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

D4.2 - Software for delivering intelligence at the edge intermediate release

Deliverable No.	**D4.2**	**Due Date**	**29-FEB-2024**
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.*		

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 (*third linked party*)	AT
SIEMENS AKTIENGESELLSCHAFT	DE
FIWARE FOUNDATION EV	DE
TELEFONICA INVESTIGACION Y DESARROLLO SA	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 S.A.	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
Vasilis Pitsilis	P02 NCSRD	vpitsilis@iit.demokritos.gr
Harilaos Koumaras	P02 NCSRD	koumaras@iit.demokritos.gr
Ignacio Dominguez Martinez-Casanueva	P07 TID	ignacio.dominguezmartinez@telefonica.com
Lucía Cabanillas Rodriguez	P07 TID	lucia.cabanillasrodriguez@telefonica.com
Ioannis Chouchoulis	P10 IQB	giannis.chouchoulis@inqbit.io
Konstantinos Kefalas	P10 IQB	konstantinos.kefalas@inqbit.io
Ioannis Makropodis	P10 IQB	giannis.makropodis@inqbit.io
Vasiliki Maria Sampazioti	P10 IQB	vasiliki.maria.sampazioti@inqbit.io
Aris Farao	P10 IQB	aris.farao@inqbit.io
Panagiotis Bountakas	P10 IQB	panagiotis.bountakas@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
Nikolaos Gkatzios	P15 INF	ngkatzios@infolysis.gr
Eugenia Vergi	P15 INF	evergis@infolysis.gr
Álvaro Martínez Romero	P16 PRO	amromero@prodevelop.es
Riccardo Leoni	P19 DST	r.leoni@dstech.it
Federico Corazza	P19 DST	f.corazza@dstech.it

History

Date	Version	Change
08-01-2024	0.1	Finalize D4.2 ToC
09-01-2024	0.2	Start first round of contributions
22-01-2024	0.5	Finalized first round. Merging document with first round contributions.
24-01-2024	0.6	Start second round of contributions
16-02-2024	0.9	Version ready for round of internal reviews
29-02-2024	1.0	Final submitted version

Key Data

Keywords	Semantics, data fabric, data governance, AI, explainability, analytics, trustworthiness, management, federation, portal
Lead Editor	P07 TID – Ignacio Domínguez Martínez-Casanueva
Internal Reviewer(s)	P08 COSMOTE – Fotini Setaki P27 SIPBB – Lucie Stutz

Executive Summary

The present deliverable D4.2 "Software for delivering intelligence at the edge intermediate release" is the second of the three deliverables outlining the outcomes of the aerOS Work Package 4 tasks and provides the advances following the preliminary release (D4.1) that were achieved between the months M12-M18 of the project.

The document introduces the concept of **aerOS Minimum Viable Product (MVP)** and addresses it from the standpoint of WP4. The aerOS MVP 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 MVP by enabling the sharing of data, for a trusted and decentralized AI-based orchestration of the resources in the aerOS continuum.

IMPORTANT: This deliverable is of type OTHER. This means that D4.2 is mostly a software deliverable. While this document reports the advances of tasks T4.1-T4.6 in the period M12-M8, it must be understood together with the software release that is uploaded alongside it.

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

- **Data homogenization** task 4.1, brings an update of the two main building blocks under development: 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 introduced and applied for creating an ontology that enables the orchestration of the continuum.
- **Data governance** task 4.2, has consolidated the definition of a (semantic) data product. Building upon this definition, the architecture of the aerOS Data Fabric was specified, and the development of the building blocks that compose it has started. 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.
- Several advances on **decentralized frugal AI** have been made. Management of AI workflows includes the implementation of the AI Local Executor and AI Task Controller components, as well as the Federated Learning Training Collector and Federated Learning Repository; while explainability methods for an aerOS use case based on reinforcement learning has been conducted.
- The **embedded analytics multiplane** has finalized a first 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.
- The **trust management** component has been extended to identify how each attribute affects the trust of Infrastructure Elements to define the appropriate weight for the trust score calculation algorithm. Additionally, the trust score calculation function has been implemented and prepared for future deployments in aerOS IEs.
- Finally, the **management services and the aerOS management portal** bring several updates. A first release of the aerOS management portal has been developed, along with the formal definition of the entrypoint balancer and the benchmarking tool. Regarding management services, the aerOS Federator has been designed, which combined with the developed extended capabilities of the Orion-LD Context Broker, have enabled data sharing across aerOS domains.

The finalisation of the activities will be reported with the submission of D4.3, the conclusive deliverable of the WP4 series, in M30.

Table of contents

List of tables

List of figures

List of acronyms

Acronym	Explanation
AAA	Authentication, Authorisation & Accounting
AI	Artificial Intelligence
API	Application Programming Interface
CORS	Cross-Origin Resource Sharing
CQ	Competency Question
CSR	Context Source Registration
DevPrivSecOps	Development, Privacy, Security and Operations
EAT	Embedded Analytics Tool
EB	Entrypoint Balancer
FaaS	Function-as-a-Service
FL	Federated Learning
HLO	High-Level Orchestrator
IdM	Identity Manager
IE	Infrastructure Element
LB	Load Balancing
LC	Least Connection
LDAP	Lightweight Directory Access Protocol
LLO	Low-Level Orchestrator
LOT	Linked Open Terms
LOV	Linked Open Vocabularies
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
MVP	Minimum Viable Product
NGSI-LD	Next Generation Service Interface Linked Data
OWL	Web Ontology Language
Protobuf	Protocol Buffers
RDF	Resource Description Framework
RML	RDF Mapping Language
URL	Uniform Resource Locator

1. About this document

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

In coordination with D3.2 from WP3, this document introduces the concept of the aerOS Minimum Viable Product (MVP) and details how the different tasks within WP4 have contributed to its realization. The aerOS MVP represents an intermediate stable architecture before the final architecture is delivered by WP2 later in M21.

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 D4.2 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 D4.2 influence the following work packages: • WP5 (integration, use case deployment, validation, evaluation, assessment): To later materialize solutions in pilot deployments. The contributions of D4.2 are coordinated with: • WP3 (infrastructure components): To define functional boundaries (e.g., networking, cybersecurity, orchestration) and interactions.
Milestones	This deliverable does not mark any specific milestone but constitutes an important step towards the realization of *MS6 –Software structure finished: Components of aerOS system ready, apart from final improvements and integration-related activities*, that will be achieved in M24. Although far in time (M30), this deliverable also represents a central part of *MS7 –Integrated solution: Final integrated use cases deployed and working*.
Deliverables	This deliverable builds upon the baseline architecture defined by D2.6 (aerOS architecture definition). D4.2 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. Additionally, this deliverable is coordinated with deliverables D3.2, D5.2,

	and D6.2, which are to be delivered together.

1.2. The rationale behind the structure

This deliverable is structured into three sections. **Section 2** provides an overview of the aerOS MVP from the perspective of the WP4. **Section 3** includes subsections dedicated to each of the tasks within WP4 presenting the advances made with respect to the previous D4.1. Finally, the document concludes with **Section 4**, drawing conclusions and setting future lines of work to be addressed in D4.3, which represents the final deliverable planned for WP4.

1.3. Outcomes of the deliverable

First, data homogenization has provided new releases of the Semantic Annotator and the Semantic Translator tools. Additionally, the LOT methodology has been introduced to the consortium and was applied for the creation of the aerOS continuum ontology, used in WP3 for the smart orchestration of continuum resources.

The architecture of the aerOS Data Fabric has been refined and implementations of the Data Product Pipeline and the Data Product Manager are provided. Similarly, a first version of the RDF-to-NGSI-LD translator has been delivered. Integration with OpenLDAP for collecting metadata aimed at improving data governance, and integration with cybersecurity services from WP3 for enabling authentication and authorization in the Data Fabric.

Decentralized frugal AI has defined the architecture and the mechanisms provide components to enable execution of AI workflows and to support AI explainability services.

Trustworthiness and decentralized trust management has delivered an initial implementation of the trust score algorithm supported by Trust Manager and Trust Agents. Similarly, a first implementation of trustful decentralized exchange based on IOAT is provided.

The Analytics Engine Tool (EAT) rolled out a new release, including templates for creating user-defined functions as well as built-in functions based on popular data science libraries.

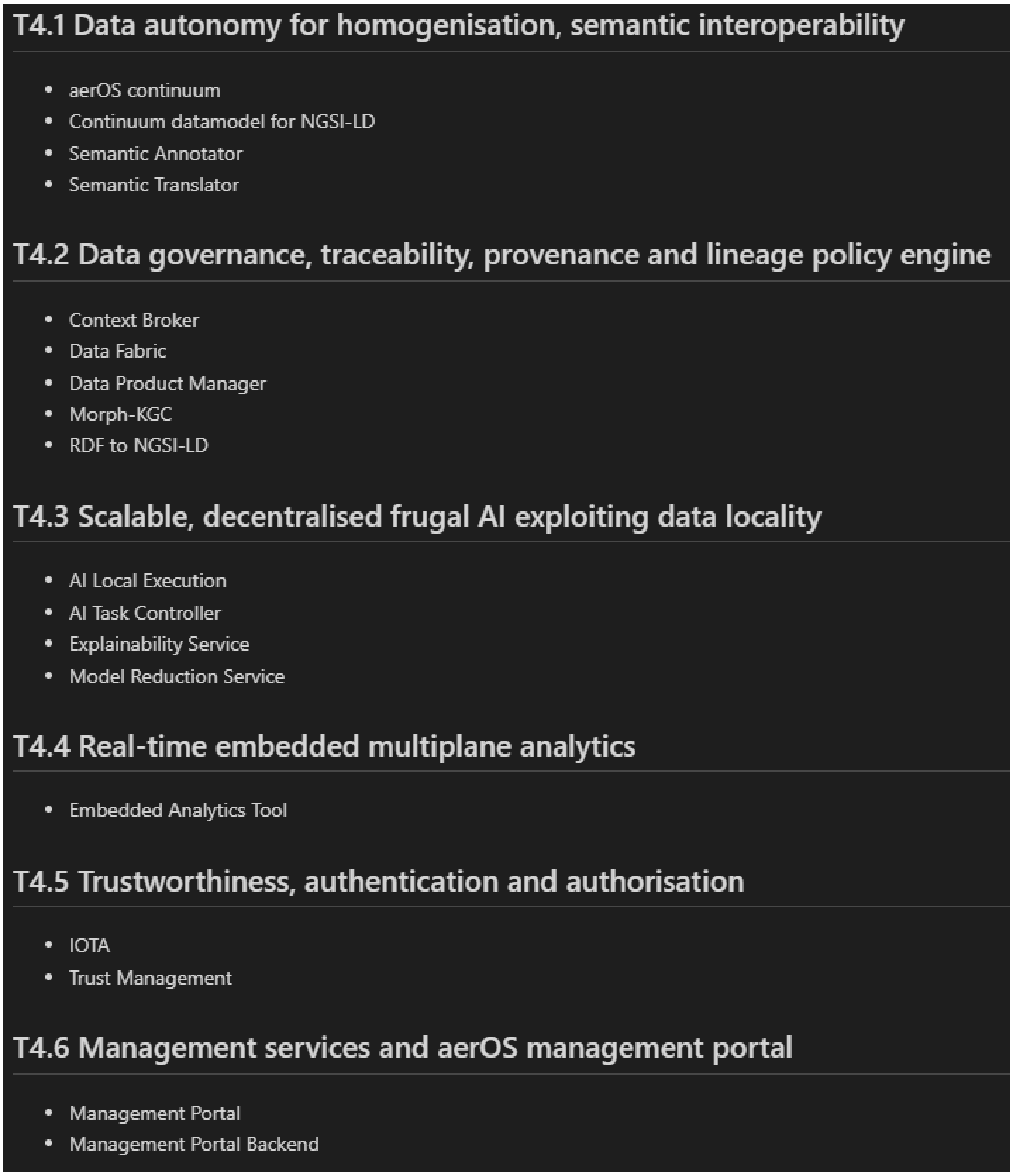
Lastly, regarding the management services and aerOS management portal, this is the first deliverable where this task reports results. A first version of the management portal has been implemented with features for managing aerOS continuum. This document includes snapshots depicting the frontend of the portal. On the other hand, the aerOS federator has been designed as part of the management services, along with extension on Orion-LD for sharing data between multiple Context Brokers.

1.4. Version-specific notes

As mentioned above, this deliverable is of type OTHER. This means that D4.2 is mostly a software deliverable. While this document reports the advances of tasks T4.1-T4.6 in the period M12-M8, 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 reflect 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 M18.

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 (D4.2). 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. Another, equivalent, release will be done by the end of the WP4 (in deliverable D4.3, due in M30).



T4.1 Data autonomy for homogenisation, semantic interoperability

- aerOS continuum
- Continuum datamodel for NGSI-LD
- Semantic Annotator
- Semantic Translator

T4.2 Data governance, traceability, provenance and lineage policy engine

- Context Broker
- Data Fabric
- Data Product Manager
- Morph-KGC
- RDF to NGSI-LD

T4.3 Scalable, decentralised frugal AI exploiting data locality

- AI Local Execution
- AI Task Controller
- Explainability Service
- Model Reduction Service

T4.4 Real-time embedded multiplane analytics

- Embedded Analytics Tool

T4.5 Trustworthiness, authentication and authorisation

- IOTA
- Trust Management

T4.6 Management services and aerOS management portal

- Management Portal
- Management Portal Backend

Figure 1. Software release of D4.2

2. MVP Overview

The aerOS project in the first year of its realization has carefully designed an architecture which is intended to provide to IoT developers a coherent, common execution environment enabling IoT services deployment and reuse that leverages the distributed capabilities all along the edge-cloud continuum. With the vision to functionally unify a multitude of diverse computing and network resources from cloud to edge even to IoT devices, aerOS has employed and combined many state-of-the-art concepts and technologies.

