domain, where they can browse a complete list of domain information, including the underlying IEs.

- **Deployments**

The Deployments section provides an overview of active service deployments in a structured table format. The table includes key details such as the ID, Name, and Description of each deployed service. The ID column contains unique identifiers, while the Name column displays service names formatted as URNs. The Description column provides additional context about each deployment, such as its purpose or function.

Each row in the table represents a deployed service and includes a set of action icons on the right side. These icons allow users to view details, pause the service, or delete it. A pagination control at the bottom enables users to navigate through multiple pages of deployments.

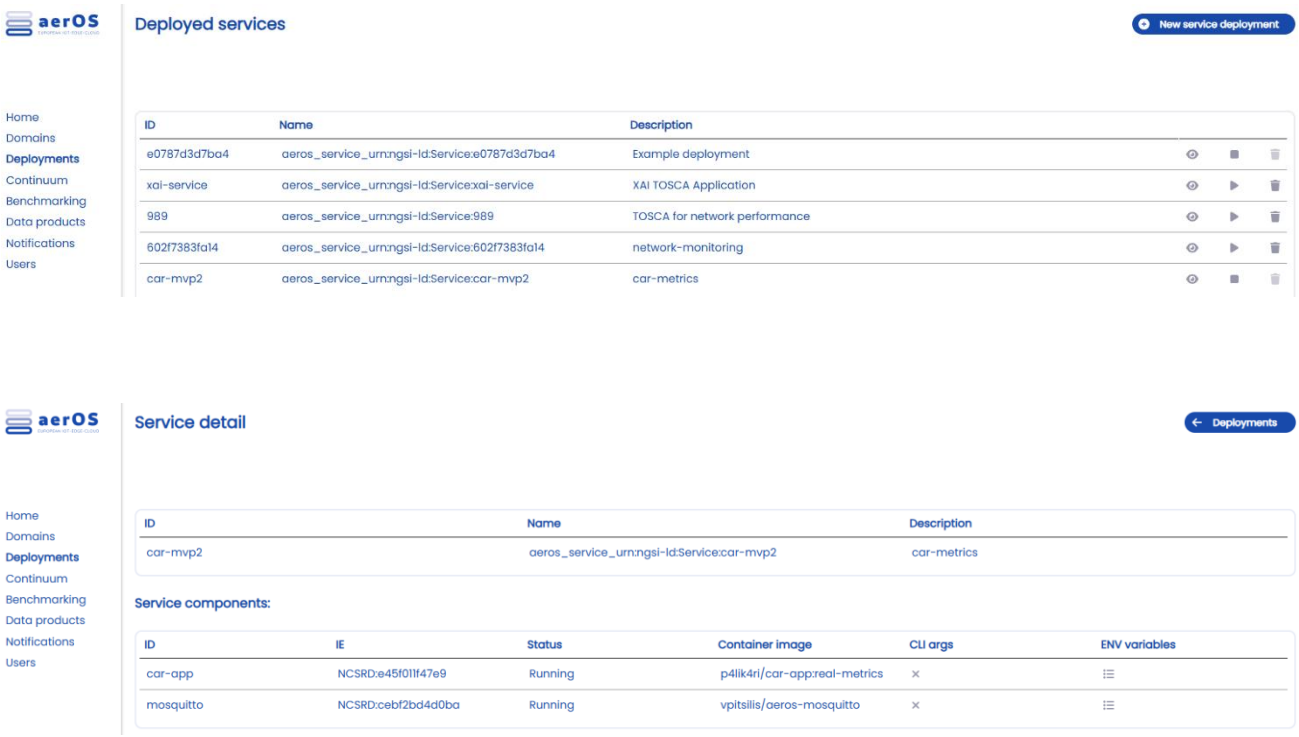


Figure 51. aerOS Management Portal deployment view.

At the top right corner of the section, a "New Service Deployment" button allows users to initiate a new deployment. The interface is designed for easy management, providing administrators with a clear view of active services and essential controls for managing them efficiently.

The steps shown in the image below, involves the configuration of the service component deployment parameters through multiple input fields and selection options. The component's name and a brief description can be specified in their respective text input fields. A field for the container image is provided to define the image to be used for the component.

For resource allocation, the users can select the CPU usage and required RAM from the available dropdown menus. A separate dropdown menu allows the users to choose the CPU architecture, with options such as x64. Additionally, there are toggle options to specify if the component is real-time capable and whether ports should be exposed, with "Yes" or "No" options for each.

At the bottom of the form, buttons are available for configuring command-line arguments, environment variables, and network ports. Once these configurations are complete, the users can proceed by clicking the "Next" button to continue to the next step in the deployment process. This step ensures the proper configuration of various parameters required for the automatic deployment of the service component.

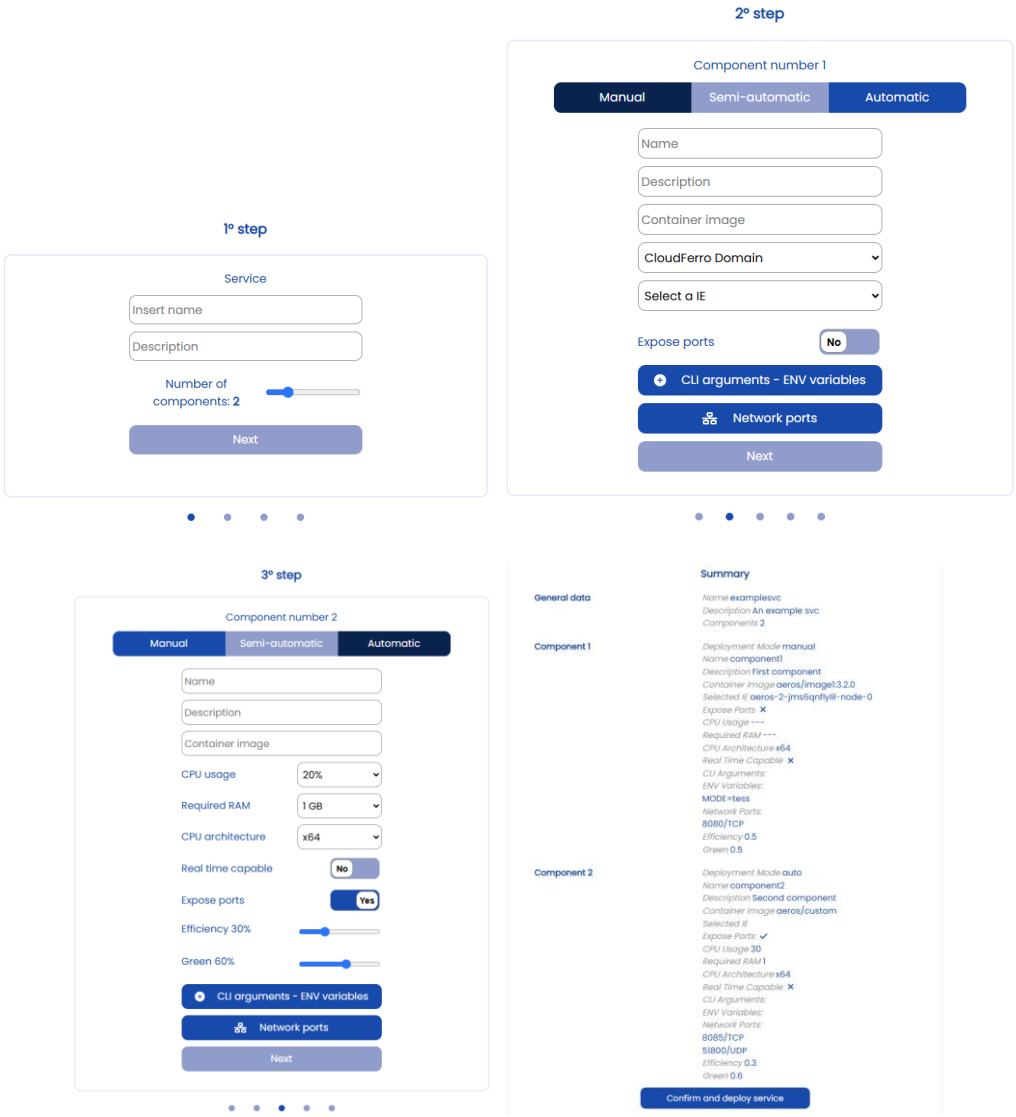


Figure 52. aerOS Management Portal deployment form.

- **Continuum view**

The Continuum section of the aerOS system provides a visual representation of all domains and their corresponding Infrastructure Elements (IEs). This interface is structured as a network graph, where each domain serves as a central node, branching out to its associated IEs. The visual layout enhances the user's ability to understand relationships, dependencies, and system structure at a glance.

Each node in the graph is labelled with relevant details, including the domain or IE identifier and its current status. Domains typically serve as the backbone of the structure, connecting to multiple IEs that handle various processing tasks. The status of each IE is indicated, helping users quickly assess operational conditions, such as whether an IE is functional or in a ready state. Additionally, network addresses and URLs are provided, facilitating direct access to specific components for further inspection or interaction.

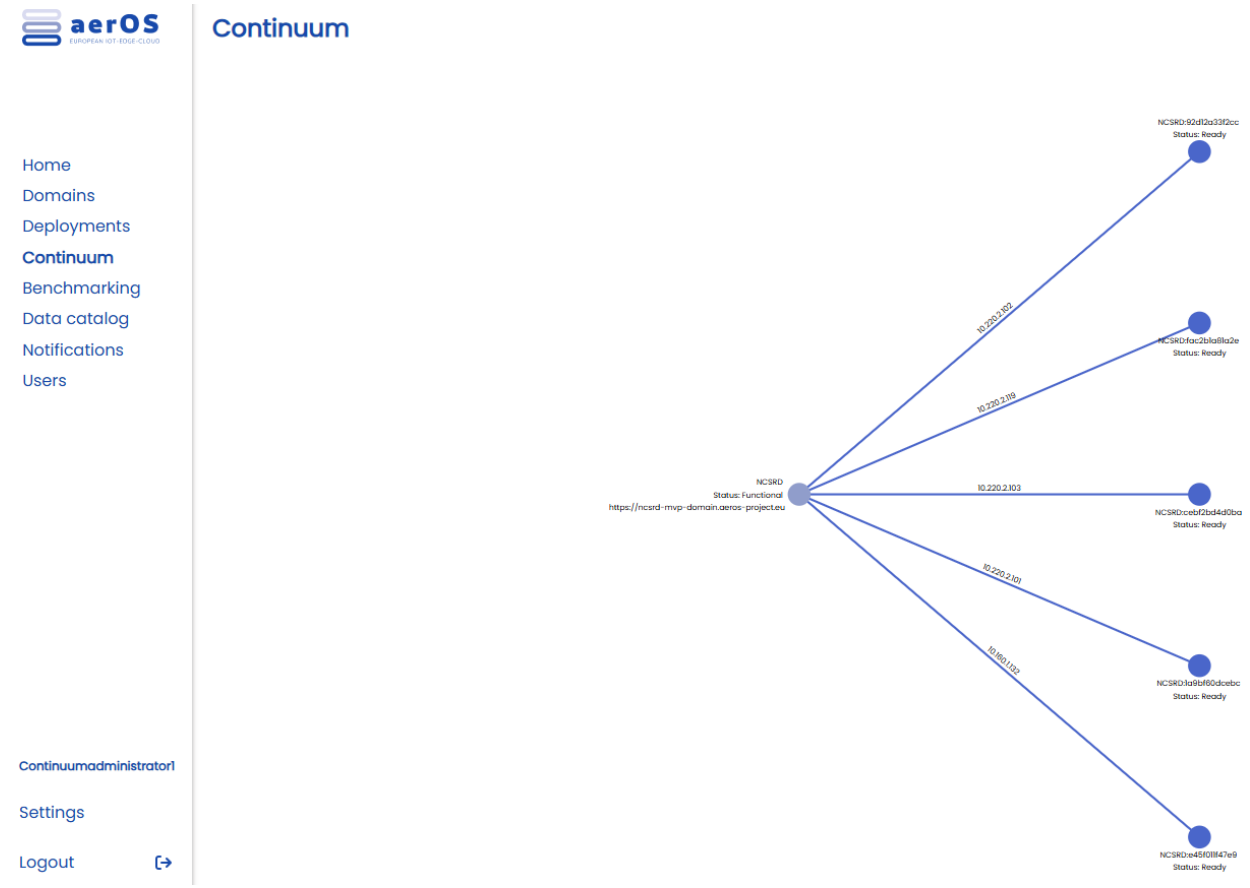


Figure 53. aerOS Management Portal continuum view.

- **Benchmarking**

The benchmarking section showcases the performance of IEs in terms of CPU and network interfaces across various aerOS domains. Users can select an IE within a specific domain to view its performance in common operations and compare it against benchmark results. If the requested benchmarking has not yet been performed, users can initiate its execution to access detailed performance specifications. In addition, the full set of CPU benchmarks results are available on the Geekbench website.