In parallel (and after) the design of the preliminary aerOS architecture and until M18, beyond state-of-the-art advancements have been achieved through the research and implementation progress in the fields of compute and network fabric, service fabric and data fabric, that have introduced development and integration complexities, as reflected in the great variety of technologies and tools involved. More specifically, 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 need to be employed to provide the minimum for the execution environment that aerOS requires. Additionally, cybersecurity tools and trust management need to ensure private and secure communications and access to services over the whole aerOS continuum. All Infrastructure Elements (IEs) and aerOS domains should seamlessly expose APIs for fully defined communication among components and services. Respectively, data management technologies and integrated components should support the transition from heterogeneous data silos to a unified data fabric over the continuum, 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. Finally, AI tasks are designed to run over different 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 needed to realize the continuum and all these are envisioned to be implemented encompassing assimilable cloud native practices to enable stakeholders to design, deploy, and operate scalable and resilient applications over the aerOS meta operating system. The goal is to encompass cloud-native techniques in continuum deployments, where infrastructure (physical and virtualized) ranges from IoT devices all the way up to cloud data centres. This fact implies that the great complexity emerging from the research, development, and integration of all components suggests an iterative development which should 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 worthwhile mentioning that addressing all those complexities and successfully achieving the results cannot be tackled in a single stage. Thus, following the agile schema of the project, the aerOS team has defined the Minimum Viable Product (MVP) to be available by M18 and bring together all the aforementioned technologies and tools as an integrated product. In this framework, MVP serves as a tangible aerOS prototype that can be quickly deployed and tested, realizing all the basic functionalities of the continuum. Through the development of the MVP, the aerOS team has gained invaluable insights into the feasibility and viability of the architectural concepts and has appropriately refined them in the process. Additionally, MVP supports resources' efficient use, optimizing the light-weighted systems usage to the extent possible, without compromising the core of its existence. Finally, the MVP guides the secure transition to aerOS of the pilot sites and reduces the risks in doing so, and therefore enables the pilot teams to early adapt without compromising the effectiveness of the proven key concepts.

Practically, MVP is the sandbox of validating architectural concepts and evaluating components viability and synergy. The aerOS development team consists of many technical partners working on individual assignments with the responsibility to deliver components that should be integrated and work seamlessly one with the other. Although this development is based on specifications, development contracts, APIs, and data model definitions, it could not be possible to verify the interaction and the interworking of these components as an integrated system without a sandbox environment where all things come together.

The aerOS MVP encompasses the most essential aerOS functionalities (to be enhanced in future deliveries) and integrates two aerOS domains, which are deployed in two different locations, both in geographic and administrative terms, to demonstrate its functionality over the public cloud. One domain is designed to be the entrypoint to the continuum, while the other as 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 domain 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.

As part of the project’s technical activities, the aerOS team has exhaustingly worked on designing the architecture, providing blueprints that guide the development of all components, their functionalities, and interactions. As noted, the MVP platform has been specifically designed to showcase the aerOS architectural concepts and functionalities, and an initial MVP demonstrator is already available as a month M18 achievement.

Furthermore, the MVP supports an iterative development phase. The feedback from the vertical stakeholders, the partners of the aerOS project on the first demonstrator will lead development towards enhancing existing features, fixing issues, and, perhaps, introducing new functionality guided by user demand. Additionally, the aerOS open calls are expected to contribute components which will be validated in the MVP. At the final stage, the MVP will act as a guide for deploying the aerOS stack for the project pilot use cases. The validation of core functionalities stability will pave the way for replicating aerOS deployment in the five (5) use case pilot locations. Not all of them have the exact same needs and they do not have to deploy the full aerOS stack, just these services that make them aerOS compatible and anything more suitable for their vertical domain purposes.

The aerOS MVP builds upon outcomes from both WP3 and WP4, which deliver the implementation of the aerOS concepts. WP4 undertakes all data management and AI activities and builds the aerOS “Data Fabric”, which enables the seamless integration and exploitation of a variety of data from various heterogenous sources, with the aim to support the delivery of intelligence across the aerOS continuum and over a diverse set of infrastructure resources, by optimizing the data usage o 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, Figure 2 represents the building blocks which WP4 addresses.

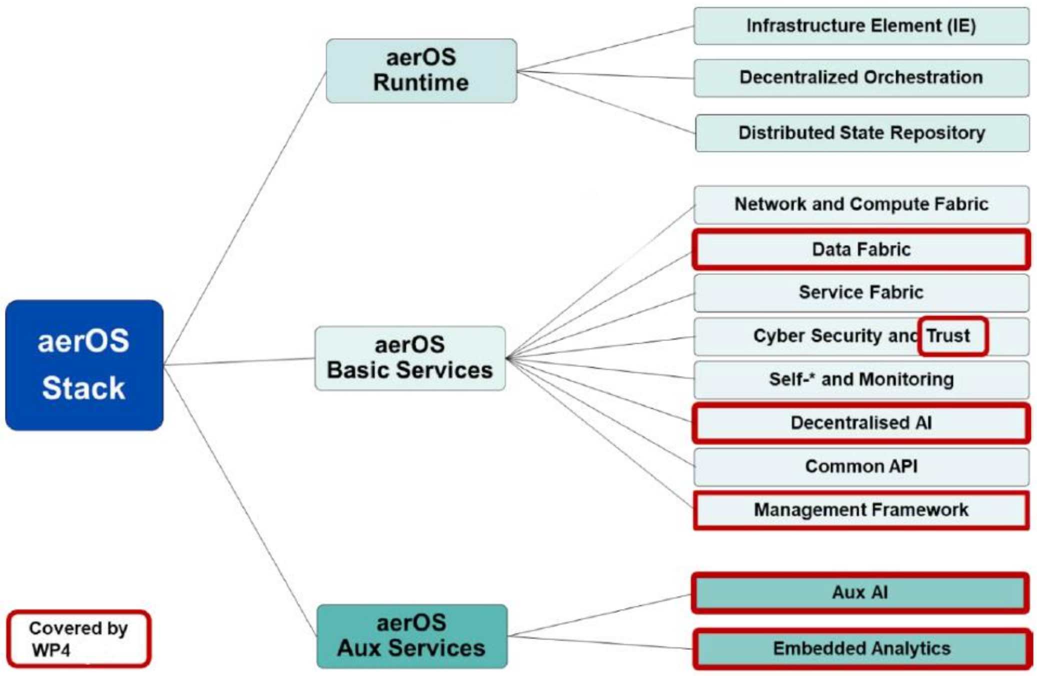


Figure 2. WP4 components in the aerOS stack

Distributed over six tasks, many diverse technologies are addressed within WP4. Each task further breaks down to a set of relevant technologies related to its domain of interest. To build an early MVP release, **priority has been given to components which are estimated to play a prominent role in establishing the aerOS continuum** with core functionalities enabled, over components that can be integrated in a second round as they will be based on core aerOS features and are not crucial to provide a prototype capable to demonstrate overall aerOS functionality.

In the grounds of MVP deployment, an early integrated environment will be implemented to fully demonstrate federated orchestration capabilities of a variety of services over a multitude of heterogeneous resources. WP4 focused on delivering the underlying mechanisms and required ontologies for building a federated knowledge graph that provides a semantic-based integration of data from heterogeneous sources and exposes them through a unified, standard interface. This provides WP3 orchestration mechanisms with the federated access to aerOS state data that reflect the status of the continuum, at each point of time, enabling thus the most efficient decisions regarding workloads deployment or mitigation on top of a set of diverse computing resources. Efforts 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.
- 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 entrypoint to the aerOS ecosystem.
- Management service to establish federation mechanisms among the multiple aerOS domains that form the continuum.

In the following paragraphs the summary of the WP4 tasks' outcomes which have been prioritised for the MVP realization and their relevant significance to the aerOS continuum establishment are presented:

- In the realm of **Data autonomy for homogenization** the ontologies which can model and reflect the state of the aerOS continuum were prioritized. The research and effort include modelling of all aerOS functional entities including aerOS domains features, Infrastructure Element (IE) capabilities, service components, and aerOS users.
- The main consideration of **Data governance, traceability, provenance, and lineage** research towards MVP, was on the mechanisms for onboarding and building data products within the aerOS Data Fabric. The target was to develop the components to facilitate the creation of semantic, interoperable data products that their respective data product owners want expose to the rest of the continuum via the Data Fabric.
- **Decentralized frugal AI** focus was to deliver explainability services which could provide insights on the reasoning of HLO allocation engine as to why specific IEs were chosen to host IOT services with declared requirements.
- Regarding the **Embedded multiplane analytics,** two main directives were explored; the first being to set up all the infrastructure which can employ real-time metrics and provide insights and alerts, and the second to provide functionless capabilities ready to host functions and algorithms which, in a second stage, are able to support programmable reactions to alerts and events coming up at the aerOS runtime.
- When it comes to **Trustworthiness and decentralized trust** management, research is focusing on defining the information that enables trustworthiness evaluation of the IEs and the IOTA mechanism to handle this information, relevant for the aerOS ecosystem but not prioritised for the MVP.
- In the area of **Management services,** aerOS development has focused on the delivery of the management portal which will be the entrypoint for accessing and managing the aerOS registered resources and services, which integrates Authentication, Authorisation and Accounting (AAA) services from WP3 to ensure controlled access to resources. Equally important effort has been invested on the management of the registration and federation mechanisms provided by the Orion-LD Context Broker, responsible to federate the aerOS domains.

The above-mentioned outcomes are varied and originate from distinct domains of expertise. The integration of these diverse components, along with these provided by WP3, within the MVP reflects the project's progress and clarifies the activities that need to be addressed next. The MVP's role within the project lifecycle is dynamic, adjusting according to the timeline requirements, the achieved milestones, and objectives targeted at each phase.

In the initial design phase, it served as a guide to the prioritization of developments by supporting the definition of what is the minimal set of features that will make aerOS viable to its first set of internal users and capable for an initial demonstration of its visions and functionalities.

Accordingly, the MVP is the field of validating architectural concepts and evaluating components viability and synergy. aerOS development team consists of many technical partners working, each one of them, on defined assignments with the responsibility to deliver components that should be integrated and work seamlessly one with the other. Although this development is based on specifications, development contracts, APIs, and data model definitions, it would not be possible to verify the interaction and the interworking of these components as an integrated system without a deployment environment where all things come together.

The MVP development and the status of its progress are used to plan the demonstrator to be showcased at the mid-term review. aerOS team has exhaustingly worked on designing the architecture, providing blueprints that guide the development of all components, their functionalities, and interactions. The MVP has been specifically designed as a platform that enables timely and efficient demonstration of architectural concepts and aerOS functionalities. Scenarios planed for the demonstrator showcase are based on a realistic environment that is underpinned by the MVP implementation.

Accordingly, the MVP will support an iterative development phase. Vertical stakeholders, i.e. partners of the aerOS project, will provide their feedback which will lead development towards enhancing existing features, fixing issues, and, perhaps, introducing new functionality guided by user demand. Additionally, aerOS open calls will provide components that will be validated in the MVP.

At the final stage, the MVP will act as a guide for deploying aerOS stack in the project's pilot use cases. The validation of core functionalities stability will pave the way for replicating aerOS deployment in the several pilots. Not all of them have the exact same needs, so they do not have to deploy the full stack. But they do at least need these core services that make them aerOS compatible. Then, they can add any other services more suited to their vertical domain purposes. The MVP facilitates the understanding of all functionalities provided and enables the selection of the ones that provide what is needed for each one of them.

As a last word it should be mentioned that the MVP development success is directly connected with the use of the aerOS DevPrivSecOps platform. aerOS development lifecycle management is based on the GitLab platform deployment1, on the UPV's premises. aerOS Gitlab does not only foster a unified development environment but also ensures that every iteration, enhancement, and refinement made to the MVP is systematically documented and each version is controlled. aerOS Gitlab enables workflows streamline, from coding and testing to deployment, allowing for real-time tracking of the MVP's evolution. Internally, the structure of aerOS GitLab groups, subgroups and projects is in accordance with the tasks and research domains of the development work packages (WP3 and WP4). Each group is built around a research domain concept. In WP4, the groups include data homogenization, data fabric, decentralized AI, analytics, trust, management. Within each group, there is a dedicated project for every component that must be developed. For example, in the data fabric group, there are projects ranging from the Orion-LD Context Broker, which stores the knowledge graph, to the "Data Product Manager", which orchestrates data product creation. All projects are documented with content that can support both development and deployment processes. This documentation supports the transfer of all aerOS development to the MVP. Permissions to these groups and projects are configured based on the internal assignments. Beyond the source code versioning workspace, aerOS GitLab also hosts the project container image and package repository, where development teams push their final products that are ready to be deployed on the MVP and the pilots, and that will be used in the future to deploy aerOS on candidate computing resources. Thus, it is obvious that the whole development process, and accordingly the MVP setup, is robustly supported by the DevPrivSecOps platform integration and processes enforcement. It is also important that what is deployed in the MVP and what is versioned in GitLab (each GitLab, tagged, version), corresponds to an MVP version (tag 1.0.0-mvp). As a result, software accompanying this deliverable, exported directly from GitLab, can be demonstrated in the aerOS MVP. More information about this process can be found in D5.2.

[1] https://gitlab.aeros-project.eu

3. Intermediate proposal of software solutions

3.1. Data autonomy for homogenization

3.1.1. Semantic annotation

The Semantic Annotator component was ported from the ASSIST-IoT project. The source code is published in the aerOS internal repository, including example integrations with aerOS and latest bugfixes. The annotation core and examples were adjusted to support annotation of JSON, XML or CSV directly into NGSI-LD. Figure 3 depicts an overview of the components that compose the Semantic Annotator and the interactions among them.

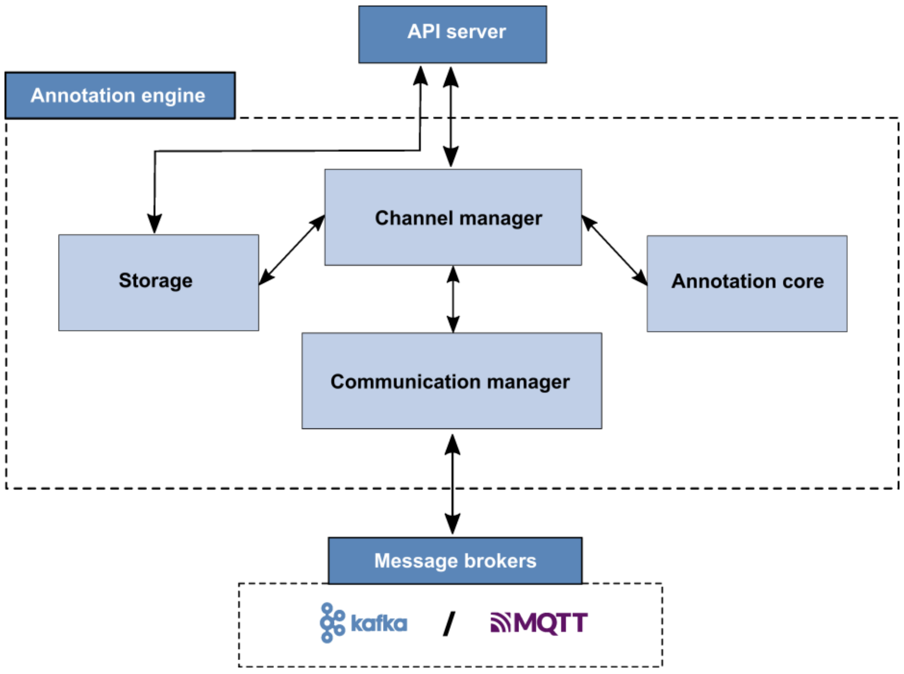


Figure 3. Semantic Annotator components overview

3.1.1.1. Technologies and standards

Table 1. Candidate technologies for Semantic Annotator

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

Since D4.1, the Semantic Translator component code, based on the ASSIST-IoT semantic-translation enabler, was successfully ported to Scala 3. In addition, the reactive Apache Pekko streaming library was used, replacing the now commercial Akka, which was used to form the backbone of the original implementation. The component code has also undergone refactoring and updating of the dependencies used. Currently, the semantic translator's communication infrastructure is able to utilize two types of communication channels: Kafka-Kafka and MQTT-MQTT, but more heterogeneous configurations are planned as an enhancement for future versions. The component was also provided with a Kubernetes configuration. The current architecture of the Semantic Translator component is depicted in the figure below.

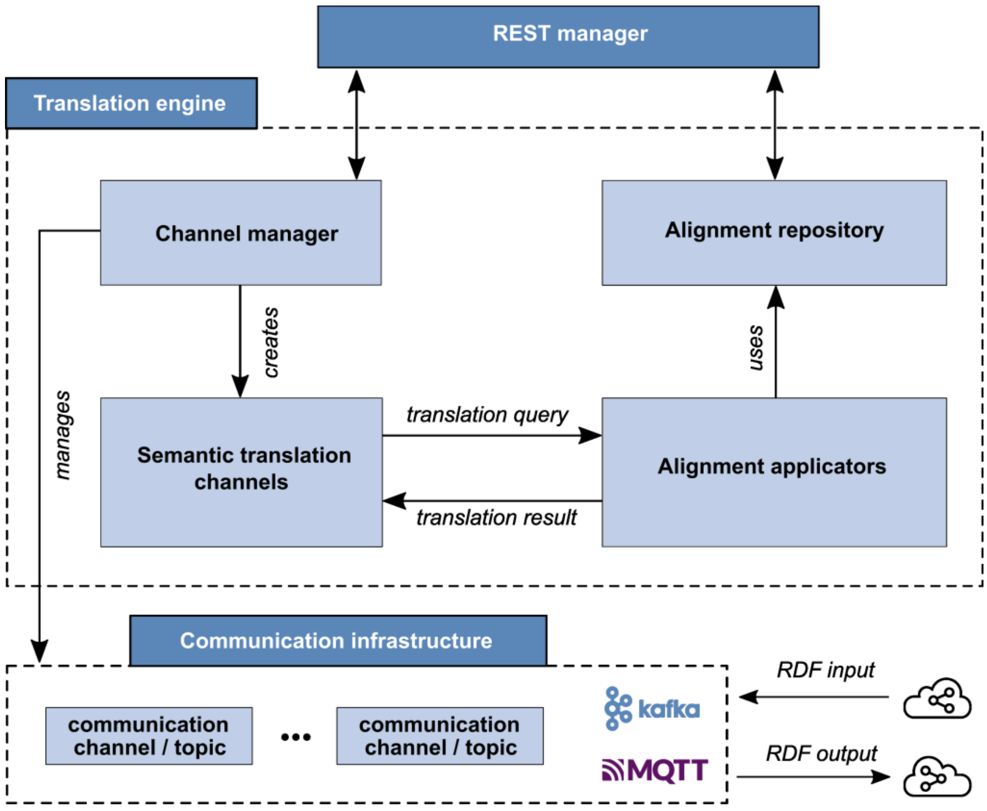


Figure 4. Semantic Translator component architecture

3.1.2.1.1. Technologies and standards

Table 2. Candidate technologies for Semantic Translator

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	RDF handling library	Translation engine

3.1.3. Ontology development

Data semantic interoperability in aerOS is achieved by means of ontologies. As described previously in D2.6 and D4.1, the aerOS Data Fabric is based on a knowledge graph, which in turn, relies on ontologies to integrate linked data. To facilitate the development of ontologies by partners throughout the lifetime of the project, some standard methodologies have been explored.

3.1.3.1. Linked Open Terms (LOT) methodology

Linked Open Terms (LOT) [1] 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.

The LOT methodology, which evolves from the NeOn methodology [2], 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 5, which is composed of the following main activities: 1) Ontology requirements specification; 2) Ontology implementation; 3) Ontology publication; 4) Ontology maintenance.

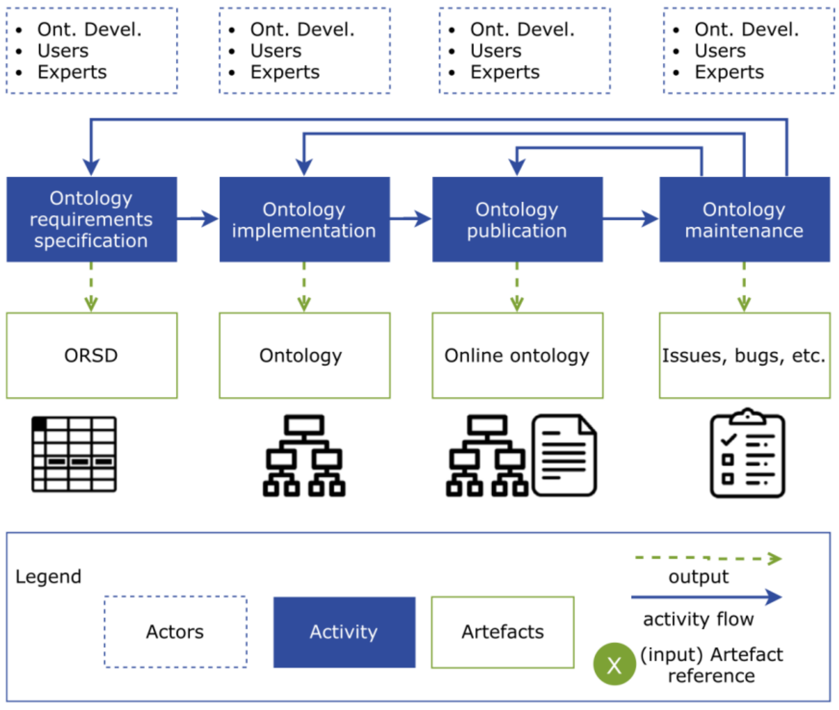


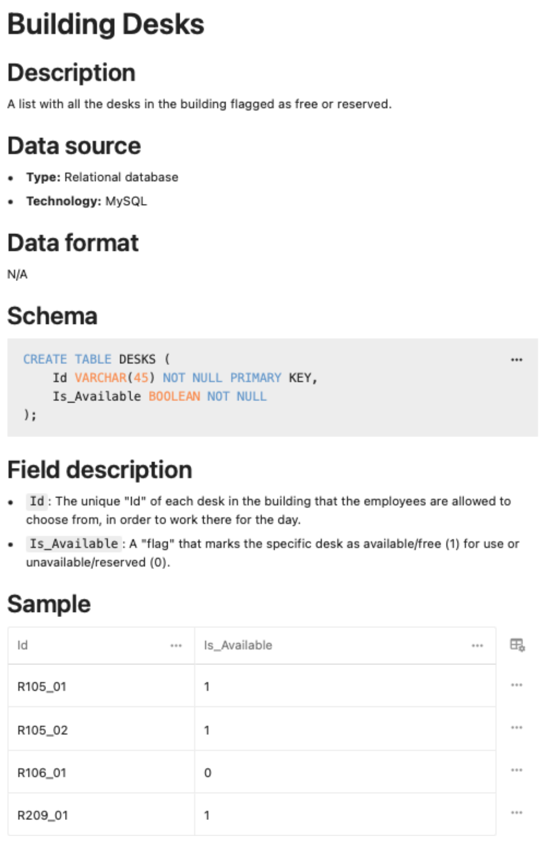
Figure 5. High-level workflow of the LOT methodology. Source [1]

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 6. 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

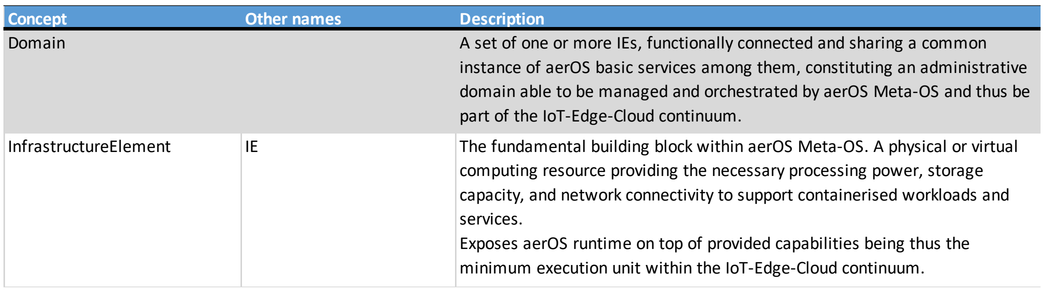
- `Id` : The unique "Id" of each desk in the building that the employees are allowed to choose from, in order to work there for the day.
- `Is_Available` : A "flag" that marks the specific desk as available/free (1) for use or unavailable/reserved (0).

Sample

Id	Is_Available
R105_01	1
R105_02	1
R106_01	0
R209_01	1

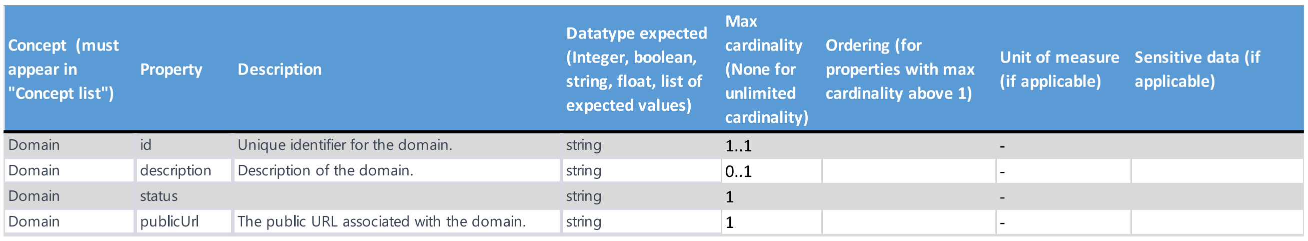
Figure 6. Sample of catalogued SQL dataset

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 [3]. 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 7) 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.



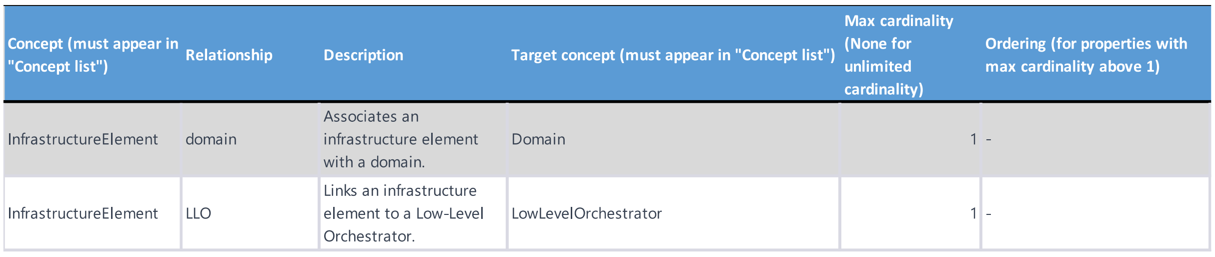
Concept	Other names	Description
Domain		A set of one or more IEs, functionally connected and sharing a common instance of aerOS basic services among them, constituting an administrative domain able to be managed and orchestrated by aerOS Meta-OS and thus be part of the IoT-Edge-Cloud continuum.
InfrastructureElement	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 7. 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 8. 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 9. 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 [4] as the standard notation and the draw.io2 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[3] 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, 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, the WIDOCO4 open-source

[2] https://app.diagrams.net
[3] https://protege.stanford.edu
[4] https://github.com/dgarijo/Widoco

tool has been chosen to generate HTML-based documentation from the ontology. This documentation includes further descriptions, examples, and the conceptual diagram from the previous activity.

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.

3.1.3.1. 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 [5]) and standardization initiatives such as OASIS TOSCA5. This ontology, as shown in Figure 10 and Figure 35 (with a larger size), 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. To create these classes, the FOAF ontology has been used, so aerOS users have assigned with an identifier from LDAP, and also present additional information such as their given and last name following the FOAF Person concept. This aerOS user is member of an organization, which has a set of predefined roles that will map to the permissions to perform certain actions in the aerOS Meta-OS. Therefore, each user is member of an organization and has assigned a role within this organization. Finally, organizations are owners of the aerOS domains.

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. Thus, Infrastructure Element emerges as the central piece of the ontology. This class contains attributes that describe general information (IP address, MAC address, geographical location, Operating System), aerOS related information (domain, linked Low-Level Orchestrator, computing tier, status), the hardware computing capabilities (number of CPU cores, CPU architecture, RAM capacity, real time capability) and real time status for monitoring (status, CPU usage, available RAM and current usage, average and current power consumption). The status of an IE plays a crucial role in the aerOS orchestration process because only IEs with a ready status will be available for running new workloads. Currently, four possible statuses have been defined for the IEs, following the TOSCA specification: untrusted, unsecure, overload and ready.

Nevertheless, isolated IEs entities are not enough to depict the aerOS continuum, so conceptual layers must be added on top of IE entity. The domain entity groups a set of IEs and Low-Level Orchestrators (LLO), presents a single public URL, a boolean attribute to indicate if the domain is the entrypoint of the continuum and a custom status as the IEs (preliminar, functional or removed). More detailed information about the domain status can be found in section 3.2 of D5.2, where it is described the aerOS installation process indeed. Moreover, each IE is linked to a single LLO that is mapped to a certain container orchestration technology: Docker, Kubernetes,

[5] TOSCA Version 2.0 Draft 05 https://docs.oasis-open.org/tosca/TOSCA/v2.0/TOSCA-v2.0.html

containerd… For the MVP, only have been developed LLOs of Docker and K8s orchestration type, as described in section 4.3.1.3 of D3.2.

The last block represents the deployment of services in the IEs of the continuum. These entities not only represent the status of already deployed and running services, but also all the stages of the services through the whole 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 and the usage of these entities by the different aerOS orchestration components is depicted in the aerOS orchestration process, which is described in detail in section 4.3.1.1 of D3.2.

In the above section 3.1.3.1.3 it is described how it is finally published an ontology through a web page using Gitlab CI/CD pipelines. Therefore, some screenshots of the published aerOS continuum ontology have been added in annex B, as the resulted Gitlab page is still non-open to the public internet.