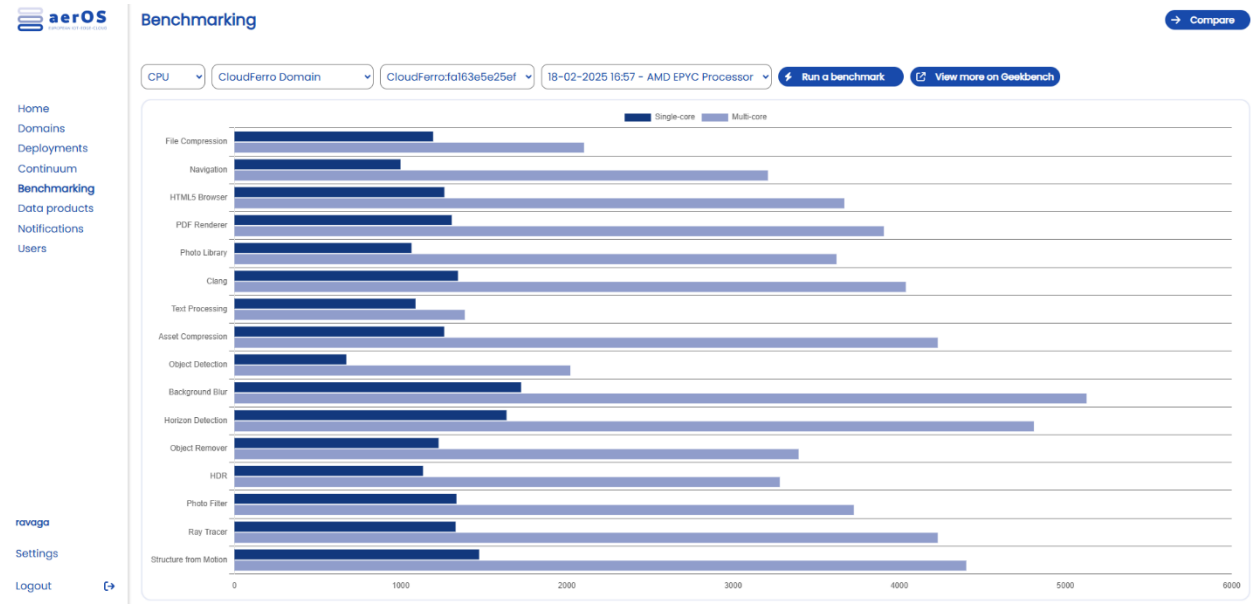


Figure 54. aerOS Management Portal CPU benchmarking view.

Additionally, users can compare benchmarking results using the "Compare" button located at the top right. This functionality allows for the comparison of two different IEs, either within the same domain or across different domains. By leveraging this feature, users can conduct a comprehensive performance evaluation, gaining insights that may highlight opportunities for optimizing their infrastructure.

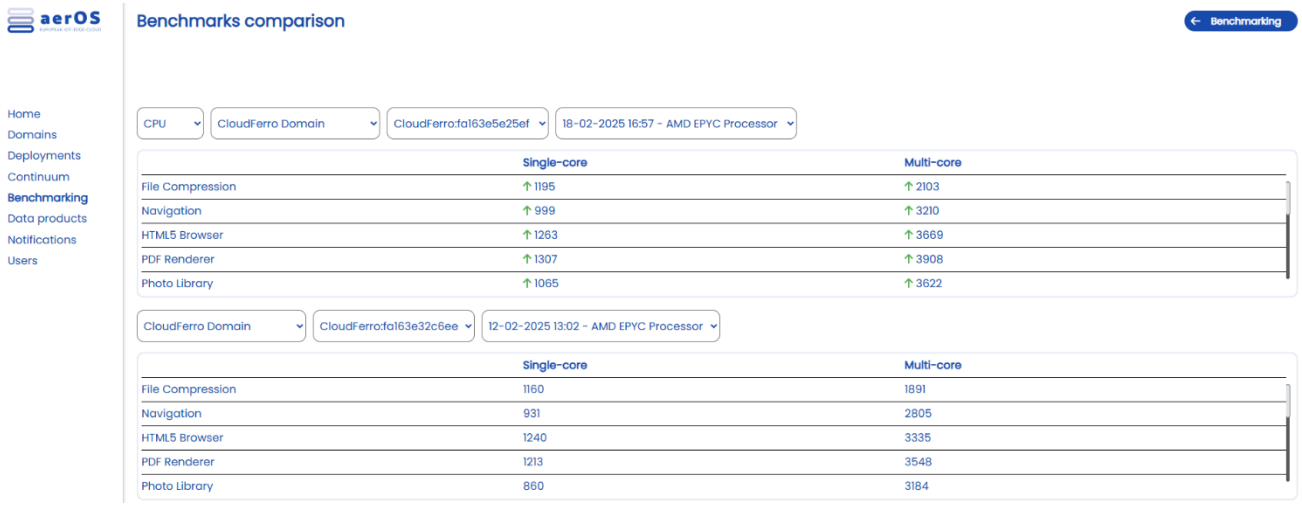


Figure 55. aerOS Management Portal CPU benchmarking comparison view.

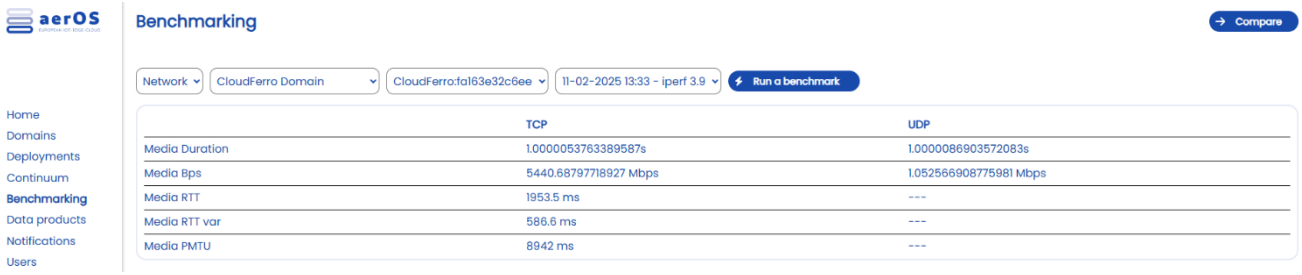


Figure 56. aerOS Management Portal network benchmarking view.

- **Data Catalog**

The Data Products section provides an organized interface for managing and accessing data products. It includes a search bar and several dropdown filters, allowing users to refine their search based on data product names, glossary terms, owners, and tags. Each data product is displayed in a structured card format containing key details such as the product name, owner, description, the context broker in through which is accessible and relevant metadata. The owner is represented with an icon and name, while context broker fields include information like a link to its specification and its endpoint URL.

Tags and glossary terms are visually represented as blue pill-shaped labels, making it easy to identify and categorize each data product. Additionally, each product card features a trash icon for deletion and an option to expand or collapse details. Positioned at the top right corner, the "Add Data Product" button enables users to introduce new data entries. The interface is designed for quick navigation and efficient data management.

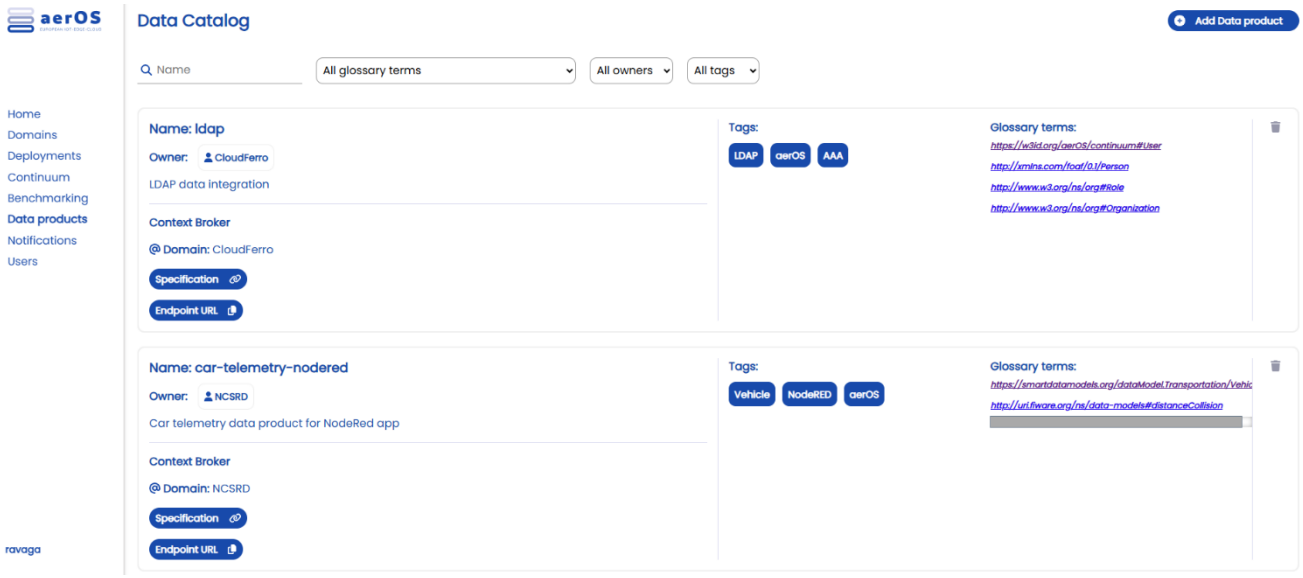


Figure 57. aerOS Management Portal Data Catalog section.

The image below displays the Data Product modal for adding new Data Products, with required fields that vary depending on the selected Data Product type.

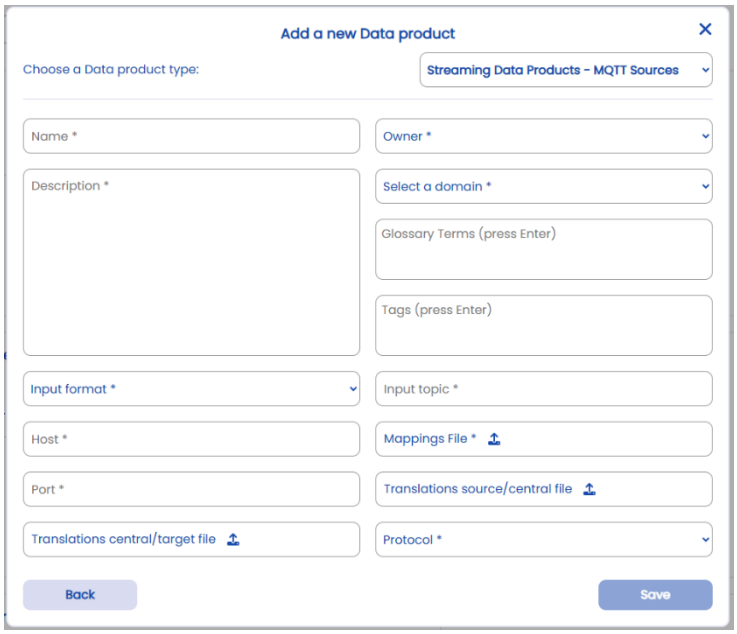


Figure 58. aerOS Management Portal Data Product creation.

- **Users**

The Users Section in aerOS serves as a central interface for managing registered users within the system. This section presents a comprehensive list of users, displaying key details such as their name, assigned role, and current status. The role designation indicates the level of access and responsibilities within the system, including categories such as Continuum Administrator, Data Product Owner, or standard aerOS user. The status field reflects whether the user is active or otherwise restricted.

To facilitate user management, the section includes search and filtering capabilities. A search bar enables quick identification of specific users by name, while dropdown menus allow filtering by role and status, ensuring an efficient navigation experience.

Continuum Administrators can perform various actions on user accounts. Each entry in the list is accompanied by options to view detailed user information, modify user roles or other attributes, and delete accounts if necessary. These actions are represented by intuitive icons that streamline the management process.

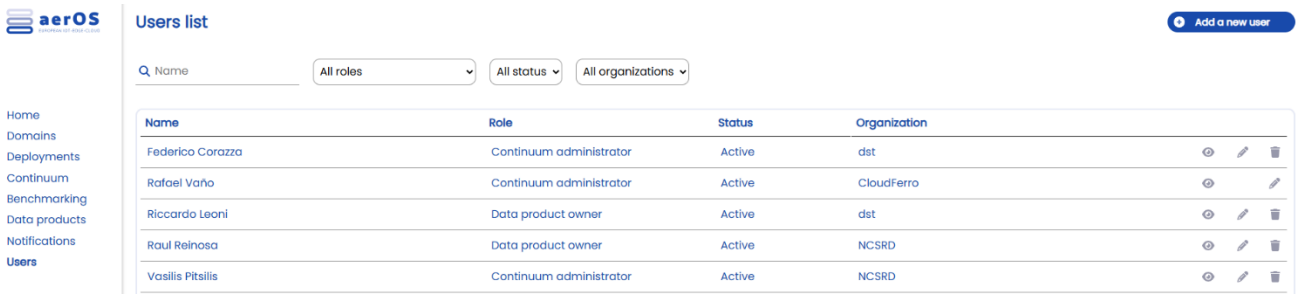


Figure 59. aerOS Management Portal users view.

The section also supports the addition of new users to the system. A call to action, "Add a new user", at the top-right corner provides a straightforward method for onboarding new users. The user creation modal enables administrators to create a new user, assign a role, and associate the user with an existing organization or create a new organization from scratch. This ensures that administrators can easily expand the user base while maintaining control over access levels and system permissions.

To enhance usability, pagination controls are available at the bottom of the interface, allowing users to adjust the number of displayed entries per page and navigate through multiple pages of user records. This structured approach to user management ensures that administrators can effectively oversee access control within the aerOS environment, maintaining a secure and organized system.

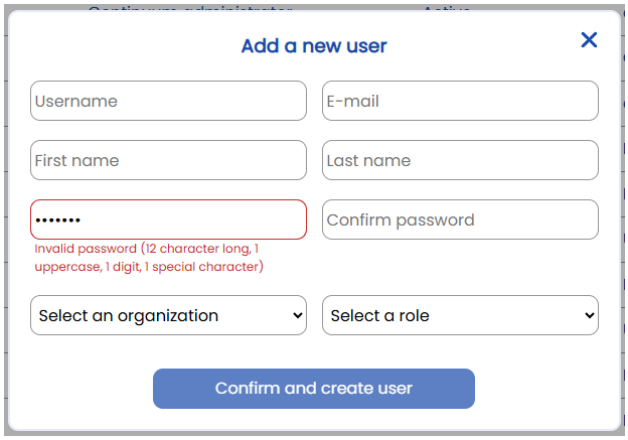


Figure 60. aerOS Management Portal user creation modal.

- **Notifications**

The Notification Settings section provides users with a centralized view of all notifications related to operations performed within the aerOS system. This includes updates on service deployment, benchmarking execution, and other relevant system activities.

Users can filter notifications by priority, ranging from Critical to Informational, ensuring quick access to the most relevant updates. They can view notification details, delete individual notifications, or clear all at once.

This section ensures that users stay informed about key system events, enabling timely decision-making and efficient management of their infrastructure.

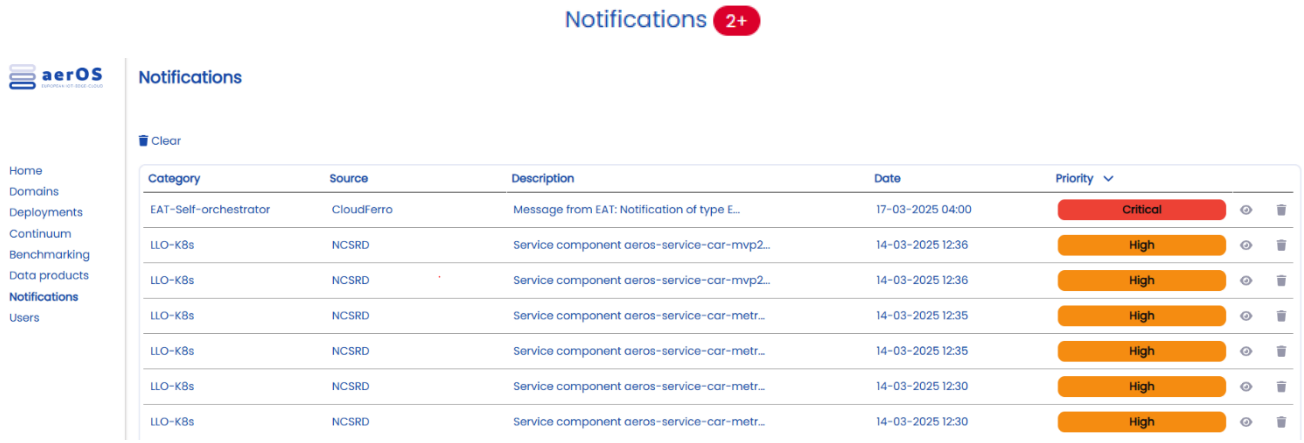


Figure 61. aerOS Management Portal notification view.

- **Settings**

The Settings section allows users to personalize their aerOS experience by configuring display preferences, notifications, and system-related options. In addition, it is displayed a disclaimer.

This section is organized into a main category: General Settings. Within this category, users can customize their experience in several ways. Dark Mode can be toggled on or off to switch between light and dark themes. The Language option provides a dropdown menu for selecting a preferred language, with English set as the default. Font Size can be adjusted using a slider, allowing users to modify text size for better readability. Additionally, Push Notifications can be enabled or disabled with a simple toggle switch.

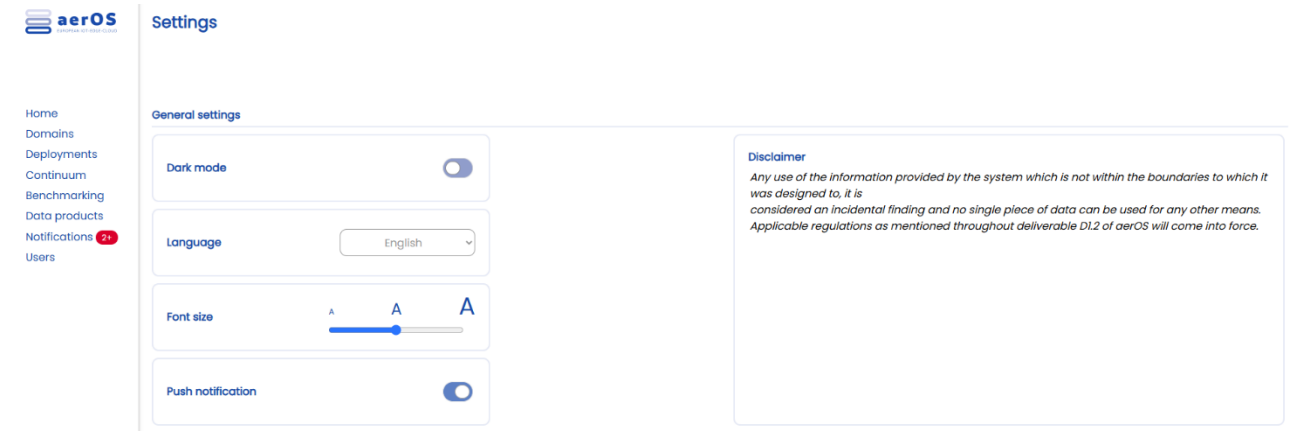


Figure 62. aerOS Management Portal settings menu.

3.6.1.2. Backend

Developing a backend component for a web application depends on the specific requirements and functionalities of this application. In the scope of aerOS, that considers the aerOS Basic Services as a suite of lightweight microservices (including the potential workloads to be deployed in the computing continuum), heavier computing processes (e.g., data processing) must be removed from the web application. Removing this logic from the web application allows to develop the frontend to just react to user actions and obtain the needed data in the expected format to be then efficiently interpreted, as well as avoiding some security issues like the well-known CORS. Therefore, the backend component acts as a middleware between the web page and third-party APIs. This component has been developed as a Spring Framework application, starting the project with Spring Boot and using some complementary libraries such as Spring Security or Spring Cloud.

It is important to highlight that this backend doesn't interact with a database to store configuration or state, which is a common practice in web applications dashboards, as the Management Portal can be moved to other aerOS domains in a flexible way if the Entrypoint domain changes. All the needed data by the portal can be obtained from the aerOS Basic Services in a decentralized and federated way, for instance, users' data is stored in Keycloak Identity Manager, and continuum and services data can be obtained through Orion-LD taking advantage of the Data Fabric mechanisms.

When it comes to cybersecurity, the backend protects the incoming requests by checking authorization JWT tokens with Keycloak. This way, the backend first ensures that these requests are sent by the Frontend to then check if the requested action can be performed by the role granted to the user that initiates actions in the web application. In addition, this token can be reused in case of the backend must send some requests to endpoints protected by KrakenD API Gateway.

When integrating with external systems like OpenLDAP, the backend acts as a bridge that facilitates secure and efficient communication. The backend manages key LDAP entities, including users, roles, and groups, through specific Java classes, while handles operations such as user and group management, role assignment, and querying directory services. This integration ensures that the backend can fetch and update user information dynamically, which is transparently updated in Keycloak to enable the proper creation of access tokens.

The interaction between the backend and OpenLDAP is performed through RESTful API calls, encapsulated within the Spring Boot application and is implemented via dedicated controllers, including UserController, GroupController, and RoleController, which facilitate CRUD (Create, Read, Update, Delete) operations for LDAP entities. These controllers handle HTTP requests for each LDAP entity and interact with the corresponding repository classes (UserRepository, RoleRepository, and GroupRepository), that leverage Spring LDAP ODM (Object-Directory Mapping) to execute LDAP operations. These operations correspond to LDAP actions and ensure seamless management of directory data.

Finally, the backend integrates a WebSocket-based communication channel among it and the Frontend to allow the exchange of real-time messages, such as notifications to be displayed to final users. This channel has been developed using Socket.IO, which is a modern framework that enables low-latency, bidirectional and event-based communication by going a step beyond traditional WebSockets. Notifications are added to the Meta-OS via a POST endpoint that is part of the Backend, which can be called by any component or service of the Meta-OS as long as it has the proper permissions to retrieve a valid access token. When notification is received, the Backend creates a new NGSI-LD entity for it in Orion-LD and then sends it to the Frontend through the Socket.IO channel to be finally displayed in real-time to the users as pop-up notification. As the notification has previously been added into Orion-LD, users will also be able to check it until it's manually deleted. The format of the notifications is aligned with the Alert FIWARE Smart Data Model.

In addition, the backend also supports the reception of NGSI-LD notifications (emitted by Orion-LD when a subscription is triggered) in order to be translated into the format expected by the notifications in the system (the FIWARE Alert data model), to then be added to Orion-LD and displayed to the users in the portal. For instance, the Benchmarking tool leverages this functionality to display notifications each time a benchmark execution has started or finished.

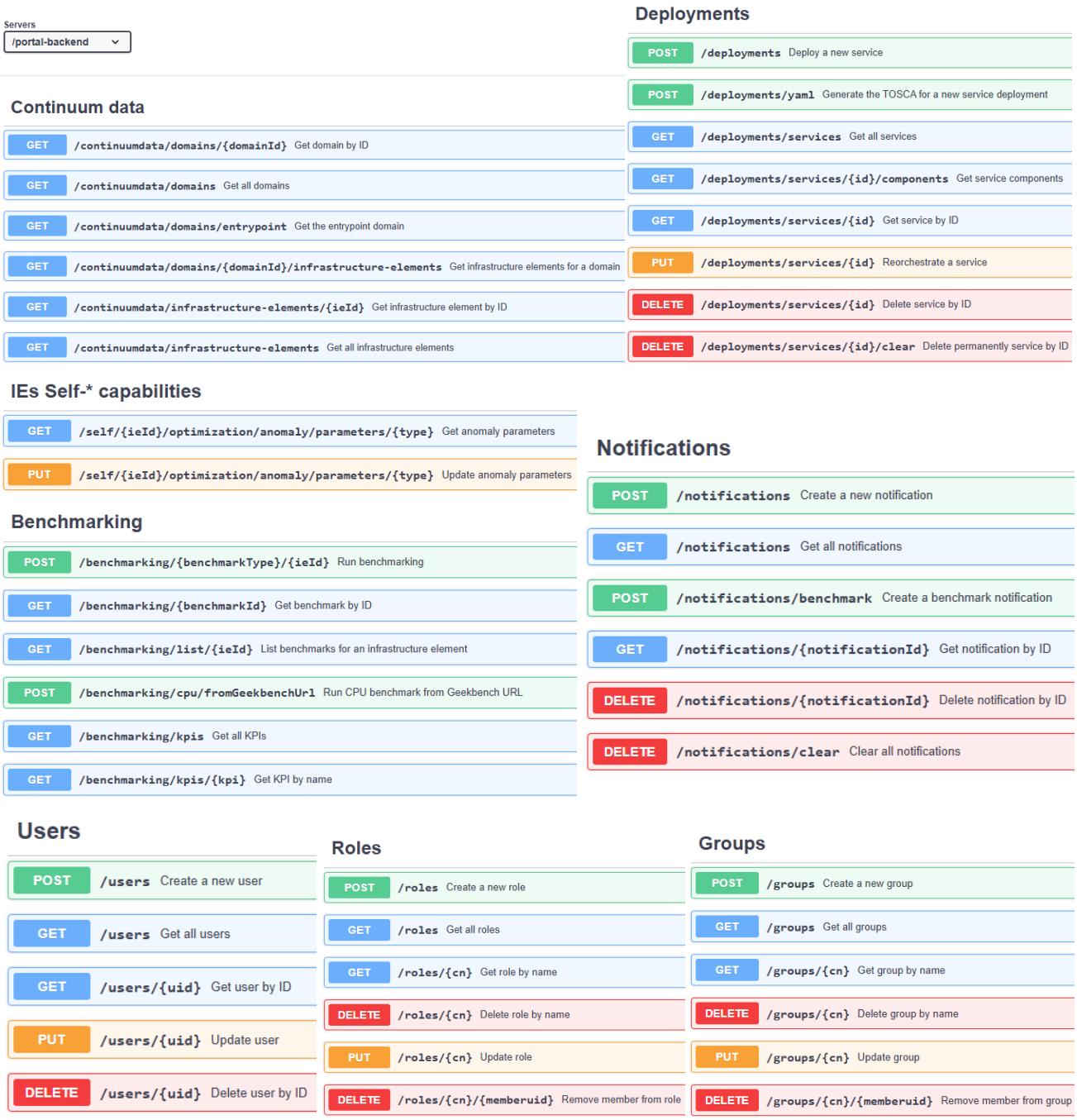


Figure 63. OpenAPI definition of the aerOS Management Portal Backend

3.6.1.3. Entrypoint balancer

Before proceeding to the analysis of existing load balancing (LB) approaches, it is necessary to outline the requirements that must be accomplished by the aerOS entrypoint balancer. First, in terms of decision-making, the LB algorithm should rely on the balancing rules and not operational requirements. It should forward the first received Intention Blueprint request and focus strictly on workload distribution since task re-transferring is part of the HLOs' responsibilities. Finally, the chosen algorithm should have a low level of overhead and possibly be simple to implement.