Finally, it is important to highlight that this is a live ontology that will be enhanced as the project progresses, so some parts of the continuum may not be covered yet or may even be expanded. For instance, the networking definition of Service Components must be improved, and persistent storage requirements must be included in the ontology. In addition, this first consolidated version will need a proper testing process for tine-tuning. Therefore, this testing process has started along with the development of the MVP but will be improved in the next months, as the development of the aerOS Basic Services evolves.

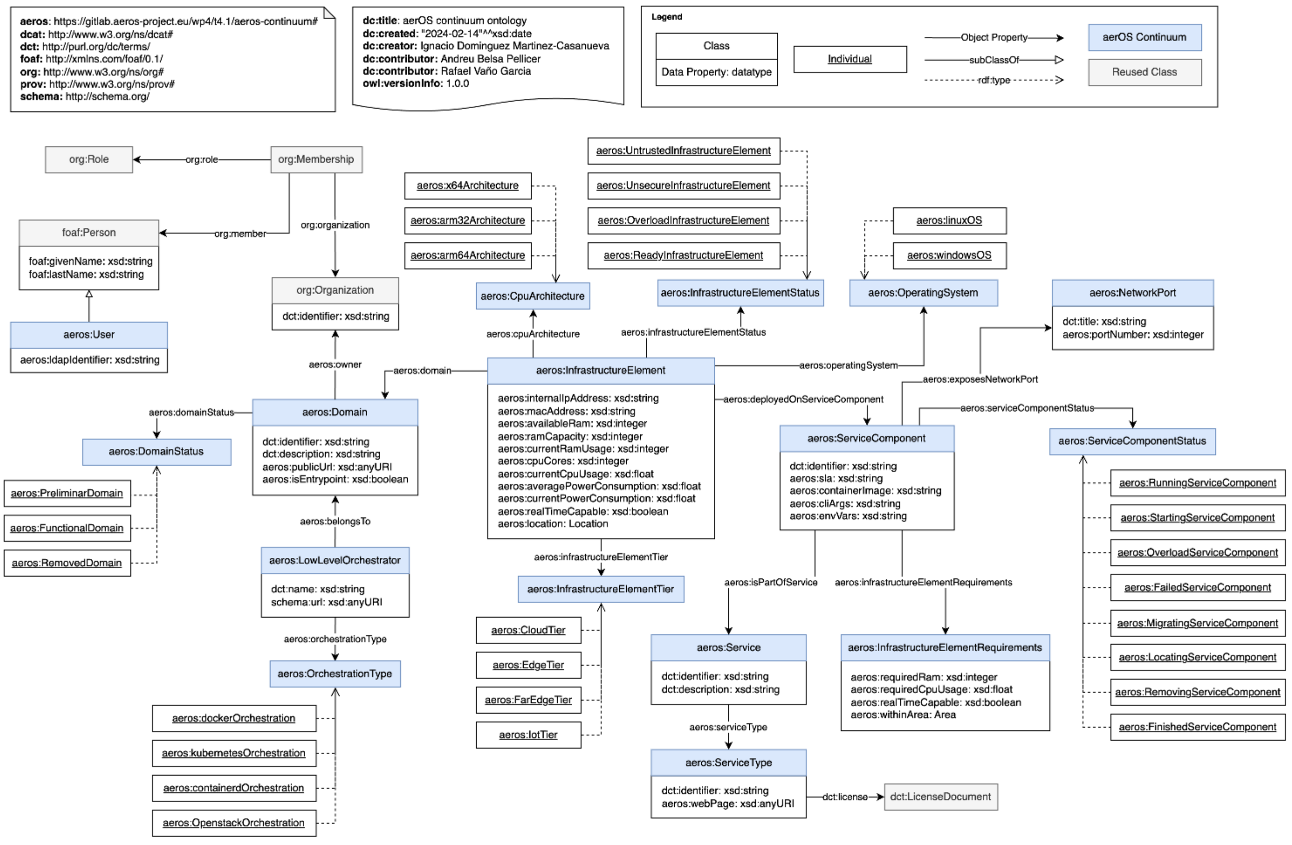

aeros: https://gitlab.aeros-project.eu/wp4/t4.1/aeros-continuum#
dcat: http://www.w3.org/ns/dcat#
dct: http://purl.org/dc/terms/
foaf: http://xmlns.com/foaf/0.1/
org: http://www.w3.org/ns/org#
prov: http://www.w3.org/ns/prov#
schema: http://schema.org/
dc:title: aerOS continuum ontology
dc:created: "2024-02-14"^^xsd:date
dc:creator: Ignacio Dominguez Martinez-Casanueva
dc:contributor: Andreu Belsa Pellicer
dc:contributor: Rafael Vaño Garcia
owl:versionInfo: 1.0.0
Legend
Class
Data Property: datatype
Individual
Object Property
subClassOf
rdf:type
aerOS Continuum
Reused Class
org:Role
org:role
org:Membership
org:member
org:organization
foaf:Person
foaf:givenName: xsd:string
foaf:lastName: xsd:string
aeros:User
aeros:ldapIdentifier: xsd:string
org:Organization
dct:identifier: xsd:string
aeros:owner
aeros:domain
aeros:domainStatus
aeros:DomainStatus
aeros:PreliminarDomain
aeros:FunctionalDomain
aeros:RemovedDomain
aeros:Domain
dct:identifier: xsd:string
dct:description: xsd:string
aeros:publicUrl: xsd:anyURI
aeros:isEntrypoint: xsd:boolean
aeros:belongsTo
aeros:LowLevelOrchestrator
dct:name: xsd:string
schema:url: xsd:anyURI
aeros:orchestrationType
aeros:OrchestrationType
aeros:dockerOrchestration
aeros:kubernetesOrchestration
aeros:containerdOrchestration
aeros:OpenstackOrchestration
aeros:x64Architecture
aeros:arm32Architecture
aeros:arm64Architecture
aeros:CpuArchitecture
aeros:cpuArchitecture
aeros:UntrustedInfrastructureElement
aeros:UnsecureInfrastructureElement
aeros:OverloadInfrastructureElement
aeros:ReadyInfrastructureElement
aeros:InfrastructureElementStatus
aeros:infrastructureElementStatus
aeros:InfrastructureElement
aeros:internalIpAddress: xsd:string
aeros:macAddress: xsd:string
aeros:availableRam: xsd:integer
aeros:ramCapacity: xsd:integer
aeros:currentRamUsage: xsd:integer
aeros:cpuCores: xsd:integer
aeros:currentCpuUsage: xsd:float
aeros:averagePowerConsumption: xsd:float
aeros:currentPowerConsumption: xsd:float
aeros:realTimeCapable: xsd:boolean
aeros:location: Location
aeros:infrastructureElementTier
aeros:InfrastructureElementTier
aeros:CloudTier
aeros:EdgeTier
aeros:FarEdgeTier
aeros:IotTier
aeros:linuxOS
aeros:windowsOS
aeros:OperatingSystem
aeros:operatingSystem
aeros:NetworkPort
dct:title: xsd:string
aeros:portNumber: xsd:integer
aeros:exposesNetworkPort
aeros:deployedOnServiceComponent
aeros:serviceComponentStatus
aeros:ServiceComponent
dct:identifier: xsd:string
aeros:sla: xsd:string
aeros:containerImage: xsd:string
aeros:cliArgs: xsd:string
aeros:envVars: xsd:string
aeros:ServiceComponentStatus
aeros:RunningServiceComponent
aeros:StartingServiceComponent
aeros:OverloadServiceComponent
aeros:FailedServiceComponent
aeros:MigratingServiceComponent
aeros:LocatingServiceComponent
aeros:RemovingServiceComponent
aeros:FinishedServiceComponent
aeros:isPartOfService
aeros:infrastructureElementRequirements
aeros:Service
dct:identifier: xsd:string
dct:description: xsd:string
aeros:serviceType
aeros:ServiceType
dct:identifier: xsd:string
aeros:webPage: xsd:anyURI
dct:license
dct:LicenseDocument
aeros:InfrastructureElementRequirements
aeros:requiredRam: xsd:integer
aeros:requiredCpuUsage: xsd:float
aeros:realTimeCapable: xsd:boolean
aeros:withinArea: Area

Figure 10. Conceptual model for the aerOS continuum ontology

3.2. Data governance, traceability, provenance, and lineage

The architecture of the aerOS Data Fabric has been designed upon the data mesh principles, especially the management of data as a product. In this release of the Data Fabric for the aerOS MVP, the data product definition has been consolidated, thus determining which components are required by the architecture.

A data product in aerOS is the combination of data, metadata and software that make the data align with the FAIR principles [6]. 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.

Based on this data product definition, the architecture of the Data Fabric for the aerOS MVP has evolved from the original architecture presented in D4.1. Figure 11 illustrates the high-level architecture of the aerOS Data Fabric, which is composed of the following blocks:

- **Data Catalogue:** Maintains a registry of all the available data, their data sources, and additional governance metadata such as ownership or domain. Can collected metadata from external sources but also receives governance input from the data product owners.
- **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.
- **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 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 Catalogue and Data Security components to govern the new data products.
- **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., multiple aerOS domains), each Context Broker will provide a fragment of the global Knowledge Graph of the continuum.

Concerning the deployment of the Data Fabric in the aerOS MVP, only 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. In the following subsections, the building blocks of aerOS Data Fabric are explained in more detail.

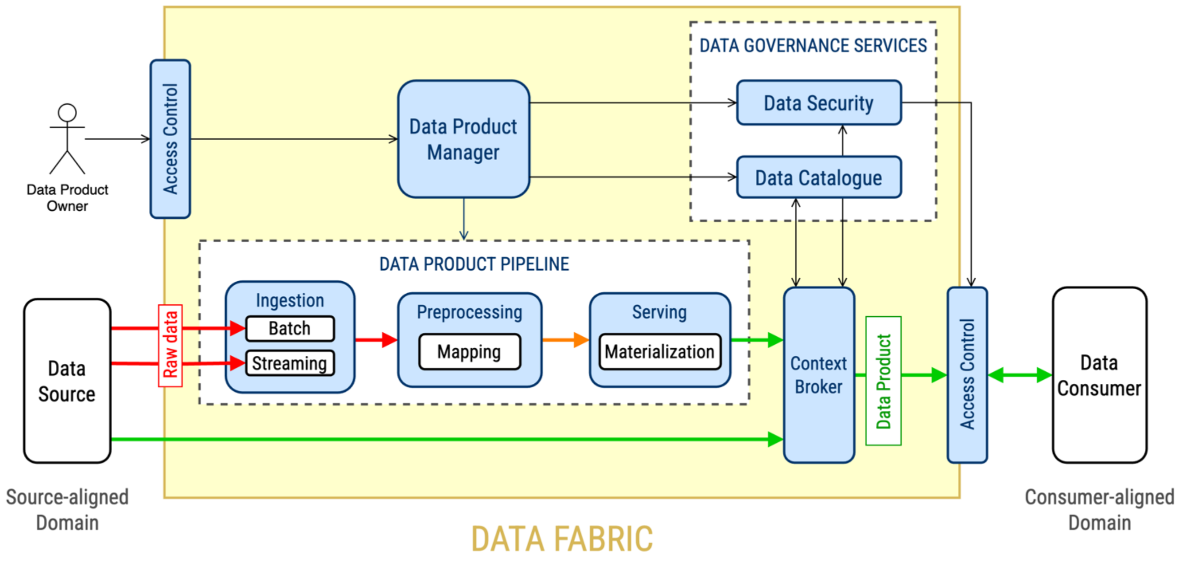


Figure 11. High level architecture of the aerOS Data Fabric

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 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 enable the temporal NGSI-LD API.

3.2.2. Data Product Pipeline

Datasets semantically annotated according to an ontology, and following 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.

The Data Product Pipeline, as illustrated in Figure 11 and previously presented in deliverable D4.1, 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 12 depicts a low-level architecture of the Data Product Pipeline, indicating which technologies have been used for components involved in each stage. In the following subsection, these components are described in detail.

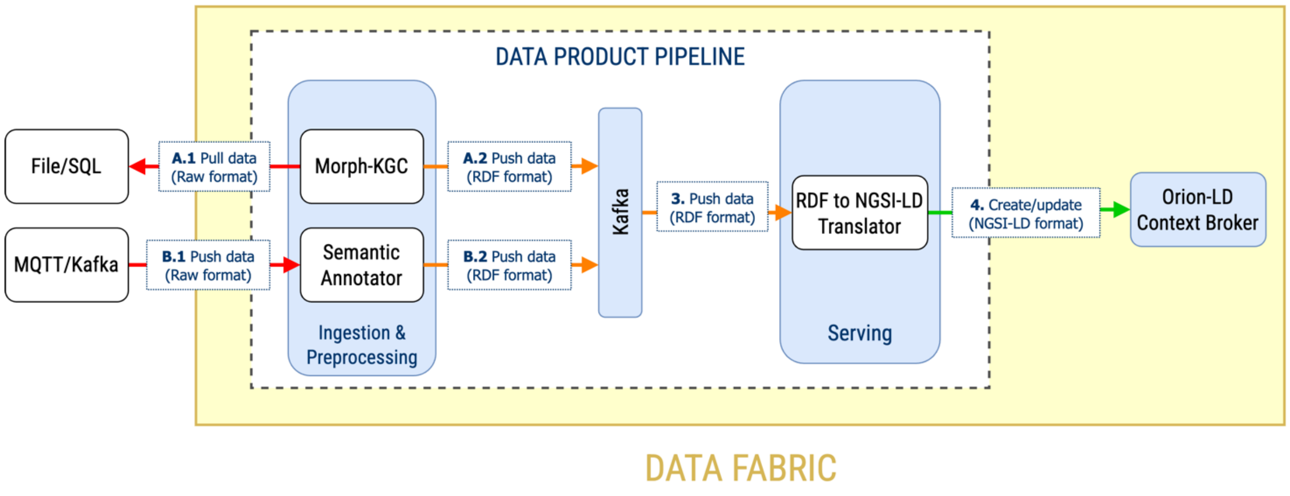

DATA PRODUCT PIPELINE
File/SQL
MQTT/Kafka
A.1 Pull data (Raw format)
B.1 Push data (Raw format)
Morph-KGC
Semantic Annotator
Ingestion & Preprocessing
A.2 Push data (RDF format)
B.2 Push data (RDF format)
Kafka
3. Push data (RDF format)
RDF to NGSI-LD Translator
Serving
4. Create/update (NGSI-LD format)
Orion-LD Context Broker
DATA FABRIC

Figure 12: Low level architecture of data product pipeline

3.2.2.1. Ingestion and Preprocessing

3.2.2.1.1. Morph-KGC

Morph-KGC [7] 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 13. The work takes inspiration from rdflib6 plugins that store RDF data in backends like Neo4j7. In this sense, this project 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.