These conditions were considered in the selection of quality metrics used to evaluate existing LB algorithms. Among the metrics proposed in [M1], [M2], the ones that are the most relevant include scalability, degree of imbalance, and performance. In terms of areas of application, the focus has been placed on algorithms that aim

to maximize the throughput and avoid bottlenecks. The resource-utilization-tailored ones were of less importance since the entrypoint balancer does not process Intention Blueprints.

The literature proposes many taxonomies used to categorize LB algorithms. The most common one, introduced among others in [M3], [M4] divides them into static, dynamic, meta-heuristic, and ML-centric ones. Here, it should be pointed out that meta-heuristic and ML-centric algorithms (e.g., Genetic Programming based Load Balancing (GPLB) [M5]) have a high level of complexity and are considered mostly in case of large solution spaces. Therefore, regarding the specifications for the entrypoint balancer, they are architecturally overcomplicated, and, as such, they were not considered in further analysis.

Static LB algorithms use predetermined knowledge and assumptions about resources. The most widely used algorithms within this group include different versions of Round Robin (RR) [M6], [M7], which assigns tasks according to a circular list, or Opportunistic Load Balancing (OLB) [M8], which focuses strictly on keeping components busy while assigning workloads in arbitrary order. Since these algorithms do not adjust to the current state of the system, they have minimal overhead, however, at the same time, they are of low flexibility. As such, they may not be able to properly handle dynamic changes in the aerOS infrastructure such as re-transferring tasks between HLOs.

The better suited are the dynamic LB algorithms, which consider the current system state, by, for example, re-evaluating the load of its components. Consequently, algorithms from this group are more difficult to implement, but they also are a better fit for heterogeneous systems. The scope of dynamic LB comprises a variety of different algorithms. The simpler ones, in the majority, are the modifications of the Least Connections (LC) which redirects the requests to the infrastructure component that has the least number of active connections (e.g., Weighted Least Connections Round Robin -WLC RR [M9]). The more complex ones are, for example, Resource-based Load Balanced Min-Min (RLBMM) or LBMM, which take into account also task re-distribution and resource utilization. However, considering the restricted amount of information to be processed by the entry point balancer implementation of these more complex algorithms would not be possible without re-adjustments of current architectural concepts.

Therefore, based on the conducted research, it has been determined that the most suitable LB algorithm in the case of the entrypoint balancer is the LC (or one of its modifications). The proposed algorithm meets all established criteria and yet is simple enough that it doesn't require major changes in existing architecture.

The algorithm implemented within Entrypoint balancer is based on the Improved Weighted Least Connections. It works in the following way. The Entrypoint balancer first retrieves all the Domain entities from the aerOS distributed state repository and then assigns to each of them the flag availability, which indicates whether a given domain is going to be considered in the processing of next client request. Initially, all domains are marked as available. The information about the domains' availability is stored in the Entrypoint balancer's cache and is being updated in consecutive computations of the algorithm.

Whenever the client request is received, the Entrypoint balancer begins by evaluating the scores of each available domain using the weighting function. By default, the weighting function computes the CPU usage of all IEs belonging to the given domain. However, its code can be easily modified (e.g. to take into account different properties such as RAM usage) depending on the high-level requirements. The domain's score is obtained by dividing the number of services running within a domain (services with Running status) by the weight computed using weighting function. Then the domain, which has a highest score is selected by the algorithm.

To prevent the cases in which a single domain would be consecutively selected by the algorithm (e.g. when it was newly added to the continuum), the Entrypoint balancer uses a parameter maxAssignments. If a given domain is selected more than maxAssignments time, it is being temporarily blocked for the next *maxAssignments - 1* selections [M10]. After that, the domain is made, once again, available. Obviously, this feature is applicable if there is at least one additional domain to the entrypoint in the continuum.

3.6.1.4. Benchmarking tool

The benchmarking tool provides two main functionalities within the management portal:

1) IE performance in terms of CPU/RAM processing as well as network interfaces.
2) Dashboard of those technical KPIs that are directly measurable in an aerOS continuum.

The first functionality of the benchmarking tool is in turn split into two services. On the one hand, the IE processing performance service is carried out by an agent that is installed on every IE of the aerOS continuum. In particular, the Geekbench open-source agent is used, which, by executing a series of standardized tests measure both CPU and GPU capabilities that put the hardware through rigorous tasks, mimicking real-world applications to generate scores. These scores cover multiple aspects, including single-core and multi-core performance. Thus, the benchmarking tool helps aerOS practitioners to gauge how well an IE performs under different workloads. The generated results of Geekbench are stored in the Orion-LD context broker as new entities of type "cpuBenchmark", following NGSI-LD data model, so that each benchmark result contains relevant attributes reflecting different aspects of the performance, such as `singleCoreScore`, `multiCoreScore`, `deviceName`, and `timestamp`.

On the other hand, the networking service is designed to measure the bandwidth performance of network connections of any IE. For that purpose, iPerf3 open-source tool is implemented. iPerf3 is a command-line tool that allows users to test the throughput of their network by generating TCP and UDP traffic, and measuring how much data can be transferred over a network within a specified time frame, offering insights into both the speed and quality of their network connection. During the test, iPerf3 generates network traffic from the client to the server and measures the rate at which data is transferred, providing reports on bandwidth, packet loss (for UDP), jitter, etc. Likewise, the performance benchmarking service, the results of the networking tests are stored as dedicated "networkBenchmark" entities. Finally, both types of entities are retrievable by the management portal for visualization purposes in the form illustrated in Figure 63. The overall benchmarking tool system can also be observed in the next figure:

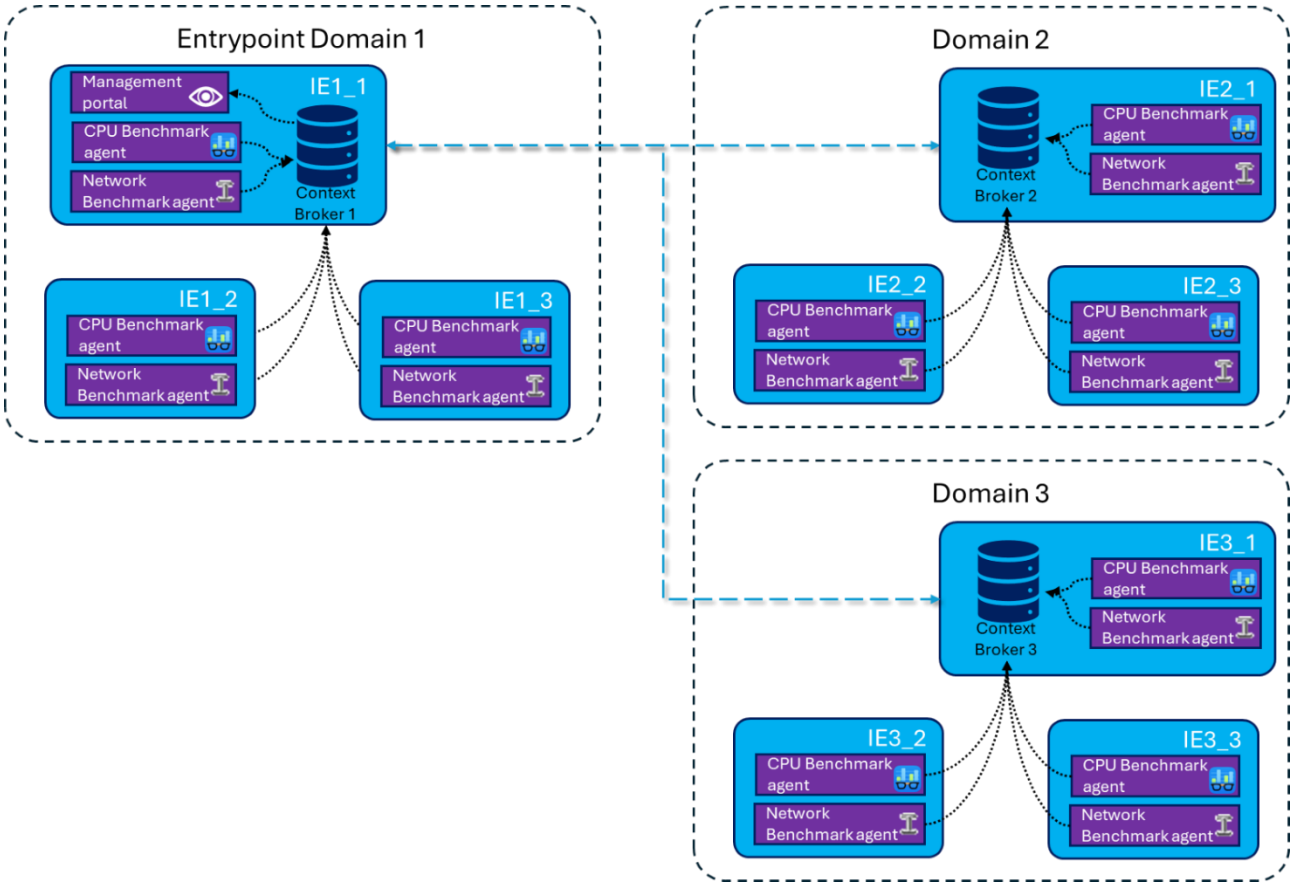


Figure 64. Benchmarking tool architectural diagram.

For the second functionality of the benchmarking tool, the technical KPIs dashboard, the following values are displayed. These are the values of technical KPIs that are directly measurable from the data provided by the aerOS distributed state repository:

- # of federated aerOS domains
- # of IEs that belong to the domains
- # of container management frameworks used by IEs and % of usage
- # of CPU architectures used by IEs and % of usage
- # of deployed user services and # of their underlying components
- # of authenticated users

3.6.1.5. Technologies and standards

Table 21. Technologies and standards for aerOS Management Portal.

Technology/ Standard	Description	Justification
Vue.js	Vue.js is an open-source JavaScript framework for building web user interfaces.	Vue.js is currently one of the most popular frameworks to build web applications along with React, so it was decided to take advantage of previous experience in the technology by the partners involved in the task.
Pinia	Pinia is a state store library for Vue.js	State management plays an important role in web applications because allows to store the current state of the application, which depends on the actions previously performed by users.
Keycloak.js	A Keycloak client library in JavaScript to interact with Keycloak	This library is used to interact with Keycloak to perform the user login process which is based on the Authorization Code Flow with Proof Key for Code Exchange (PKCE) of the OAuth2.0 protocol. It also allows to retrieve key data from the logged user such as its name, organization and role.
v-network-graph	Vue3 library for creating interactive network graphs.	The Management Portal is intended to show the current state of the computing continuum in a visual friendly way, so this library has been selected to show this information following a network graph visualization.
nginx	nginx is a modern web server and proxy	nginx has been selected for the portal packaging to act as (i) a web server to serve the frontend web application and (ii) a reverse proxy to enable the communication among frontend and backend
Spring Framework	One of the most popular Java frameworks to develop microservices, completely oriented to build cloud-native microservices.	Partners involved in the task have expertise in the technology and it has resulted a good option to develop backends of dashboard built as web applications.
Spring Security	Spring Security is the de-facto standard for securing Spring applications.	Requests sent by the frontend must be authenticated with Keycloak tokens, so the backend must check the validity and scope of these tokens through the interaction with Keycloak. It can be achieved by using the built-in Oauth2.0, OIDC and Keycloak support.

Spring Cloud	Spring Cloud provides tools for building applications in the scope of distributed systems.	The Backend will forward some requests to aerOS Basic Services that will not need any modifications, so it must be able to act as a reverse proxy (Spring Cloud Gateway). In addition, the Backend can leverage more capabilities provided by Spring Cloud libraries.
Socket.IO	Socket.IO is a library that enables low-latency, bidirectional and event-based communication between a client and a server.	This library has been selected to implement a real-time WebSocket-based communication between the frontend and backend to display notifications in the portal. It is used in both frontend (as the client) and backend (as the server) components.
Improved Weighted Least Connections load balancing algorithm	Dynamic load Balancing algorithms take into account the current state of the system by re-evaluating the load of its components.	Service orchestration requests from the portal's backend must be forwarded to HLO instances of different domains following a distributed approach. After a research, dynamic least connection algorithms (or its modifications) are the best fitting into the aerOS distributed architecture.
Geekbench	Geekbench is a popular open-source benchmarking tool used to assess the performance of various devices, including smartphones, tablets, and computers, by executing a series of tests that measure both CPU and GPU capabilities.	Geekbench agent is installed on every IE of the aerOS continuum, and by executing a series of standardized tests (mimicking real-world applications) measure both CPU and GPU capabilities that helps aerOS practitioners to gauge how well an IE performs under different workloads.
iPerf3	iPerf3 is a widely used networking benchmarking tool designed to measure the bandwidth performance of network connections.	iPerf3 provides detailed metrics on throughput, latency, jitter, and packet loss, which are very helpful for identifying network bottlenecks, validating optimizations, and ensuring network performance meets required standards in aerOS federated IEs and domains.

3.6.2. aerOS Federator

The aerOS Federator serves as a management service responsible for controlling the establishment and maintenance of federation mechanisms among the multiple aerOS domains that form the continuum. According to its block architecture, it is composed by two main components: (i) custom aerOS Federator component and (ii) Orion-LD context broker. The core federation functionalities are provided by the context broker through the establishment of Context Source Registrations (CSR), which allows an Orion-LD instance to retrieve information (in NGSI-LD entities format) from another Orion-LD instances, which in the aerOS continuum means that data from one domain can be obtained just by calling the context broker of another domain once a proper registration has been performed. In addition, Orion-LD also allows to perform its publication and subscription mechanism in a federated way, a novelty that wasn't ready for the first MVP of the project. These federation capabilities are directly linked with the aerOS Data Fabric described in section 3.2, but the aerOS Federator component has been designed to act as the starting point of this mechanism (domain discovery) and is mainly focused on the data related with the management of the continuum, which follows the ontology introduced in section 3.1.3.2. It also acts as the building block for the aerOS distributed domain repository and allows to discover and retrieve the data products (see section 3.2) created in each domain in a federated way.

As reported in the previous D4.2, the main efforts for the first MVP of the project were put in testing Orion-LD federation mechanisms to deliver a methodology to create the needed Context Source Registrations to achieve domains federation, always having into consideration the aerOS ontology for Cloud-Edge-IoT computing continuum. This work allowed to create a solid base to develop the custom aerOS Federator component on top of it, which is the main novelty of this block of the task for the reported period.

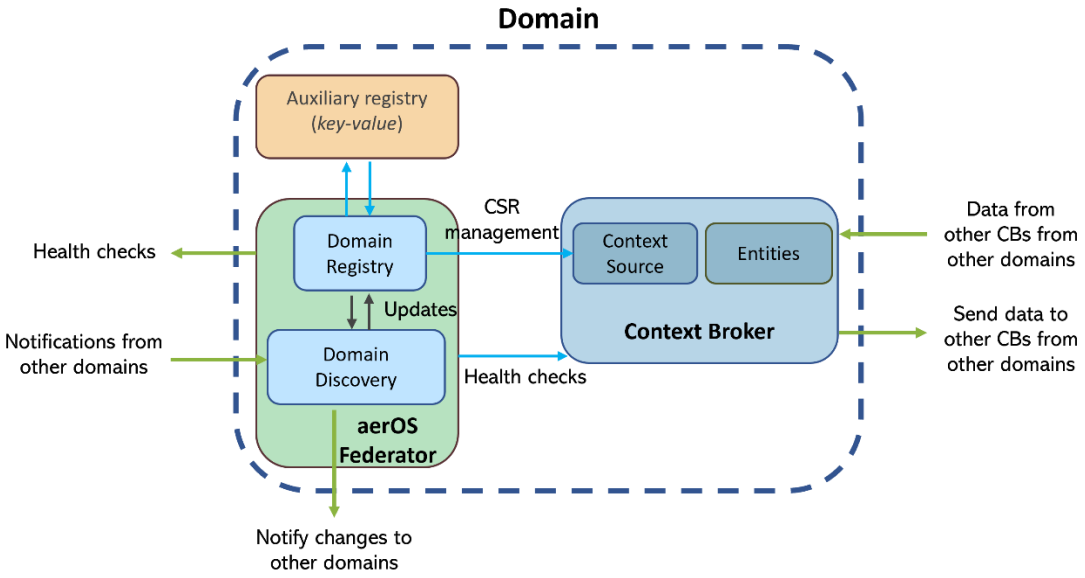


Figure 65. aerOS Federator architecture in a single domain.

3.6.2.1. Enhanced capabilities of Orion-LD context broker

It is important to highlight that Orion-LD is not only an open-source implementation of an NGSI-LD context broker that has been taken from its repository, deployed, and used in the aerOS project, but also is constantly being improved within the scope of aerOS. In addition, the main development team of Orion-LD, which is the FIWARE Foundation, is an active partner of the aerOS project, so functionalities needed for the aerOS Federation are directly added to the context broker. These improvements and solved challenges will be described below with more details. Furthermore, other teams working on the task are responsible for testing (through the creation and execution of ad-hoc functional tests) these enhancements along with previously developed functionalities to improve the quality of the broker as it is a core component in the aerOS architecture.

Regarding the code development of Orion-LD in the scope of aerOS, queries for retrieving contextual data as NGSI-LD entities actually stored in different context brokers play a main role in the aerOS federation. Thus, in the domain of distributed systems, particularly within the context of entity federation, the pagination of entity query results stands as a significant technical challenge. This challenge escalates when the queries involve dynamic attribute values, at which point the task transitions from merely difficult to seemingly unfeasible. For effective pagination, it is imperative to establish a consistent set of resulting NGSI-LD entities across different paginated queries. Otherwise, without this consistency, the realization of pagination is not viable. These challenges have been extensively deliberated within the ETSI ISG CIM, in the course of standardizing the NGSI-LD API. Consequently, two distinct solutions have been formulated:

- Entity Maps: this solution involves freezing the set of resulting entities as a list that only includes their IDs.
- Snapshots: this approach goes a step further by freezing the entities themselves, storing them within a local database.

After some technical discussions, the Entity Maps strategy was first adopted by the aerOS project in order to reduce the amount of exchanged data among the brokers, aligning with its implementation in Orion-LD for the MVP. After being successfully tested in the project, this approach was incorporated into Orion-LD version 1.5.1

(its current version is the 1.9.0) and then was slated for inclusion in the v1.8.1 of the NGSI-LD API specification. Finally, it was included in this version of the specification, which was delivered in March 2024. The Snapshots method remains under discussion in ETSI ISG CIM and is anticipated to be part of NGSI-LD API version 1.10.1, targeted for summer-autumn 2025.

Going into more technical details, the process begins with an initial query to Orion-LD API to obtain a set of NGSI-LD entities (GET */ngsi-ld/v1/entities?<queryParameters>)* during which the Entity Map is generated and stored in the broker. The response includes the Entity Map's ID in an HTTP header (NGSILD-EntityMap), along with the first batch of entities in the payload body. Subsequent paginated requests must provide this Entity Map ID, along with offset/limit URL parameters for pagination.

An Entity Map is structured as an array mapping entity IDs to arrays of Context Source Registration IDs, indicating the registrations associated with each entity. In the aerOS project, where attributes of entities are not distributed across multiple brokers, the registration ID arrays typically contain a single entry. Furthermore, for entities hosted locally in the broker, a special identifier (@none) is used. Finally, armed with the information of the Entity Map, the broker is fully informed to dispatch distributed requests and compile the responses into an array of entities. Upon processing all responses, the broker then furnishes this array to the original requester.

```
{
  "urn:ngsi-ld:entities:E1": [ "urn:ngsi-ld:registrations:R1" ],
  "urn:ngsi-ld:entities:E2": [ "urn:ngsi-ld:registrations:R6" ],
  "urn:ngsi-ld:entities:E3": [ "urn:ngsi-ld:registrations:R0" ],
  "urn:ngsi-ld:entities:E4": [ "@none" ],
  "urn:ngsi-ld:entities:En": [ "urn:ngsi-ld:registrations:Rn" ]
}
```

Figure 66. Orion-LD Entity Map example.

In addition, a first version of the distributed operations for subscriptions over contextual has been implemented, as it was a requested feature for the aerOS Meta-OS in order to enhance the federation among domains. This feature, which is also based on Context Source Registrations, enables the creation of subscriptions in one Orion-LD that target entities that are stored in other Orion-LD instances. In the scope of aerOS, it allows to subscribe to contextual data that is present in other domains to be notified in the original domain when this data changes. For instance, some key attributes of all Service Components or Infrastructure Elements of the continuum can be monitored from a module deployed in a single domain.

When a subscription is created in an Orion-LD instance, it checks its CSRs to select the ones which matches with the target entities of the subscription. Then, for each matched CSR, it creates in the target Orion-LD broker of the CSR (the broker specified in the *endpoint* attribute) a subordinated subscription. This subordinated subscription points to a special notification endpoint from the original Orion-LD in which was created: */ex/v1/notify/subscriptions/{subscriptionId}* in order to notify the local changes to the original Orion-LD instance. When the original subscription is deleted, the subordinated subscriptions are sequentially deleted.

3.6.2.2. aerOS Federator custom component

The aerOS Federator custom component is intended to provide another layer of automatization above Orion-LD to avoid direct interaction with the context brokers of the continuum when it comes to federation management, as well as federated backup mechanisms for federation critical data (e.g. domains registry).

In the design phase of this component, gRPC was taken into consideration to develop the API of this component as it provides some advantages over traditional REST such as bidirectional communications and the reduction of payload size. However, the aerOS Federator must interact constantly with the REST API of Orion-LD, so the improvements provided by gRPC would be limited at the end. In addition, the use of a REST API enables a better integration with the aerOS domain API Gateway, so that the Federator API endpoints can be seamlessly included in this API Gateway to improve the connectivity between Federator instances of all the domains. Although there are some tools such as gRPC-Gateway that translates REST request (using JSON in their payloads) into gRPC requests (using binary protobuf in their payloads), the inclusion of more reverse proxies

would also add more latency into the system, which is a matter of concern in distributed ecosystems such as the aerOS continuum.

The aerOS Federator has been developed using Go because this language focuses on the fast implementation of functionalities while provides high performance for microservices and cloud-native applications. In addition, Go provides by default a fast and easy compilation mechanism to target a wide range of CPU architectures, so it facilitates the build of multi-architecture container images to spread the federation capabilities of aerOS in several kinds of IEs and domains.

This component is mainly focused on the management of the Context Source Registrations in the context broker of its domain, because they are the key elements that enable the federation of NGSI-LD context brokers. Therefore, five types of Context Source Registrations are automatically created and then managed by the aerOS Federator. Each type targets a different set of entity types with different permissions over those entities:

- **AAA:** it allows to retrieve entities related with the aerOS AAA (types: *User*, *Role* and *Organization*).
- **Infrastructure**: entities of types - *Domain*, *InfrastructureElement* and *LowLevelOrchestrator*. These entities can only be retrieved in a federated way (*retrieveOps*) as they reflect the current status of the infrastructure of the domains, so they can only be modified locally in the CB of the domain which belongs to.
- **Services**: entities of types - *Service*, *ServiceComponent*, *InfrastructureElementRequirements*, and *NetworkPorts*. These types of entities can be retrieved, updated and deleted in a federated way (*retrieveOps*, *updateOps* and *deleteEntity*) to enable the aerOS decentralized orchestration process.
- **DataFabric**: entities of types *DataProduct*, *Concept* and *ContextBroker* to list the data products available in all the domains.
- **Benchmark**: it allows to retrieve the results of the Benchmark executions, so it only applies to entities of type *Benchmark* with *retrieveOps* operations.


```
{
  "id": "urn:aeros:federation:ncsrd:services",
  "type": "ContextSourceRegistration",
  "information": [
    {
      "entities": [
        {
          "type": "Service"
        },
        {
          "type": "ServiceComponent"
        },
        {
          "type": "NetworkPort"
        },
        {
          "type": "InfrastructureElementRequirements"
        }
      ]
    }
  ],
  "contextSourceInfo": [
    {
      "key": "Authorization",
      "value": "urn:ngsi-ld:request"
    }
  ],
  "mode": "inclusive",
  "operations": [
    "retrieveOps",
    "updateOps",
    "deleteEntity"
  ],
  "hostAlias": "Domain-NCSRD",
  "management": {
    "localOnly": true
  },
  "endpoint": "https://ncsrd-mvp-domain.aeros-project.eu/orionld",
  "aerosDomain": "NCSRD",
  "aerosDomainFederation": true
}
```


Figure 67. Example of a Context Source Registration created by the aerOS Federator.

Each instance of the aerOS Federator must be assigned with another Federator instance, which is named as peer federator, to be able to spread the addition of its own domain and to stablish a starting point for additional spreading processes.

This component provides a complete REST API to expose its capabilities, which can be split into two groups: (i) initiation of a federation process (spread its own domain creation or deletion); and (ii) reaction to a in the continuum (e.g. new domain notification, spread a new domain when acting as a peer federator, deletion of a domain…).

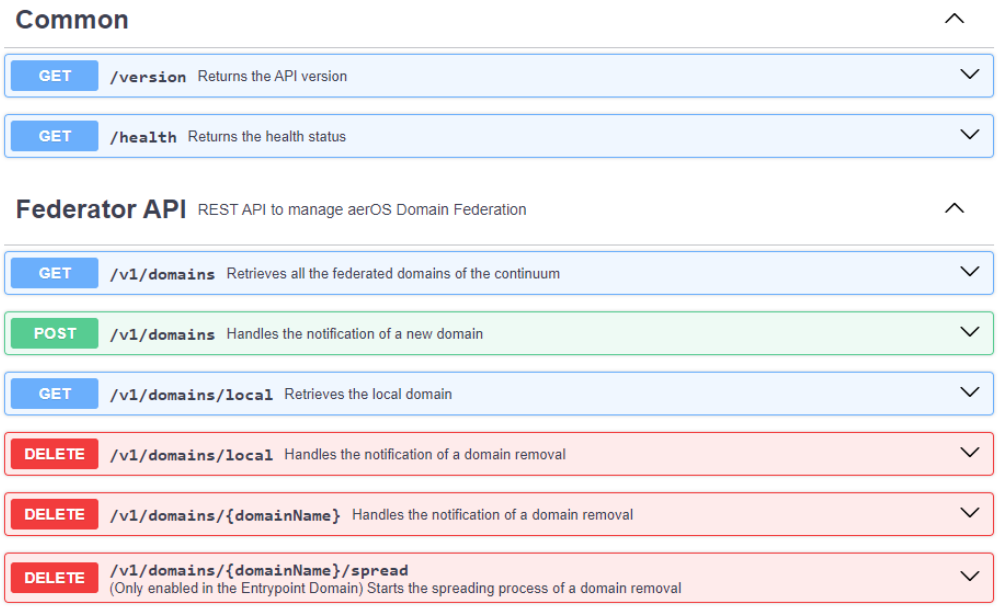


Figure 68. Endpoints of the aerOS Federator API.