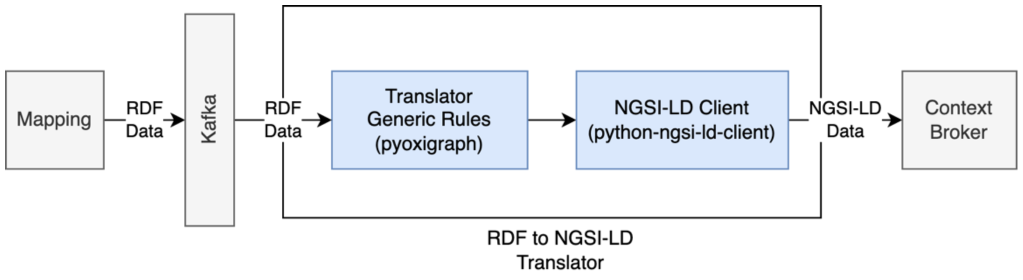

Mapping
RDF Data
Kafka
RDF Data
Translator Generic Rules (pyoxigraph)
NGSI-LD Client (python-ngsi-ld-client)
NGSI-LD Data
Context Broker
RDF to NGSI-LD Translator

Figure 13. Detailed behaviour of the RDF to NGSI-LD. Depicts the libraries used by the component and the data flows

For the aerOS MVP, a first version of this component has been implemented with the pyoxigraph[8] Python library. It provides high performance as the core code has been developed in Rust, while a friendly interface is exposed through a set of Python functions that allows for reading and writing RDF. Additionally, pyoxigraph supports an early implementation of the still ongoing RDF-star standard, which aims at providing interoperability between RDF and property graphs.

In the following releases, an implementation of the translator based on the popular rdflib library wil also be explored. 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.
- **Predicate**: 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.
    - **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": {
```

[6] https://rdflib.readthedocs.io/en/stable/index.html
[7] https://github.com/neo4j-labs/rdflib-neo4j
[8] https://pyoxigraph.readthedocs.io/en/stable/

```
                    "@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 the 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 sends the resulting creation or update of NGSI-LD Entities to the specified Context Broker. To this end, the translator leverages the python-ngsi-ld-client project9, which implements a Python client generated from the OpenAPI specification of the NGSI-LD API.

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 library10. Building upon the proposed definition of a data product, the REST API expects the following metadata and artifacts for onboarding new data products in the Data Fabric:

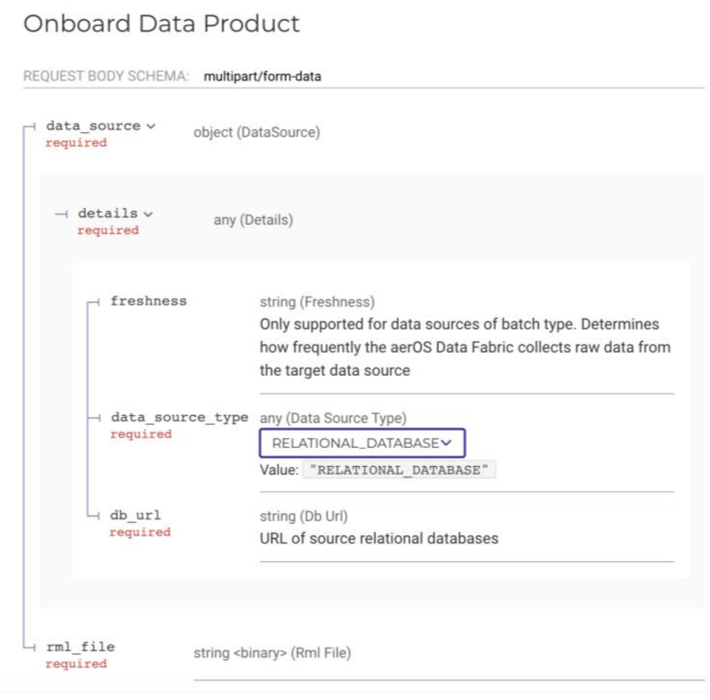
- <u>**Data Product Creation**</u>
  - **Data source configuration**: Indicates the type of data source along with connection details such as source URL (e.g., JDBC URL in relational databases) and access credentials (e.g., username/password, certificate). Additionally, the following information might 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.
    - **Data product serving:** (A) Materialization (by default), which stores the data in the Context Broker; (B) Virtualization, which builds and serves the data product on-demand by means of a Context Source.
  - **Mapping**: The following two methods for data mapping are under exploration:
    - **Declarative:** File with declarative mapping rules (e.g., RML). Artifact that describes the mappings to transform the ingested raw data into a graph structure and semantically annotate the data based on referenced ontologies.
    - **Programmatic:** Custom application that implements the data mapping step in the programming language selected by the data product developer.
- <u>**Data Governance**</u>
  - **Governance metadata:** 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:

[9] https://github.com/giros-dit/python-ngsi-ld-client/tree/1.6.1
[10] https://fastapi.tiangolo.com

    - Data domain
    - Data product owner
    - Data product developer
    - Business glossary terms
    - Tags/keywords
  - **Access control policies:** A typical scenario is that of a data product made available in the catalogue, but to which no-one has access. Consumers need to search for the data product and to send a request to the data owner to access it (establishing a kind of data contract with the purpose, data consumer identification and grant access for limited time). Furthermore, the data owner could also include access to policies upon data product onboarding. Both alternatives should be supported:
    - Data owner specifies security policies upon registration, which is useful when authorized consumers are known beforehand.
    - Data owner registers the data product without any policy associated. In this scenario, access to the policy is configured later on, when a data user requests access to the data product and the data owner grants him access to it.

In this release of the aerOS MVP, the Data Product Manager only supports onboarding data products from batch data sources of two types: relational database and file. Figure 14 and Figure 15 include snapshots of the documentation of the data product onboard API for relational database and file data sources respectively.

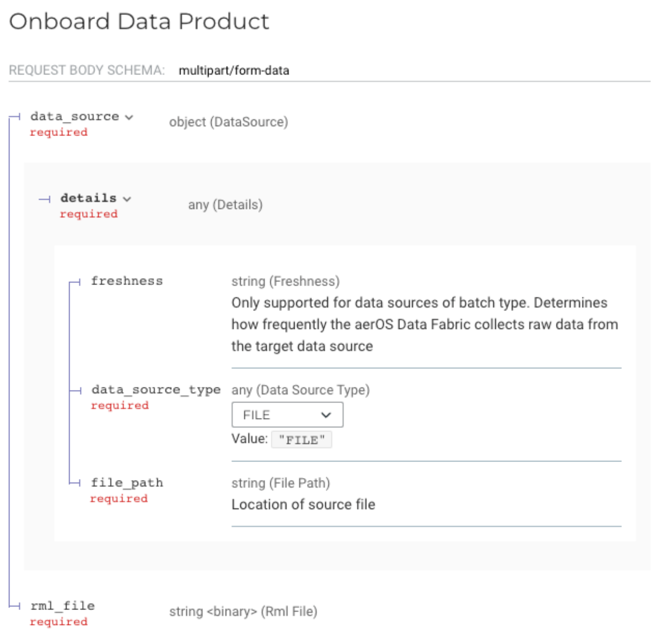
Support for streaming data sources is expected for the next release of the Data Fabric. Similarly, support for data governance features (i.e., governance metadata, access control policies) will be delivered in the future releases.


Onboard Data Product

REQUEST BODY SCHEMA: multipart/form-data

```
data_source      object (DataSource)
required
  details        any (Details)
  required
    freshness         string (Freshness)
                      Only supported for data sources of batch type. Determines
                      how frequently the aerOS Data Fabric collects raw data from
                      the target data source
    data_source_type  any (Data Source Type)
    required          RELATIONAL_DATABASE
                      Value: "RELATIONAL_DATABASE"
    db_url            string (Db Url)
    required          URL of source relational databases
rml_file         string <binary> (Rml File)
required
```

Figure 14. Documentation of the onboard data product API for relational databases



Onboard Data Product

REQUEST BODY SCHEMA: multipart/form-data

data_source required	object (DataSource)
details required	any (Details)
freshness	string (Freshness) Only supported for data sources of batch type. Determines how frequently the aerOS Data Fabric collects raw data from the target data source
data_source_type required	any (Data Source Type) FILE Value: "FILE"
file_path required	string (File Path) Location of source file
rml_file required	string <binary> (Rml File)

Figure 15. Documentation of the onboard data product API for files

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 can be deployed as Helm Charts on Kubernetes. To orchestrate the construction of the pipeline in IEs that belong to K8s clusters, 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.

3.2.4. Data catalogue

This release of the aerOS Data Catalogue brings a new component named LDAP Connector. Figure 16 shows how the component collects data from LDAP-compliant databases, like the open-source project OpenLDAP, and transforms the LDAP data into NGSI-LD data that are eventually persisted in the Orion-LD Context Broker. The LDAP Connector has been implemented as a containerized Python application that leverages the ldap312 library to interact with OpenLDAP, and the python-ngsi-ld-client to interact with the Orion-LD Context Broker.

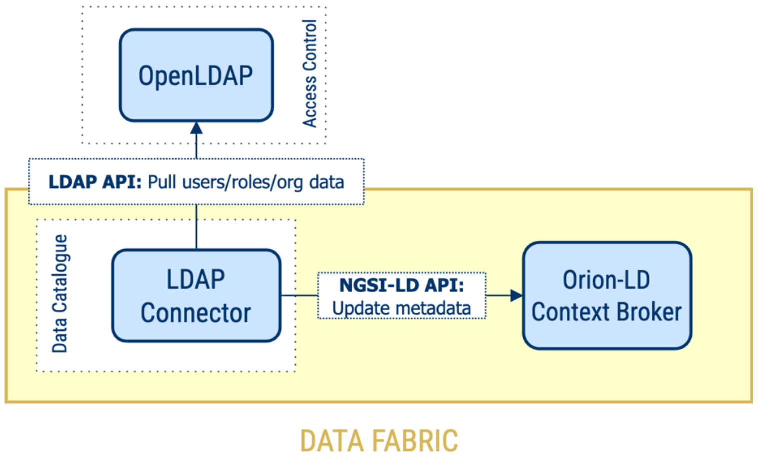


Figure 16. Data catalogue connector for LDAP

[11] https://fluxcd.io
[12] https://ldap3.readthedocs.io/en/latest/

During the workflow, the LDAP Connector periodically syncs with the LDAP database and maps the data to concepts of standard ontologies PROV [8] and ORG [9], such as Person (User), Role, and Organization. These mappings are currently under exploration and further details will be provided in the next release of the aerOS Data Fabric.

3.2.5. Data security

When it comes to security in the Data Fabric, the integration with Keycloak[13] and KrakenD[14] (T3.4) has enabled authentication and authorization in multiple components of the Data Fabric.

The first integration refers to the Orion-LD Context Broker, as depicted in Figure 17. Data consumption from the Context Broker is now only allowed to authorized consuming application (on behalf of aerOS users) that have a “data consumer” role. To this end, Keycloak is configured with a policy that only allows consumers with “data consumer” role to make GET requests to the `/ngsi-ld/v2/entities` endpoint. Before this integration, any consumer with access to a Context Broker endpoint had free access to interact with it. This policy configuration only allows to make NGSI-LD queries to the Context Broker, but next releases is expected to extend this policy to also cover the creation of NGSI-LD subscriptions to the Context Broker.

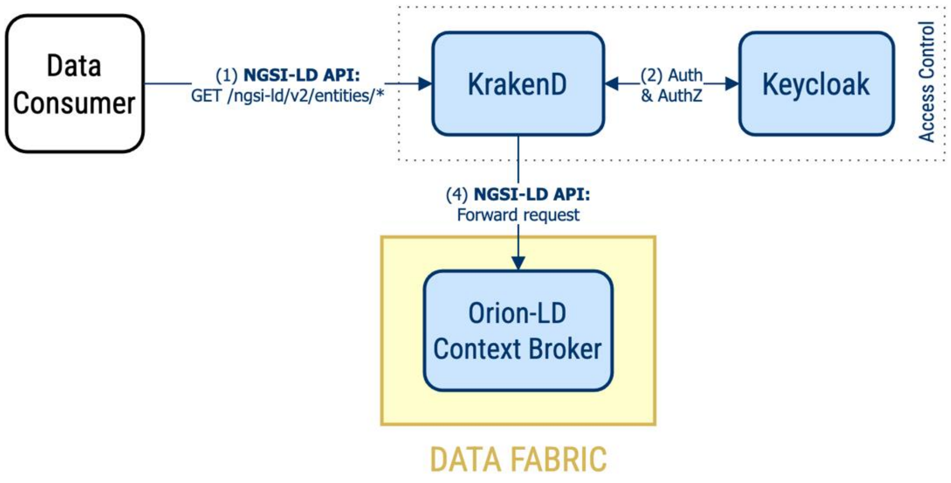


Figure 17. Authorization workflow for data consumers of the Context Broker

In future releases, this workflow will be extended to enable data product owners to define their custom access policies. For example, consuming a particular data product can be allowed only during an indicated period or for specific aerOS users, roles, or groups within an organization.

The other integration, as shown in Figure 18, involves the Data Product Manager component. In this setup, only owners of data products are authorized to onboard new data products through the REST API of the Data Product Manager. For this, Keycloak is configured with a role-based policy that only allows aerOS users with “data product owner” role to make POST requests to the `/onboard` endpoint.

[13] https://www.keycloak.org
[14] https://www.krakend.io

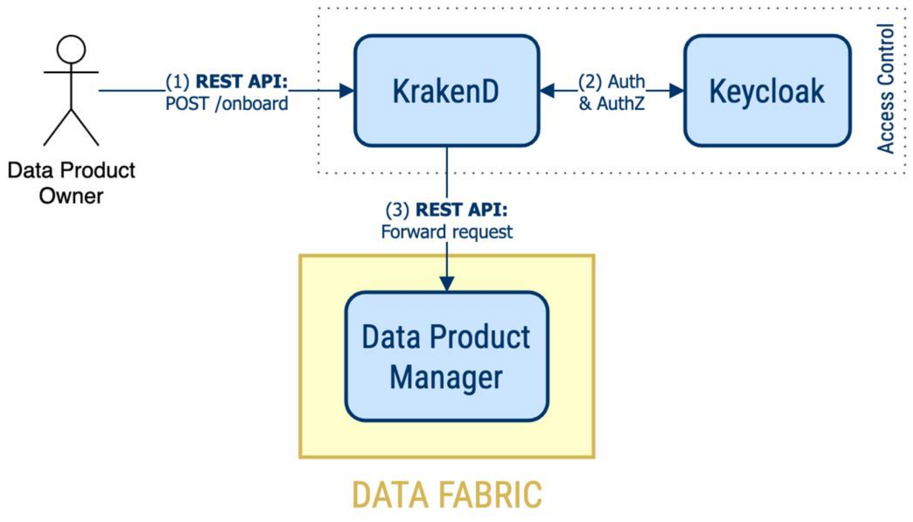


Figure 18. Authorization workflow for data product owners in the Data Product Manager

3.3. Decentralized frugal AI

The aerOS stack should enable the execution of distributed AI tasks over the continuum considering also limited resource availability that is possible close to the edge. Decentralized frugal AI mechanisms can be used for internal AI (internal aerOS mechanisms) and for external AI (the deployment of specific AI services for stakeholders over the continuum). Moreover, in T4.3 the applicability of methods for providing explainability of AI-based functionalities are investigated. This section summarizes current work status in T4.3 and refinements that have been made since D4.1.