Finally, in the next figure it is displayed a complete sequence diagram (with several technical details) that depicts the whole process when an instance of the aerOS Federator is deployed in a new domain in order to be added to the continuum. In this diagram, it is shown the interaction with the local CB, the peer federator and the process carried out by it to spread the new domain addition across the continuum. Moreover, it also acts as a matter of proof of the technical difficulties that involves the development of this tool, which is actually one of the key components to stablish the federation between domains in the continuum.

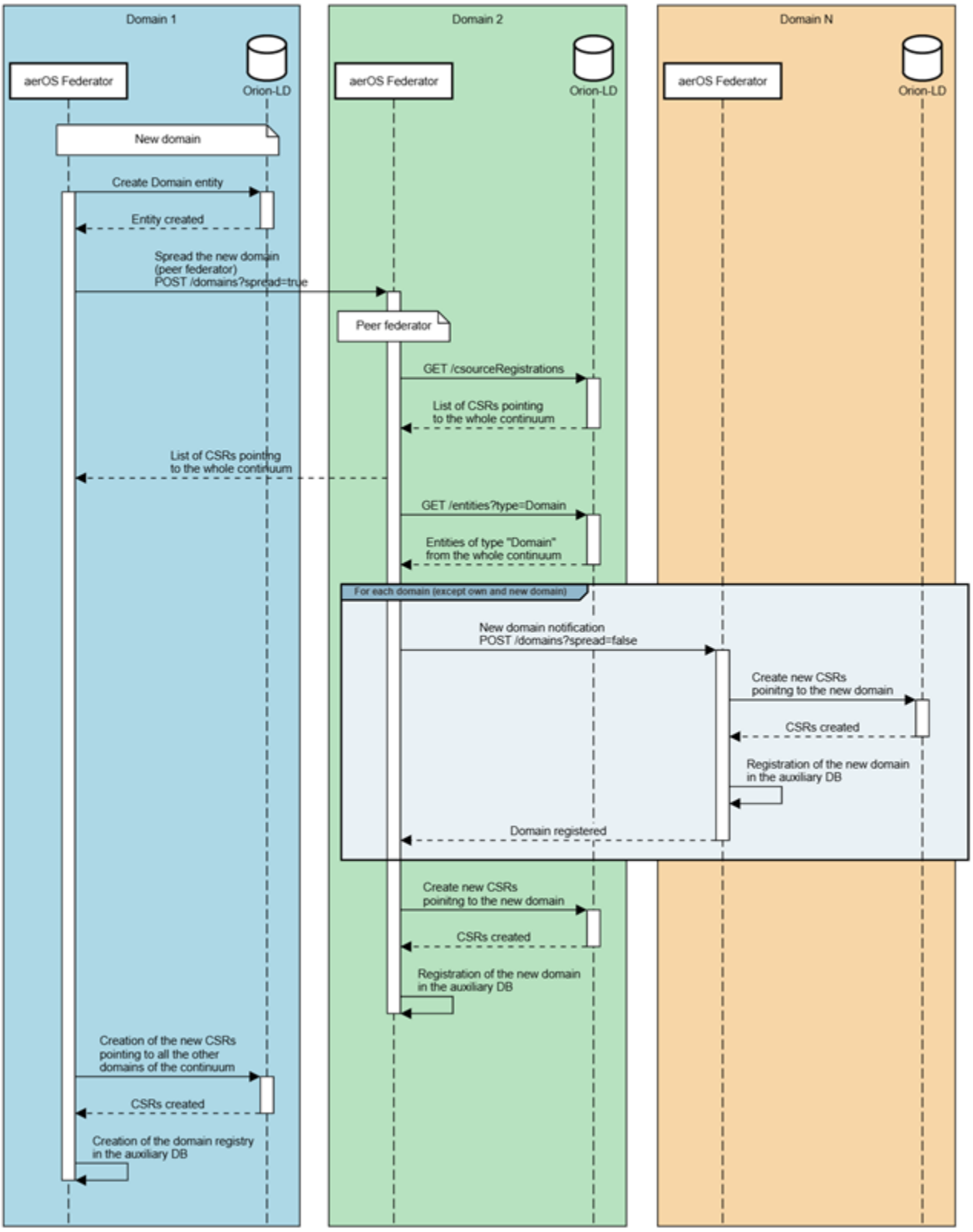


Figure 69. Sequence diagram for the process of adding a new domain to the continuum.

3.6.2.3. Technologies and standards

Table 22. Technologies and standards for aerOS Federator.

Technology/ Standard	Description	Justification
Orion-LD	Open-source implementation of the NGSI-LD context broker.	The main developer team of Orion-LD is partner of aerOS (FIWARE Foundation), so custom functionalities have been developed to improve its fitting in the aerOS domain federation. In addition, some partners have previous expertise on the usage of this context broker, so it allowed to perform the testing of these new functionalities in a more agile way.
Microservices architecture	A microservices based architecture is a software development and deployment approach where an application is built as a collection of small, loosely coupled and independently deployable services.	The aerOS Federator provides a layer of automatization and fault tolerance mechanisms on top of Orion-LD. By using a microservices architecture, the development of this component has been more agile, by allowing each development team to develop a specific feature in the most appropriate programming language. In addition, some capabilities would not be required in each domain or at every point in time.
gRPC	Open-source implementation of remote procedure calls (RPC) developed by Google, which uses the HTTP/2 protocol for communications and Protocol Buffers as its messages format.	aerOS Federator instances are expected to exchange a huge number of messages and react to some key events in near real-time due to changes in the continuum (e.g., the failure of the entrypoint domain), so it must be built on top of modern and agile communication protocols and technologies such as HTTP/2 and gRPC.
Key-value store	A key-value store (or key-value database) is a type of non-relational database that stores data as a collection of key-value pairs. It is one of the simplest types of databases, optimized for fast retrieval and storage of data.	The aerOS Federator can use a key-value store or database to store locally some key information of the continuum (e.g. key information of domains) that can be useful for recovery in case of failures or service interruptions.

4. Conclusions

This deliverable presents the final release of the WP4 software components for delivering intelligence at the edge. The document has additionally provided an overview of the second version of aerOS demonstrator MVPv2 from the standpoint of WP4, detailing how each of the building blocks has contributed to the realization of features of the aerOS MVPv2 such as the intelligent orchestration of the continuum or data sharing with Data Fabric.

Regarding data homogenization, the Semantic Annotator and Semantic Translator components have been integrated into the aerOS Data Fabric. The Linked Open Terms (LOT) methodology for ontology development was used in the project and applied during the creation of the aerOS continuum ontology and aerOS data catalog ontology. This methodology will be further explored and followed for developing ontologies in the use cases identified within aerOS.

Research activities around data governance have consolidated the definition of a data product in aerOS. This definition has derived into a stable architecture of the aerOS Data Fabric, introducing Data Product Pipeline as a framework to facilitate the creation of data products, along with the Data Product Manager to interface with data product owners and orchestrate these pipelines. The aerOS Data Fabric was additionally extended data security and data catalog features.

When it comes to decentralized frugal AI, the AI Local Executor and AI Task Controller components have been prepared for AI workflow execution (specifically federated learning). The AI Local Executor service is deployed using aerOS service deployment mechanism allowing for setting restrictions on the needed Infrastructure Elements. In addition, an explainability for an aerOS use case based on reinforcement learning has been proposed, which can be replicated and adjusted for other use cases of providing explainability in the aerOS environments. Different frugality techniques were evaluated for their applicability to be used on a dataset typical for edge-based computing use cases and conclusions were drawn.

An Embedded Analytics Engine has been implemented as a platform for the creation, distribution and deployment of analytical functions to provide insights and advanced decision making to the aerOS Meta Operating System. It includes template capabilities to help in the creation of user-defined functions, as well as a set of pre-packaged functions based on popular data science libraries.

Trustworthiness and secure communication between Infrastructure Elements are key features of the aerOS deployments. Here, two components were developed: the aerOS Trust Manager, responsible for the dynamic assessments of the IEs, and the IOTA, ensuring decentralization, data integrity, scalability, and efficiency in data exchange.

The crucial component for aerOS is the Management Portal and management services that provide an entrypoint to the system, as well as mechanisms to manage the environment. aerOS Management Portal allows for interaction with Basic Services, support real-time notifications, AAA management, Data Fabric's data products management and includes benchmarking tool. Moreover, the custom aerOS Federator component has been implemented to automate and coordinate the federation process among the multiple aerOS domains that build the Cloud-Edge-IoT continuum.

References

[D1] M. Poveda-Villalón, A. Fernández-Izquierdo, M. Fernández-López, and R. García-Castro, 'LOT: An industrial oriented ontology engineering framework', *Eng. Appl. Artif. Intell.*, vol. 111, p. 104755, May 2022, doi: 10.1016/j.engappai.2022.104755.

[D2] R. García-Castro, M. Lefrançois, M. Poveda-Villalón, and L. Daniele, 'The ETSI SAREF ontology for smart applications: a long path of development and evolution', in *Energy smart appliances: Applications, methodologies, and challenges*, 2023, pp. 183–215. doi: 10.1002/9781119899457.ch7.

[D3] M. C. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López, 'The NeOn Methodology for Ontology Engineering', in *Ontology Engineering in a Networked World*, M. C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta, and A. Gangemi, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–34. doi: 10.1007/978-3-642-24794-1_2.

[D4] 'BIMERR Ontologies'. Accessed: Feb. 18, 2024. [Online]. Available: https://bimerr.iot.linkeddata.es/

[D5] D. Garijo and M. Poveda-Villalón, 'Best Practices for Implementing FAIR Vocabularies and Ontologies on the Web', 2020, doi: 10.48550/ARXIV.2003.13084.

[D6] P.-Y. Vandenbussche, G. A. Atemezing, M. Poveda-Villalón, and B. Vatant, 'Linked Open Vocabularies (LOV): A gateway to reusable semantic vocabularies on the Web', *Semantic Web*, vol. 8, no. 3, pp. 437–452, Dec. 2016, doi: 10.3233/SW-160213.

[D7] D. Garijo, 'WIDOCO: A Wizard for Documenting Ontologies', in *The Semantic Web – ISWC 2017*, vol. 10588, C. d'Amato, M. Fernandez, V. Tamma, F. Lecue, P. Cudré-Mauroux, J. Sequeda, C. Lange, and J. Heflin, Eds., in Lecture Notes in Computer Science, vol. 10588. , Cham: Springer International Publishing, 2017, pp. 94–102. doi: 10.1007/978-3-319-68204-4_9.

[D8] 'FOAF Vocabulary Specification'. Accessed: Feb. 18, 2024. [Online]. Available: http://xmlns.com/foaf/spec/

[D9] A. G. Beltran, R. Albertoni, D. Browning, S. Cox, A. Perego, and P. Winstanley, 'Data catalog vocabulary (DCAT) - version 3', W3C, W3C Proposed Reccommendation, Jun. 2024.

[M1] E. Jafarnejad Ghomi, A. Masoud Rahmani, and N. Nasih Qader, 'Load-balancing algorithms in cloud computing: A survey', *J. Netw. Comput. Appl.*, vol. 88, pp. 50–71, Jun. 2017, doi: 10.1016/j.jnca.2017.04.007.

[M2] S. S. Tripathy *et al.*, 'State-of-the-Art Load Balancing Algorithms for Mist-Fog-Cloud Assisted Paradigm: A Review and Future Directions', *Arch. Comput. Methods Eng.*, vol. 30, no. 4, pp. 2725–2760, May 2023, doi: 10.1007/s11831-023-09885-1.

[M3] M. Hamdan *et al.*, 'A comprehensive survey of load balancing techniques in software-defined network', *J. Netw. Comput. Appl.*, vol. 174, p. 102856, Jan. 2021, doi: 10.1016/j.jnca.2020.102856.

[M4] I. N. Ivanisenko and T. A. Radivilova, 'Survey of major load balancing algorithms in distributed system', in *2015 Information Technologies in Innovation Business Conference (ITIB)*, Kharkiv, Ukraine: IEEE, Oct. 2015, pp. 89–92. doi: 10.1109/ITIB.2015.7355061.

[M5] S. Jamali, A. Badirzadeh, and M. S. Siapoush, 'On the use of the genetic programming for balanced load distribution in software-defined networks', *Digit. Commun. Netw.*, vol. 5, no. 4, pp. 288–296, Nov. 2019, doi: 10.1016/j.dcan.2019.10.002.

[M6] T. Hidayat, Y. Azzery, and R. Mahardiko, 'Load Balancing Network by using Round Robin Algorithm: A Systematic Literature Review', *J. Online Inform.*, vol. 4, no. 2, p. 85, Feb. 2020, doi: 10.15575/join.v4i2.446.

[M7] S. B. Vyakaranal and J. G. Naragund, 'Weighted Round-Robin Load Balancing Algorithm for Software-Defined Network', in *Emerging Research in Electronics, Computer Science and Technology*, vol. 545, V. Sridhar, M. C. Padma, and K. A. R. Rao, Eds., in Lecture Notes in Electrical Engineering, vol. 545. , Singapore: Springer Singapore, 2019, pp. 375–387. doi: 10.1007/978-981-13-5802-9_35.

[M8] T. D. Braun *et al.*, 'A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems', *J. Parallel Distrib. Comput.*, vol. 61, no. 6, pp. 810–837, Jun. 2001, doi: 10.1006/jpdc.2000.1714.

[M9] G. Singh and K. Kaur, 'An Improved Weighted Least Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems', vol. 05, no. 03.