3.3.1. AI workflows

Decentralized AI in aerOS covers two main use cases: federated training of ML model (Federated Learning, FL) using continuum resources and deployment of a trained model for prediction in the continuum.

The training of models is represented by AI workflows where FL tasks can be divided into sub-tasks and delegated for execution to IEs (see Figure 19). Besides supporting FL, aerOS will allow to deploy services with already trained models in the continuum. Such deployment will also be guided by requirements that will allow to choose the best place for deployment using general aerOS orchestration mechanisms.

The AI workflow is a combination of workloads, put together in a way to achieve a specific functionality of AI. It has been designed that the orchestration of AI workflows will make use of the aerOS orchestration. Here, the specific relations among the workloads that form the workflow will be the aspect that will be managed by the AI Service Controller. The AI task/workflow will be deployed as a set of interacting services - AI Task [n] Controller and AI Local Executors.

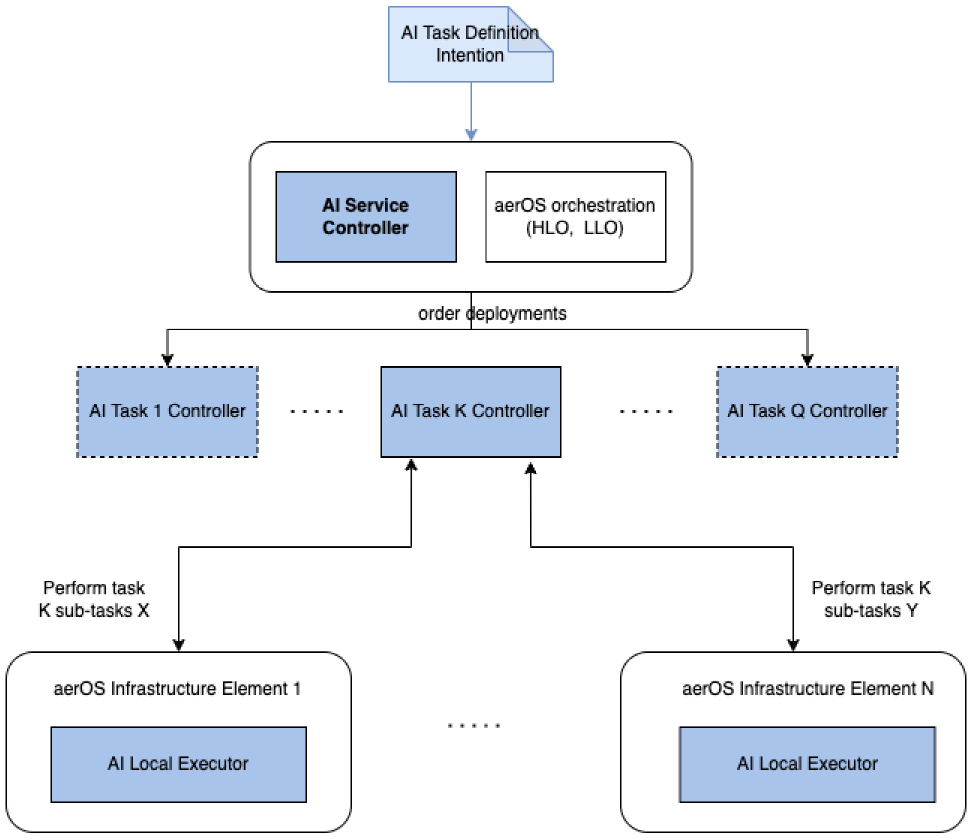

AI Task Definition Intention
AI Service Controller
aerOS orchestration (HLO, LLO)
order deployments
AI Task 1 Controller
AI Task K Controller
AI Task Q Controller
Perform task K sub-tasks X
Perform task K sub-tasks Y
aerOS Infrastructure Element 1
AI Local Executor
aerOS Infrastructure Element N
AI Local Executor

Figure 19. Update from D4.1: The concept of internal AI Local Executor called Agent was dropped. The internal components division was updated

AI Task Controller is a service deployed on aerOS infrastructure to control the execution of an AI task with respect to synchronization of partial results. AI task can be decomposed into sub-tasks that e.g., produce results that need to be aggregated or prepare data to be consumed by the model. For each AI task a dedicated AI Task Controller is deployed.

AI Local Executor is a service deployed on aerOS infrastructure to execute workload for an AI sub-task i.e., to execute a granular "step in the workflow".

3.3.1.1. Structure

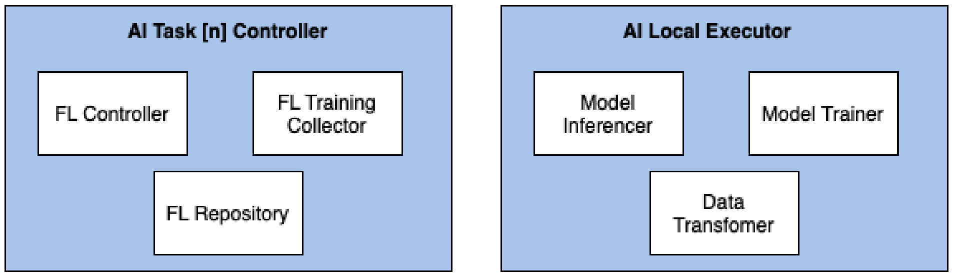


Figure 20. Internal components or AI Task Controller and AI Local Executor services

Table 3. 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 task lifecycle.

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 ML aggregation algorithms, ML models and the results of ML training.

Table 4. Components of AI Local Executor

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.

3.3.1.2. Technologies and standards

Table 5. Candidate technologies for AI workflows

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.	Communication
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. 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

3.3.1.3. Implementation status

The AI Local Executor and AI Task Controller components -FL Training Collector and FL Repository - are available. They are used to run a decentralized FL task.

Work is in progress for FL Controller (part of AI Task Controller service) component that shall accept task parameters and control task execution status. The AI Service Controller is in its design phase.

3.3.2. Frugal AI – AI Model Reduction Service

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.

AI Model Reduction Service is an auxiliary service exposing functionalities related to model compression. So far, various SotA frugal techniques have been explored, focusing mainly on the aspect of reducing the size of AI models and increasing their inference speed. These include pruning and quantization.

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.

Thus far, 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.

So far, quantization techniques have been applied in a practical setting. These include static quantization, dynamic quantization, and their combination. The tests were performed using various runtimes for inference on different hardware. 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. However, the research on the general applicability of the method in aerOS is still ongoing.

Therefore, the service features are in constant development. The current goal is to transfer the implemented methods into the aerOS ecosystem and to verify their practical usability on models coming from different services.

3.3.2.1. Technologies and standards

Table 6. Candidate technologies for AI Model Reduction Service

Technology/ Standard	Description	Component
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.	Pruner, Quantizer
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.	Pruner, Quantizer
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.	Quantizer
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 Runtime, a performance-focused engine for running ONNX models. ONNX was used to quantize models and to run them.	Pruner, Quantizer
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.	Pruner

3.3.2.2. Implementation status

The work conducted includes:

- Implementation of algorithms for structured and unstructured pruning; experimentation with RNNs and CNNs.
- Implementation of quantization algorithms; experimentation with different runtimes and hardware.

Progress is made regarding how the proposed methods can be generalized and supported by proper instructions to be easily applicable to new applications. Moreover, other frugal techniques are still to be researched.

3.3.3. Explainability support - AI Explainability Service

In aerOS, AI is used internally to support intelligent decision making when managing the continuum, and externally to enable running of arbitrary AI workflows using aerOS infrastructure. In both cases, the need may arise to explain and/or interpret predictions made by ML models. To this aim, a service will be prepared 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.

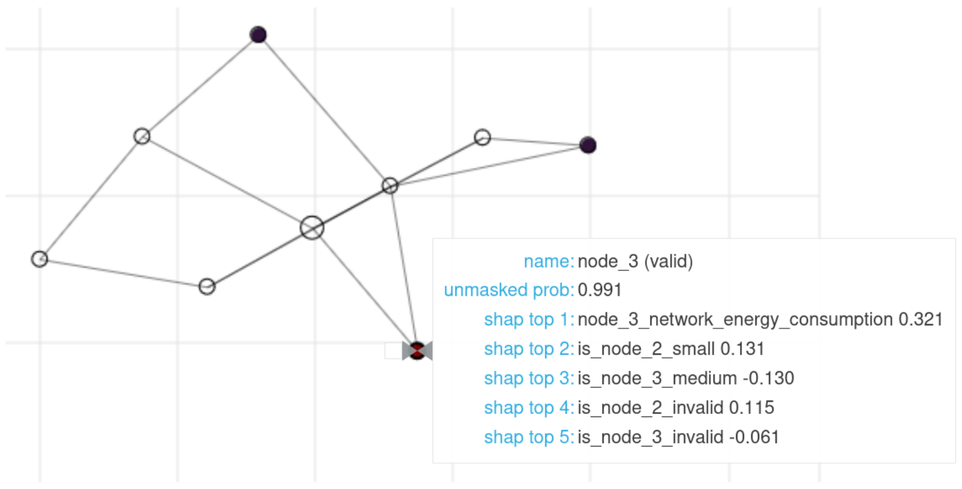
Regarding the representative aspect of the data, one can understand it as being typical or average data encountered by the model. The exact way of preparation of the dataset is something to be discovered by a service maintainer to verify that the explanations returned by the AI Explainability Service "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 behavior 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, aerOS Embedded Analytics ToolHowever, one must note that different approaches with advantages and drawbacks exist. This aspect is still to be verified in the aerOS ecosystem.

So far, various SoTA methods for explainability in AI have been analyzed. The most promising ones 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. Some examples of visualizations of explainability results are shown in the figures below.



name: node_3 (valid)
unmasked prob: 0.991
shap top 1: node_3_network_energy_consumption 0.321
shap top 2: is_node_2_small 0.131
shap top 3: is_node_3_medium -0.130
shap top 4: is_node_2_invalid 0.115
shap top 5: is_node_3_invalid -0.061

Figure 21. Example n°1 of a visualization of an explanation of a task assignment problem

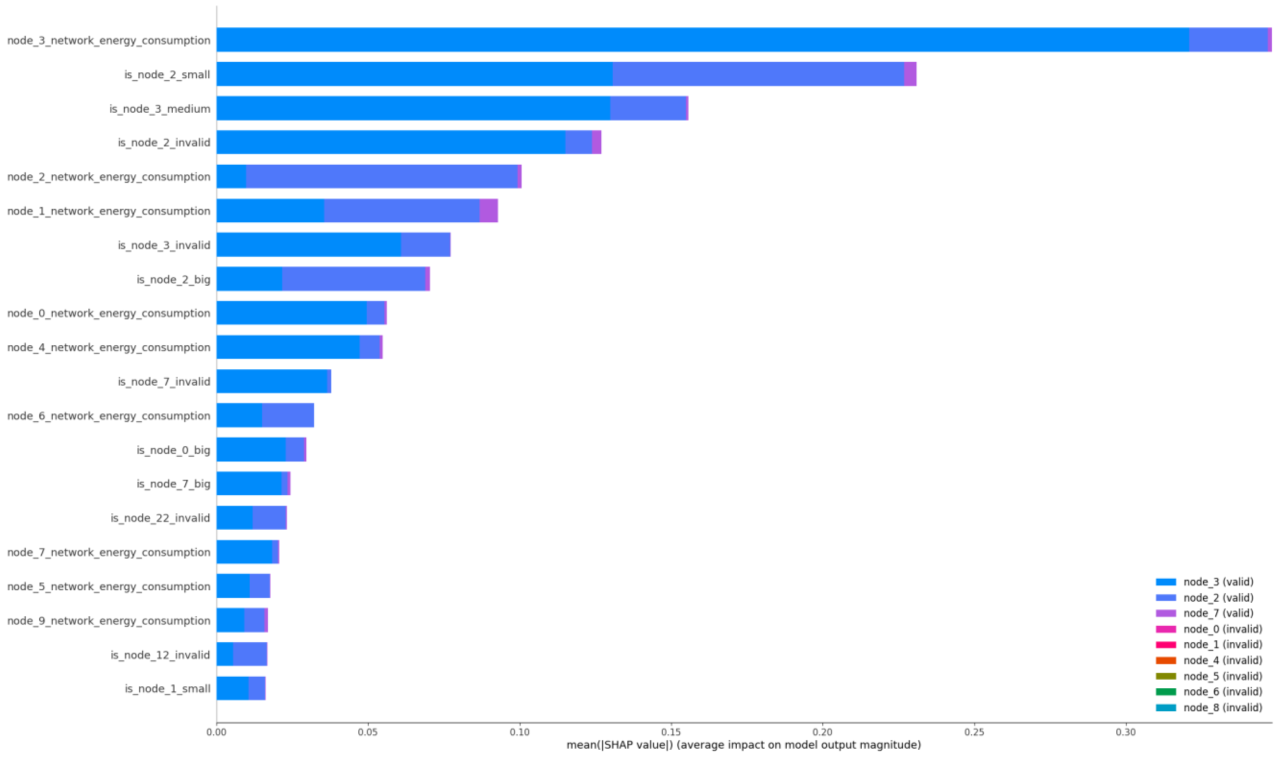


Figure 22. Example n°2 of a visualization of an explanation of a task assignment problem

3.3.3.1. Technologies and standards

Table 7. Candidate technologies for AI Explainability Service.

Technology/ Standard	Description	Component
SHAP	SHAP (SHapley Additive exPlanations) is a game theoretic approach to explain the output of any machine learning model. It connects optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions. The library would be used as a base for providing the mathematical explanations of predictions.	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

3.3.3.2. Implementation status

The work conducted so far includes research on explainability/interpretability methods that can be used for aerOS internal use case that is based on reinforcement learning. This scientific area is still being researched so unfortunately there are no “ready and easily applicable” solutions. An approach on how to proceed was designed and implemented in aerOS. Requirements for input data for the AI Explainability Service were formulated and visualization of the results was proposed.