[M10] Choi, D., Chung, K.S., Shon, J. (2010). An Improvement on the Weighted Least-Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems. In: Kim, Th., Yau, S.S., Gervasi, O., Kang, BH.,

Stoica, A., Ślęzak, D. (eds) Grid and Distributed Computing, Control and Automation. GDC CA 2010 2010. Communications in Computer and Information Science, vol 121. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17625-8_13

[A1] C. Wang et al., "Dependency-Aware Microservice Deployment for Edge Computing: A Deep Reinforcement Learning Approach With Network Representation," in IEEE Transactions on Mobile Computing, vol. 23, no. 12, pp. 14737-14753, Dec. 2024, doi: 10.1109/TMC.2024.3453069

[A2] W. Feng et al., "Exploring Collaborative Diffusion Model Inferring for AIGC-enabled Edge Services," in IEEE Transactions on Cognitive Communications and Networking, doi: 10.1109/TCCN.2024.3519320

[A3] Y. Chen, H. Yu, Q. Guo, S. Zhao and T. Taleb, "Dynamic Edge AI Service Management and Adaptation Via Off-Policy Meta-Reinforcement Learning and Digital Twin," ICC 2024 - IEEE International Conference on Communications, Denver, CO, USA, 2024, pp. 867-872, doi: 10.1109/ICC51166.2024.10622765

A. Supplementary research

A.1. Decentralized AI service deployment

Decentralized AI service deployment with network representation

The aerOS supports the deployment of distributed AI tasks across the continuum, considering heterogeneous and resource-constrained environments in terms of both network and computation. These distributed AI tasks may represent sub-tasks of a specific application, with specific logical dependencies among them. In such cases, it is crucial to account for these dependencies and the end-to-end QoS requirements of complex applications. However, factors such as dynamic system environments, the heterogeneity of computational capabilities, and the complexity of dependencies are not always thoroughly investigated, leading to issues such as excessive network resource consumption and inefficient decision-making in deployment strategies.

To achieve the ambition of aerOS, we developed a dependency-aware service deployment approaches to achieving efficient mapping between distributed AI tasks and physical edge continuum [A1]. The delivered approach is designed for the three-layer edge-cloud continuum aligned with aerOS, comprising the cloud layer, edge layer, and user equipment (UE) layer. An emerging AI application can be decomposed into a set of sub-tasks, where dependencies among these sub-tasks, represented as microservices, can be modeled as a directed acyclic graph (DAG). The proposed approach consists of two main components: the attention-based microservice representation (AMR) component and the attention-aided deployment strategy.

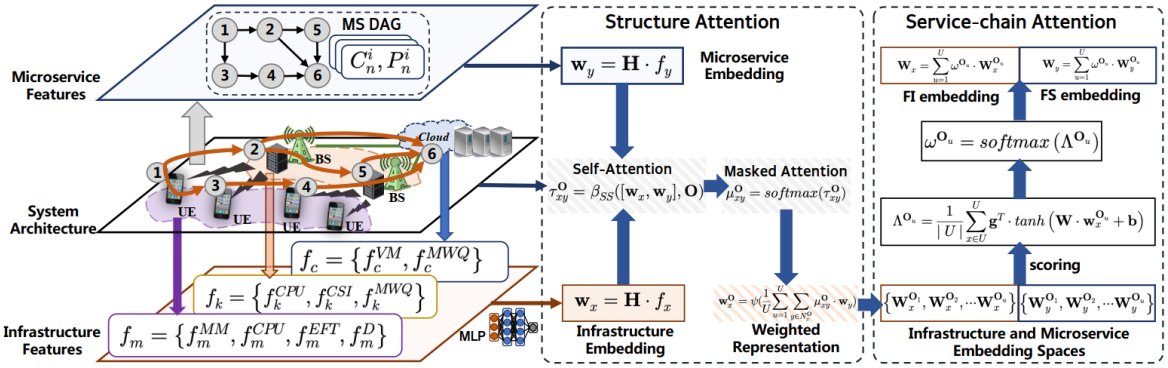


Figure 70. The attention-based distributed AI task representation [A1].

Figure 70 displays the attention-based distributed AI representation. The AMR algorithm is designed to extract system features and optimize the embedding representations of microservices and infrastructure using a multi-head attention mechanism. The process begins by extracting microservice and infrastructure features and mapping them into the structure attention space. Weighted representations are then leveraged to derive the service-chain attention space, ultimately producing the final embeddings for infrastructure and services. Initially, the algorithm takes as input the microservice DAG, infrastructure features, microservice features, and the service-chain set. For each service chain, it identifies neighboring microservices deployed on the infrastructure and computes their structure attention weights. These weights are then used to update the infrastructure representation, followed by normalizing the score function to derive the service-chain attention. Finally, the embeddings for microservices (FS) and infrastructure (FI) are generated.

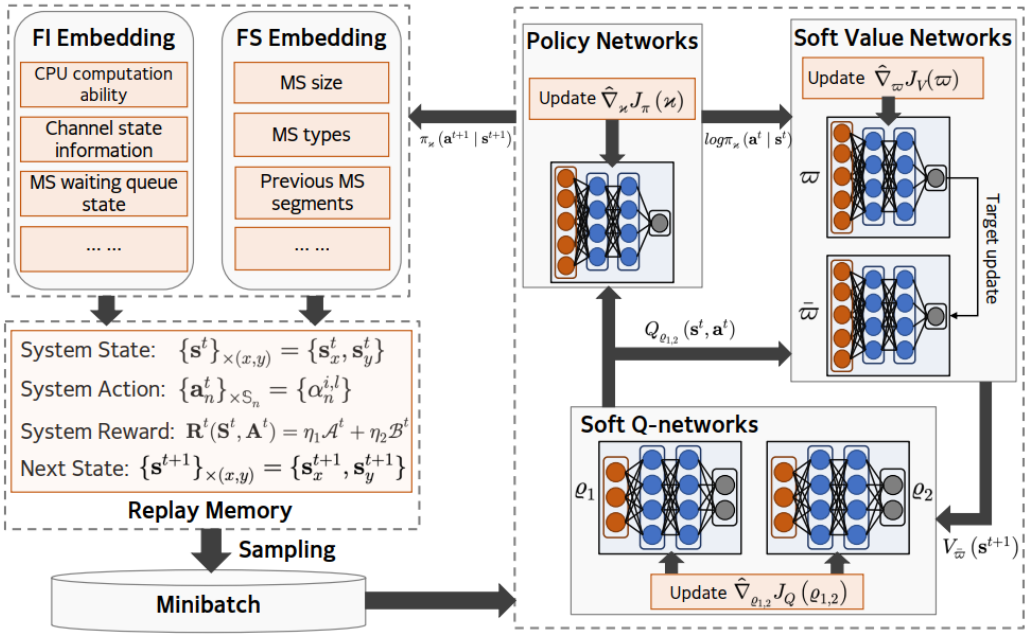
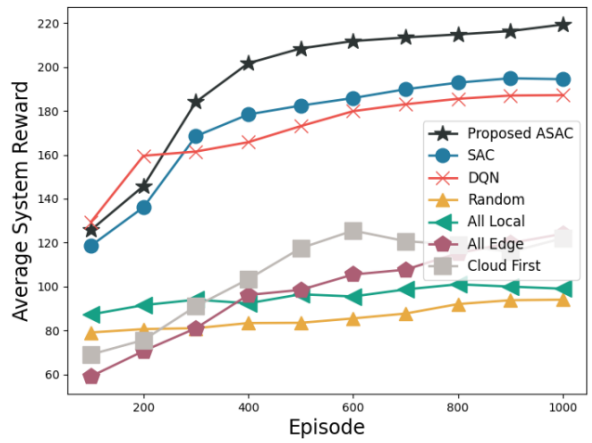
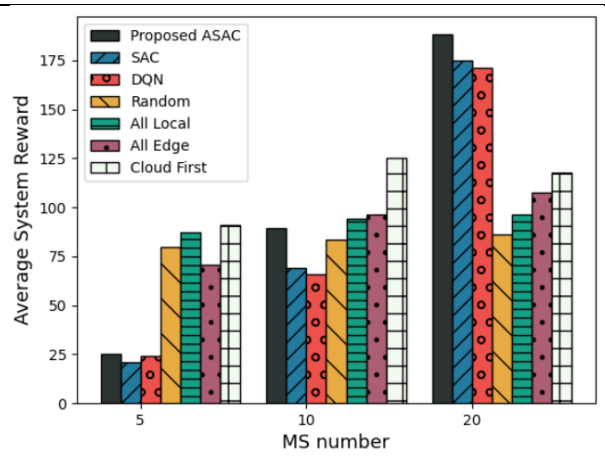


Figure 71. The attention-aided deployment strategy.

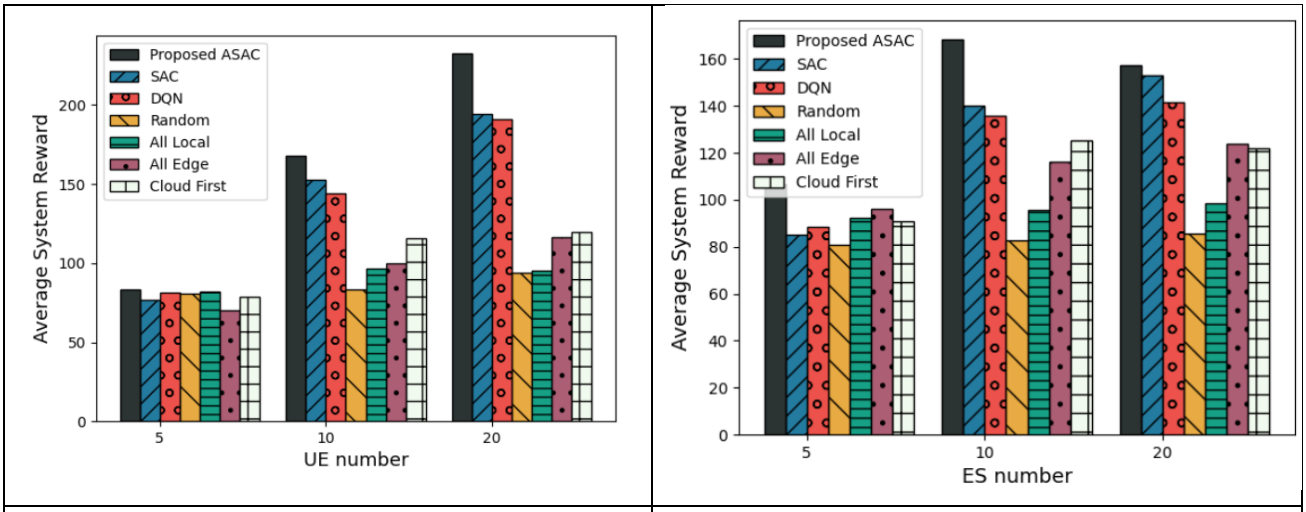
Figure 71 displays the attention-aided deployment strategy by considering the obtained representation context of FI and FS. The problem of distributed AI task deployment typically faces the challenge of large state and action spaces. To address this issue, we developed an attention-aided soft actor-critic (ASAC) method. The context of FS and FI, obtained from the AMR algorithm, is used as part of the state observation. The agent then processes this state observation to generate a service deployment decision. After executing the action, the current reward is computed based on data collected from applications and the continuum. Simultaneously, the system transitions to a new state based on its own evolution dynamics. The agent employs a soft actor-critic learning process to iteratively update its policy through interactions with the continuum until convergence is achieved [A1].



(a) Average system reward with different episodes



(b) Average system reward with different MS number.



(c) Average system reward with different UE number.

(d) Average system reward with different ES number

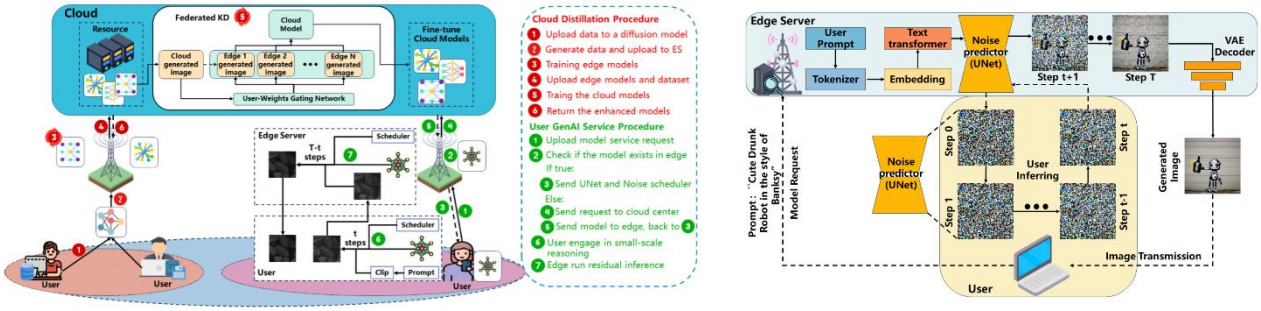
Figure 72. Comparisons of average system reward [A1].