Work in progress is for wrapping the algorithms as an aerOS service and designing final presentation of the results so that they are easily understandable to the user (note: interpretability/explainability require knowledge of the domain and problem).

3.4. Embedded multiplane analytics

Embedded Analytics Tool (EAT) is a platform which supports the creation and deployment of functions designed for the aerOS Meta-OS. These functions provide both generalised operations for aerOS users and specialised operations for specific pilot use cases. EAT also supports visualisation capabilities allowing users to investigate in-function metrics through a variety of charts and graphs appropriate to the metric.

3.4.1. Schema Implementation

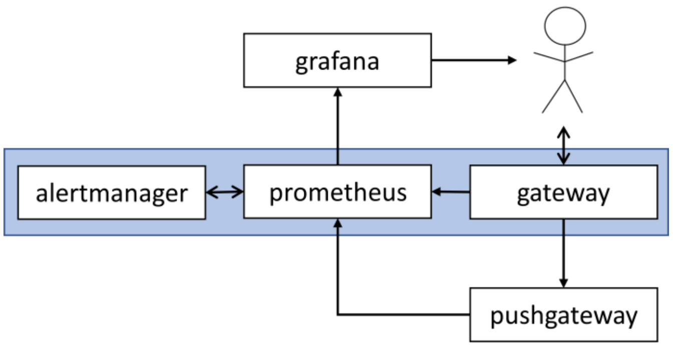


Figure 23. aerOS Embedded Analytics Tool schema

The development of the Embedded Analytics Tool is compartmentalised into subcomponents for development. The architecture shown in Figure 23 is detailed in D4.1 along with each subcomponent and interface required for communication. These subcomponents are pulled from the open-source community as independent container images and are packaged into Helm charts or Docker compose files. In addition, these images have been adapted to be deployed in the Infrastructure Elements of the continuum. Therefore, in the scope of the heterogeneous IEs that form the aerOS continuum, *faasd* stands as a promising solution for enabling the aerOS EAT capabilities in IEs with constrained resources, or in IEs that their container orchestration technology is not K8s.

While interfaces between the subcomponents are predefined and static allowing them to be configured as part of the installation, functions behave in a dynamic manner. Functions can be deployed, removed, or adapted in real-time with the capability to establish new interfaces inside the function. These interfaces are necessary for processes such as data retrieval from the Data Fabric or triggering actions from the High-Level Orchestrator.

3.4.2. Template Implementation

The flexibility of Function-as-a-Service approaches also introduces complexity regarding resource management, function design and communication. The resource management challenge can be offset through the configuration of the domains and monitoring provided through aerOS. The Function design and communication challenge can be addressed through templating. Functions designed for the aerOS Meta-OS are created using the custom aerOS template, specifically developed to map with the computing continuum features. This template provides a structure for the functions developers allowing common functionality to be abstracted and simplified. The function is divided into three distinct parts: 1) data retrieval 2) processing and 3) response. Data retrieval provides generalised code for requesting information from data sources namely the Data Fabric. Processing represents the core operation of the function where specific logic is defined. Response provides aerOS approved interfaces enabling functions to trigger additional processes such as an additional function or another aerOS component.

3.4.3. Function Implementation

The Embedded Analytics Tool is pre-packaged with three distinct functions. These are: 1) Stratified Sampling 2) Anomaly Detection and 3) Data Drift. All functions are implemented through Python using a collection of data science libraries. Stratified Sampling is a flexible function which generates a sample of data according to the parameters requested by the user. These parameters include filters for data retrieval and if the response should produce a proportional or disproportional sample. Anomaly Detection utilises a similar data retrieval process with a different logic to process the data. This function highlights outliers in the data sample which can be improved through training for use case specific scenarios. Data drift has a more complicated data retrieval process as the function is designed to test current data against historical data and identify variance over time. Based on the type of drift some algorithms are better optimised to identify this variance, our approach utilises a distribution-based algorithm to highlight data drift between historic and current data.

3.5. Trustworthiness and decentralized trust management

This section elaborates on the advances that took place in Task T4.5, since the previous deliverable D4.1. In D4.1, the concepts of trustworthiness and decentralized trust management in aerOS was thoroughly explained. However, for D4.2 to be self-sustained some repetitions might occur.

As described in D3.1, Cybersecurity and Trust are registered in the aerOS stack as Basic Services. This basic service will be covered in two tasks, namely T3.4 and T4.5. The activities of T3.4 are focused on the cybersecurity part of this basic aerOS service and are described in D3.1 and D3.2 deliverables, while T4.5 focuses mostly on the trust part; however, it also implements relevant cybersecurity actions that are in line with trust actions, such as secure information exchange using the IOTA distributed ledger technology.

3.5.1. Advances in trustworthiness of IEs in the continuum

To determine the trustworthiness of the IE (Infrastructure Element) and aerOS domain, an approximating risk assessment will be conducted for each IE. This process involves relying on reputation of neighbouring IEs to extract their trustworthiness. The summarizing scores of these IEs will the contribute to the overall trust score of the aerOS domain. The primary goal of the trustworthiness in aerOS is to provide a clear view of the level of trust for every IE and aerOS domains in the continuum. This clarity is essential to facilitate the decision-making process of aerOS components. For example, the Higher-Level Operator (HLO) depends on this trustworthiness assessment to determine which IE in the continuum are trusted for allocating services. To achieve this level of trustworthiness in aerOS, the Trust Management component has been developed, comprising the Trust Agent and the Trust Manager. These elements work together to ensure a reliable and secure aerOS environment.

On the one hand, the Trust Agent, which acts as a tracer, is responsible for collecting the necessary attributes from the IEs. These attributes are collected from the Data Fabric or the IEs using the NGSI-LD or the MQTT protocols. The list of the attributes that are collected and deployed to calculate the trust score are:

- Security events obtained from the aerOS self-security,
- Health score obtained from the health diagnose of self-*,
- Number of services that run in the IE,
- IE behaviour, namely communication with other IEs (number and trust score of these IEs), commands send/receive,
- IE updates (whether the firmware/software run in the IE is updated),
- IE reputation from other IEs or human users,
- IE system information (CPU, RAM, etc.).

On the other hand, the Trust Manager is the backbone of the Trust Management component because it contains the trust score calculation algorithm that is responsible for identifying the level of IEs' trust. This is accomplished by employing the attributes that are collected by the Trust Agent and sent to the Trust Manager. The trust score will be calculated using a weighted algorithm. The concept of calculating a trust score using a weighted algorithm is based on the understanding that not all attributes contributing to the trustworthiness of an Information Entity (IE) are equally significant. In this context, the weighted algorithm plays a crucial role in assigning appropriate importance to various attributes based on their relevance and impact.

For example, a security event occurring in an IE is considered to have a higher level of significance compared to the number of services running in an IE. This difference in significance is crucial because a security event directly impacts the trustworthiness and reliability of the IE. It could indicate potential vulnerabilities or risks associated with the IE, which are critical factors when assessing trust. On the contrary, the number of services running in an IE, while still relevant, does not necessarily have the same direct impact on the entity's trustworthiness. It might provide insights into the IE's operational capacity or utilization but does not inherently indicate how secure or reliable the IE is.

Therefore, in the weighted algorithm, greater weight or importance would be assigned to attributes like security events. This ensures that when the trust score is calculated, factors that are more critical to the trustworthiness of the IE have a proportionately greater influence on the outcome. This method allows for a more nuanced and accurate representation of an IE's trust level, ensuring that the trust score reflects the true reliability and security of the IE in its operational environment.

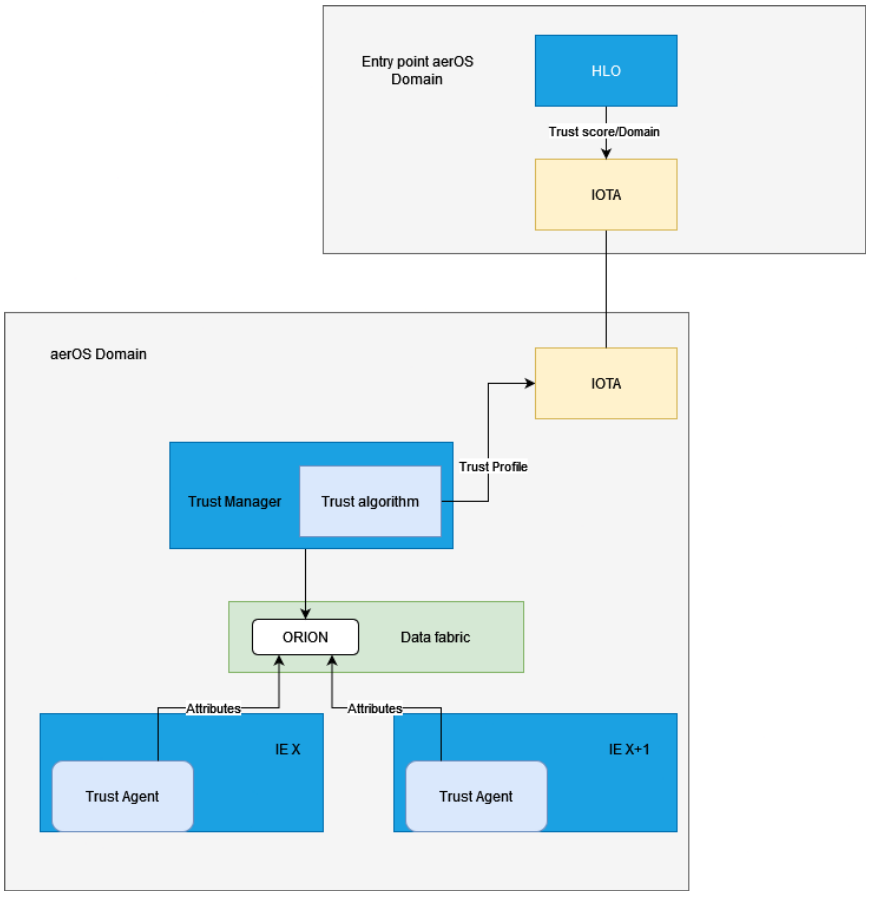


Figure 24. Updated overview of trust management structure

In the depicted overview of the trust management structure, each Infrastructure Element (IE) is equipped with a Trust Agent, who is responsible for gathering necessary information. The aerOS domain is composed of a collection of these IEs, and each domain is managed by a Trust Manager who calculates the domain's trust score. Both the Trust Agent and the Trust Manager operate using MQTT and NGSI-LD. This integration aims to facilitate communication with Orion-LD, a component within the data fabric, using the JSON-LD format, which is Orion-LD's standard communication protocol. However, the format for the trust profile, which the Trust Manager will store on the IOTA distributed ledger, has not yet been finalized.

Overall, the Trust management component interacts with several elements within the aerOS ecosystem, including the IOTA (manager), selected Self-* modules to minimize resource consumption, the data fabric/Context Broker (CB), and the IEs themselves.

From then on, the High-Level Orchestrator (HLO) in the aerOS system will use the trust score generated by the aerOS domain as a crucial input for decision-making in orchestration processes. This setup ensures that decisions regarding service allocation and system management are based on reliable and up-to-date trustworthiness assessments, enhancing the overall security and efficiency of the aerOS continuum.

3.5.2. Advances in 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 will be done accordingly with both the domain and continuum requirements.

A deployment of a private IOTA Tangle network has been done in the internal UPV infrastructure, with multiple nodes, sharing information between one another. Successful tests with data transfer between the nodes have been successful, with the results being shown in the DLT status and the capability of data to support multiple formats. The tests also include the integration of the Hornet nodes with an MQTT plugin, allowing the node itself to react to receiving relevant data from the network. 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 prototype can be seen below, with all icons taken from IOTA.

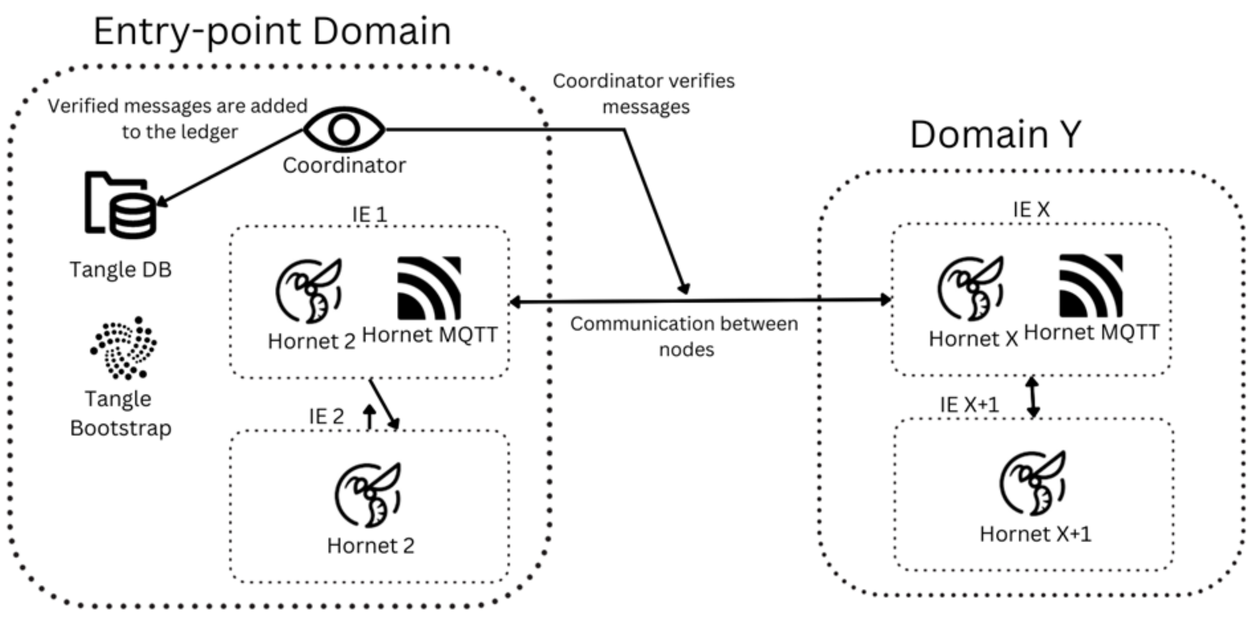


Figure 25. Planned IOTA prototype

With this done, the next steps involve further testing of the information that will be sent and the processes and pilots involved. In addition, a custom Helm chart has been created from scratch to be able to deploy IOTA in Kubernetes clusters, as packaging for K8s is not available yet in the official IOTA repository. After this the deployment will be moved into the computing resources provided by the partners, where it will be accessible for further integration.