We conducted simulations to evaluate our proposed approaches. Figure 72 depicts the evaluation results, comparing the performance of our ASAC algorithm with existing approaches. The experimental results demonstrate that ASAC significantly outperforms baseline schemes such as SAC, DQN, and Random allocation in terms of convergence, scalability, and adaptability across various system configurations. Specifically, ASAC achieves performance improvements of up to 15.8%, 22.2%, and 120% over SAC, DQN, and Random algorithms, respectively, by leveraging its advanced feature extraction capabilities, optimized resource allocation, and adaptive learning mechanisms. It excels in dynamic and large-scale environments, maintaining high performance as the number of UEs and MSs increases while effectively balancing latency, energy consumption, and computational efficiency. ASAC achieves peak reward performance at 10 ESs, though its efficiency declines when ESs increase to 20 due to higher computational demands. In contrast, SAC and DQN exhibit stable but lower performance, while the Random algorithm struggles under increasing system complexity. These results highlight ASAC's ability to intelligently manage resources, optimize system rewards, and adapt to dynamic conditions, making it a robust solution for complex network environments.

Exploring Collaborative Inferring for AIGC-enabled Edge Services

As aerOS advances the distributed deployment of AI-driven applications, edge collaborative inference for large-scale AI models emerges as a key use case, demanding strong edge system support. Traditional AI-based solutions struggle with the growing demand for high-quality content due to mobile devices' computational limitations. While AI-Generated Content (AIGC) provides innovative solutions, large models like Stable Diffusion require extensive resources, making mobile deployment impractical and leading to network congestion, excessive data traffic, and latency issues.

To address these challenges, aerOS explores the Edge-User Collaborative Inference (EUCI) framework, which partitions inference between edge servers and users. Figure 73 (a) illustrates the Edge-User Collaborative Diffusion-based AIGC Framework, designed to enable high-quality AIGC under constrained network conditions while enhancing resource efficiency and reducing cloud server load.

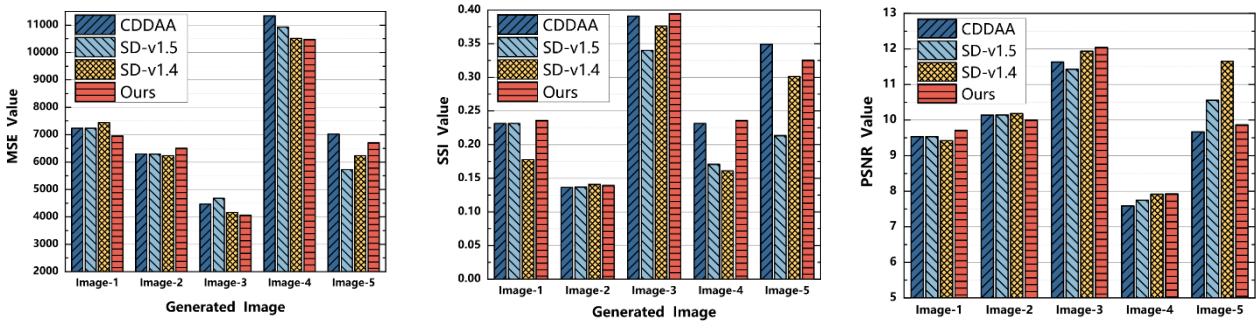


(a) (b)

Figure 73. Illustration of (a) edge-user collaborative diffusion-based AIGC framework, and (b) workflow of edge-user diffusion model collaborative inferring [A2].

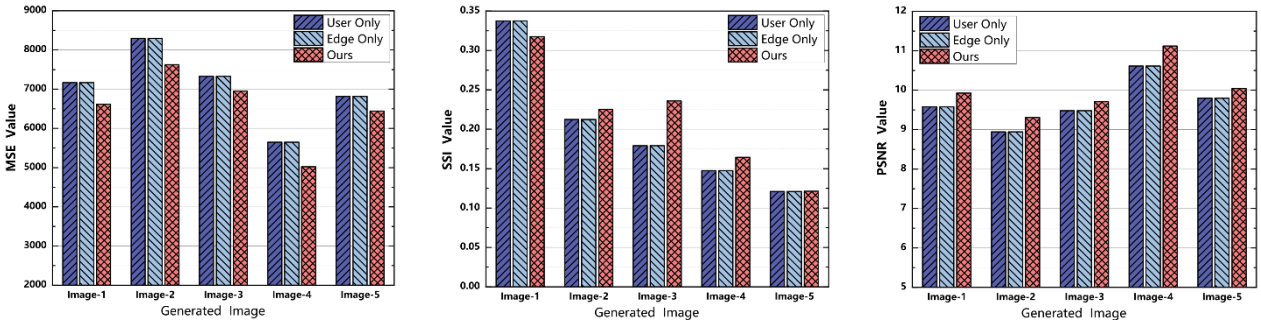
To further improve user Quality of Experience (QoE), we propose the Edge-User Diffusion Model Collaborative Inference framework (Figure 73 (b)), consisting of three phases: (1) local user request, (2) collaborative edge-user inference, and (3) image generation and transmission. In this framework, diffusion models are stored in the cloud, while edge servers and users collaborate on inference. The edge server processes text prompts and generates an initial noisy image, which the user partially refines before the edge server completes the final image generation.

We conduct two types of experiments: comparative and ablation experiments to evaluate the performance of our proposed approach. The comparative experiments assess our framework's performance against existing approaches in terms of Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSI) during image generation. The ablation experiments, on the other hand, focus on comparing service latency and computational efficiency.



(a) (b) (c)

Figure 74. Visualization of image quality generated by different baselines, (a) mean square error (MSE), (b) Structural Similarity Index (SSI), and (c) peak signal-to-noise ratio (PSNR) [A2].



(a) (b) (c)

Figure 75. Visualization of image quality generated by different ablation studies, (a) mean square error (MSE) (b) Structural Similarity Index (SSI), and (c) peak signal-to-noise ratio (PSNR) [A2].

Figure 74 and Figure 75 present the simulation results of image generation [A2]. The MSE values for our algorithm are lower than those of the comparison algorithms for most images, while the SSI values for our

method are higher than or comparable to those of the other algorithms, highlighting its effectiveness. These results suggest that our framework consistently maintains high image quality across various images and scenes, demonstrating its strong generalization ability. Additionally, the PSNR values for our algorithm are either higher than or comparable to those of the other algorithms, further validating the effectiveness and superiority of our approach.

Dynamic Edge-AI service management and self-adaption

As aerOS supports AI applications, it will facilitate the operation of various AI services across the edge continuum, as well as the entire workflow of AI applications, particularly those for IoT use cases. These workflows primarily include data acquisition, data transmission, and data processing. In an AI-powered IoT application, data acquisition refers to the sensing process, which gathers relevant data from the surrounding environment to support application decision-making. The acquired data is then processed by a pre-trained AI model to generate decision outcomes. aerOS enables the distributed deployment of AI services, ensuring efficient utilization of limited edge resources.

Additionally, frugal AI technologies such as model pruning enhance resource flexibility by allowing AI services to provide different levels of QoS through AI models with varying parameter sizes. Generally, AI models deployed within the edge continuum are trained based on available data and AI training technologies. Service providers can upgrade their maintained AI models as they collect more data or adopt new training techniques. These upgrades can improve QoS and resource efficiency, leading to higher inference accuracy, lower memory consumption, and reduced computational overhead. However, such upgrades can also alter environmental conditions, as edge AI services are increasingly integrated into infrastructure, particularly in the future AI-as-a-Service (AIaaS) paradigm. This may lead to performance degradation if deployment management policies—especially DRL-based policies—are optimized for existing AI models. Therefore, the system must adapt to the evolving characteristics of upgraded AI models to maintain optimal performance as AI-driven applications become more prevalent.

To achieve this goal in aerOS, we developed an approach based on meta-reinforcement learning [A3]. Figure 76 illustrates the architecture of the proposed approach, which consists of two major components: the meta-training part and the meta-adaptation part. The meta-training process is supported by a digital twin network and offline datasets, which provide data for offline meta-training. The digital twin network emulates various environmental conditions and stores data collected from different scenarios. Meta-training is primarily conducted using off-policy and offline meta-RL approaches, allowing independent policy updates based on collected data (i.e., from offline datasets or digital twin simulations) until convergence. Through meta-training across multiple offline datasets, the context encoder embedded within the policy learns to estimate environmental features, enabling the policy to adapt to new conditions—even those different from the training environments. The meta-adaptation process is triggered when the system transitions to a new condition, such as an AI model upgrade by service providers. This process follows several key steps. First, the policy for managing the physical edge continuum is initialized using the meta-policy obtained from the meta-training phase. The policy then interacts with the edge continuum to collect a small amount of data. After gathering a mini-batch dataset, the policy is updated through a process similar to meta-training. Notably, during meta-adaptation, the context encoder responsible for estimating environmental features remains unchanged. Instead, it continuously estimates environmental features and uses them as latent system states to assist in policy decision-making as well as the update of policy.

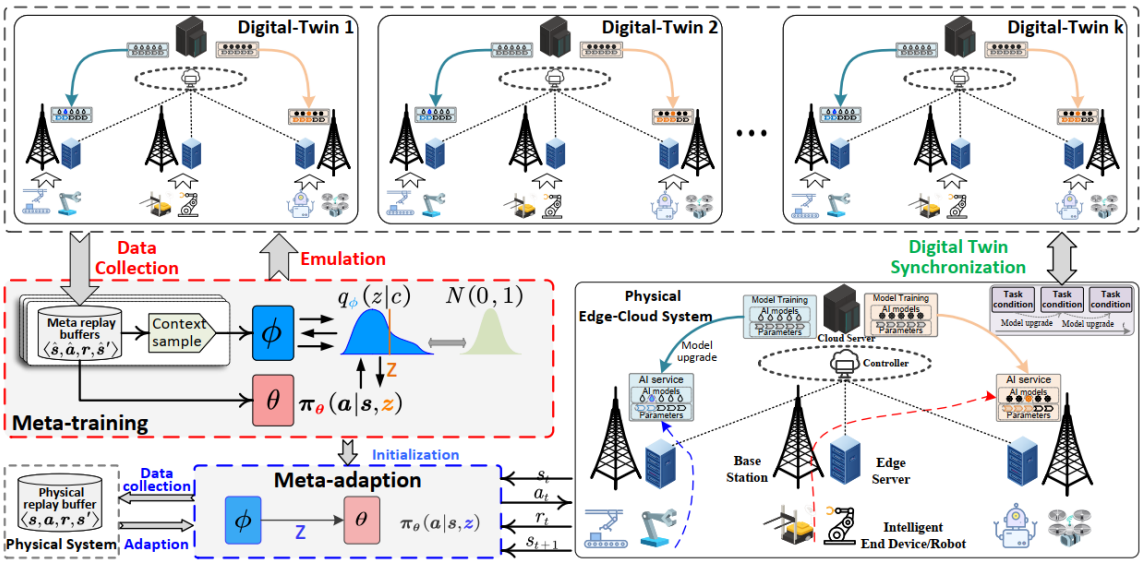


Figure 76. The architecture of meta-RL-enabled dynamic AI service adaptation [A3].

Through the proposed framework, the policy implementation and training processes are decoupled, reducing the need for online interactions with the physical edge continuum. This decoupling significantly lowers both the probability of generating risky actions during the initial implementation phase and the overall cost of policy training. We have conducted studies focused on minimizing the service latency of edge AI applications while ensuring inference accuracy through the joint optimization of data sampling, task allocation, and AI model selection. Additionally, this approach aligns with aerOS' objective of supporting AI tasks while guaranteeing performance.

Figure 77 presents the simulation results of average latency over the meta-training process in three different testing environments, where AI models exhibit varying resource and performance characteristics [A3]. The results indicate that traditional reinforcement learning (SAC) agents trained under specific environmental conditions suffer from policy mismatches and suboptimal performance, resulting in higher latency in unseen scenarios. Training in Digital Twins accelerates convergence due to increased data availability but leads to overfitting, thereby reducing adaptability. In contrast, the proposed approach effectively minimizes service latency across different AiSF conditions without requiring re-training. Furthermore, meta-RL policies enhance both stability and adaptability by inferring latent environmental contexts, ensuring robust performance in dynamic conditions.

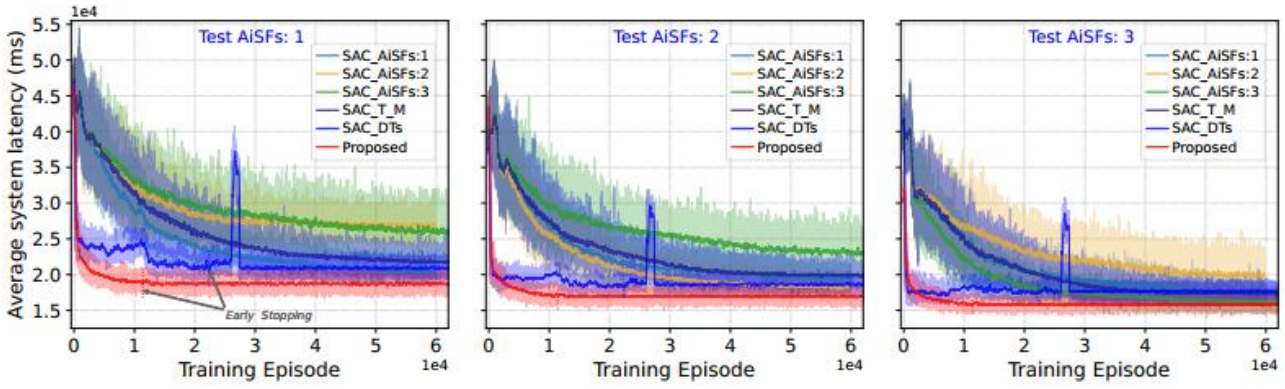


Figure 77. Average system latency comparison in different testing environments [A3].