3.6. Management services and aerOS management portal

The main aim of this task is the development of the aerOS Management Framework, which is composed by two independent components: the aerOS Management Portal and the aerOS Federator. These components are described separately in their respective subsections. In addition, the task T4.6 has started in M10 of the project, six months after the rest of WP3 and WP4 technical tasks. Thus, this task presents a strong dependence on them and some parts of it, for instance, Benchmarking tool and Entrypoint balancer, are in a more preliminary development stage.

3.6.1. aerOS Management Portal

The block schema of the aerOS Management Portal has been slightly modified when it comes to the interactions between the Backend and the aerOS Basic Services. In the first version described in D4.2, these interactions were described in a general way without going to the details and having the Entrypoint balancer as a mandatory pass-through component. Nevertheless, after performing some fine-tuning now it is clearer that currently 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 as the project advances. 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.

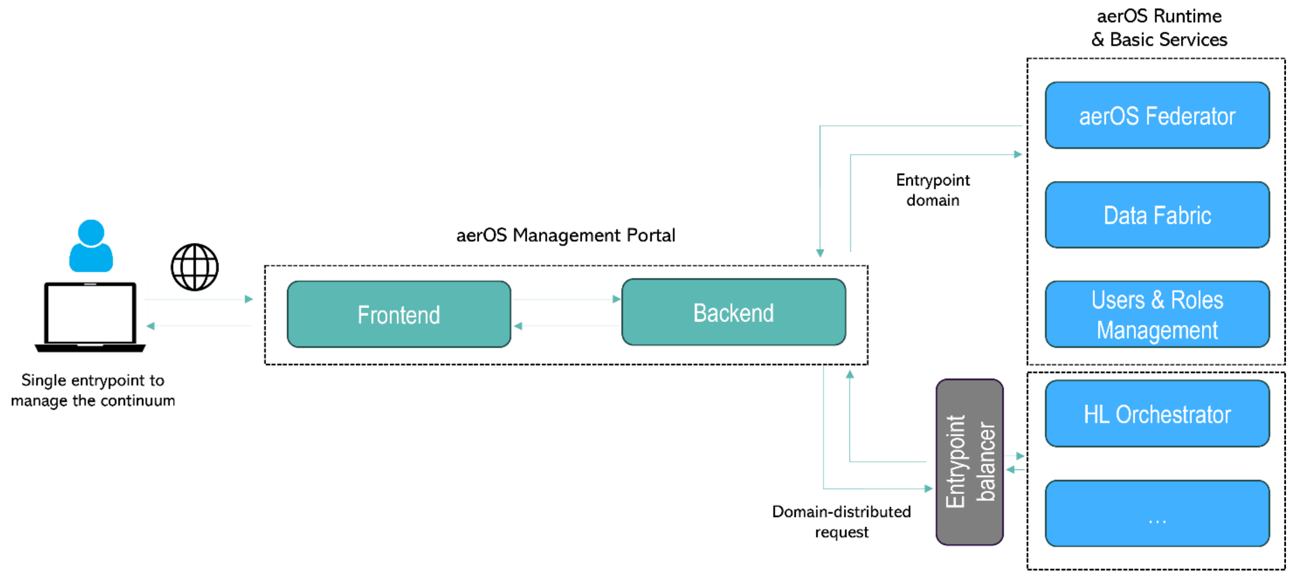


Figure 26. aerOS Management Portal architecture

T4.6 started in M10 of the project, so in D4.1 (M12) it was no possible to go through the technological details of the Management Portal components, apart from providing the list of candidate technologies to use in the development. Thus, the idea is to go component through component to provide more concise updates.

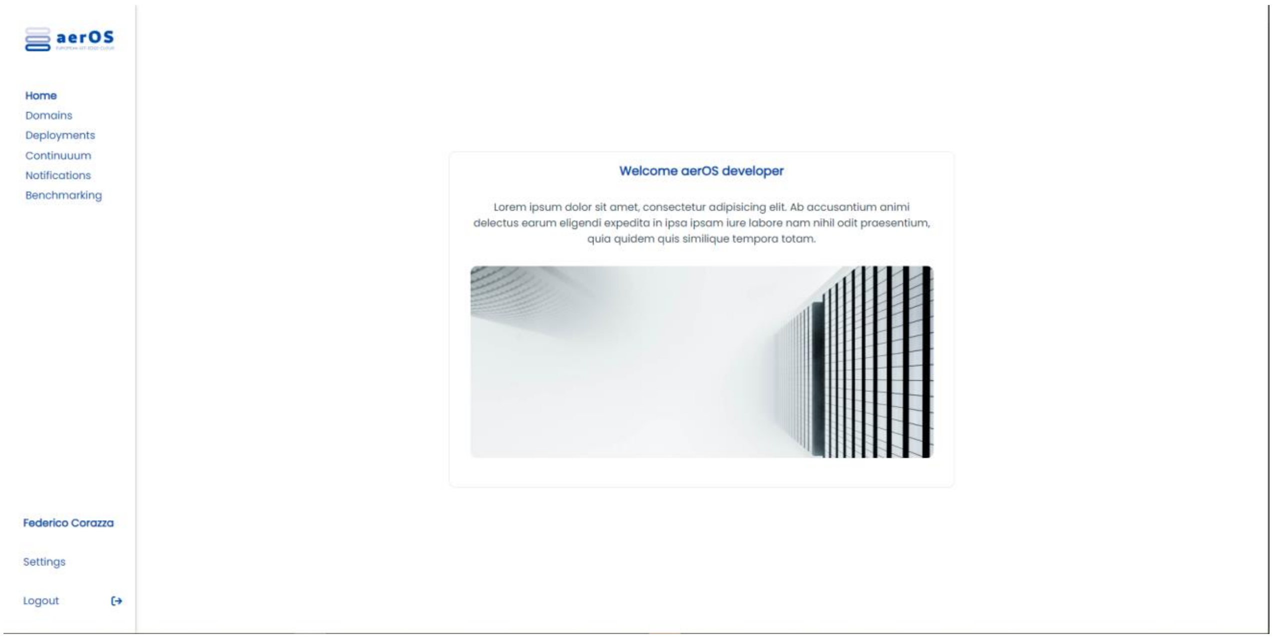
3.6.1.1. Frontend

Before developing the code for the Frontend part of the Management Portal, an analysis phase has been 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 will only show information and allow to perform actions in accordance with the role that has been assigned to the user in the aerOS AAA framework (see section 4.4.1.1 of D3.2 for more information about the defined roles in aerOS). For instance, only users with the proper rights will be able to initiate an orchestration request (e.g., a IoT service deployer) or check the status of the IEs that belong to his linked domains (e.g., Domain Administrator). Therefore, the aspect of the portal will change 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**



Welcome aerOS developer

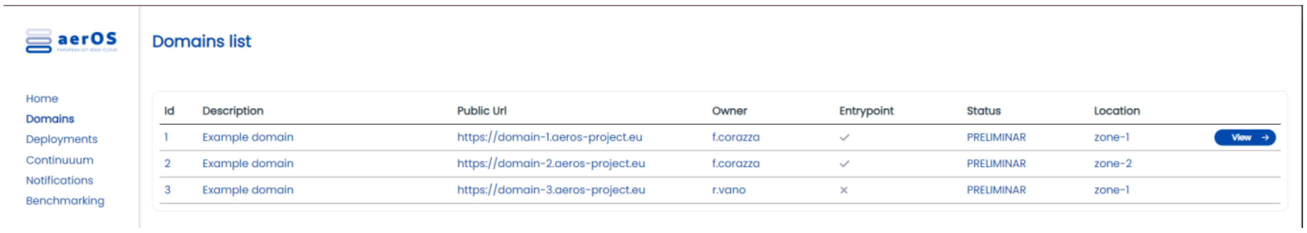
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ab accusantium animi delectus earum eligendi expedita in ipsa ipsam iure labore nam nihil odit praesentium, quia quidem quis similique tempora totam.

Figure 27. aerOS Management Portal welcome page and navigation menu

aerOS Meta-OS users land to the welcome page after performing the logging process interacting with Oauth2.0 protocol of Keycloak Identity Manager (IdM). 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.
  - 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).

- **Domains**



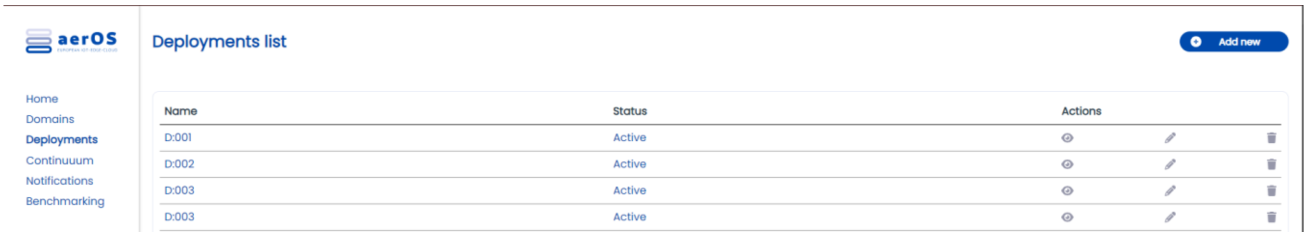
Domains list

Id	Description	Public Url	Owner	Entrypoint	Status	Location
1	Example domain	https://domain-1.aeros-project.eu	f.corazza	✓	PRELIMINAR	zone-1
2	Example domain	https://domain-2.aeros-project.eu	f.corazza	✓	PRELIMINAR	zone-2
3	Example domain	https://domain-3.aeros-project.eu	r.vano	×	PRELIMINAR	zone-1

Figure 28. aerOS Management Portal domains 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**



Deployments list

Name	Status	Actions		
D:001	Active	👁	✎	🗑
D:002	Active	👁	✎	🗑
D:003	Active	👁	✎	🗑
D:003	Active	👁	✎	🗑

Figure 29. aerOS Management Portal service deployments view

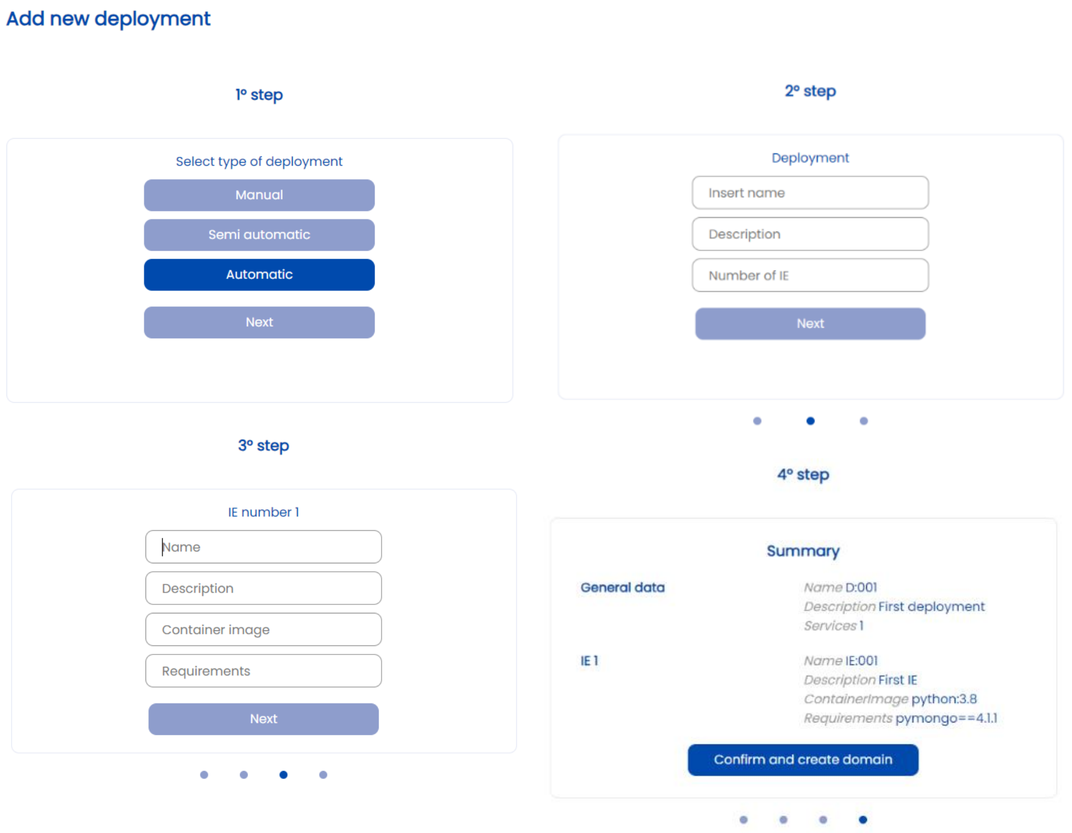
In the Deployments section, a table with a list of configured and performed deployments is shown, containing a list of the main attributes, such as description, status, and public URL. In the table there are two call-to-actions:

- *View*, by clicking the *eye* icon the user accesses to all the data related to the selected deployed service, including its service components.
- *Remove*, by clicking the *trash* can icon the user will be able to remove the selected deployed service.

Moreover, this view includes a button to request the orchestration of a IoT service in the continuum. By clicking on the top-right call-to-action, a wizard will appear to guide the user in the creation of the Intention Blueprint, which will be sent to an HLO to start the aerOS orchestration process. For this MVP release, a simple wizard has been developed, with the number of steps depending on the number of underlying service components (or containers) that compose the IoT service that the user wishes to deploy in the continuum.

The wizard is comprised of the steps depicted below:

1. The user can select among the offered orchestration flavours: (i) Manual, for selecting a specific IE; (ii) Semi-automatic, for selecting some open requirements (e.g., a set of IEs) to be taken into consideration by the orchestrator; and (iii) Automatic, to let the orchestrator decide the best fitting. At this point, only it is enabled the Automatic option to be aligned with the status of the aerOS orchestration.
2. In this step the user can add the main attributes of the deployment (Service in the aerOS continuum ontology, see section 3.1.3.1), such as name, description and number of underlying services that form the deployment (ServiceComponent in the ontology).
3. In this step the user completes the configuration of the service components: container image, orchestration requirements, etc. This stage is constantly being improved to be aligned with the orchestration process and the continuum ontology.
4. This last step shows a summary overview of the filled data, and, after a final check, the user can trigger the orchestration of the service.


Add new deployment

1º step

Select type of deployment

2º step

Deployment

3º step

IE number 1

4º step

Summary

General data — *Name* D:001, *Description* First deployment, *Services* 1

IE 1 — *Name* IE:001, *Description* First IE, *ContainerImage* python:3.8, *Requirements* pymongo==4.1.1

Figure 30. aerOS Management Portal new service deployment wizard

- **Continuum view**

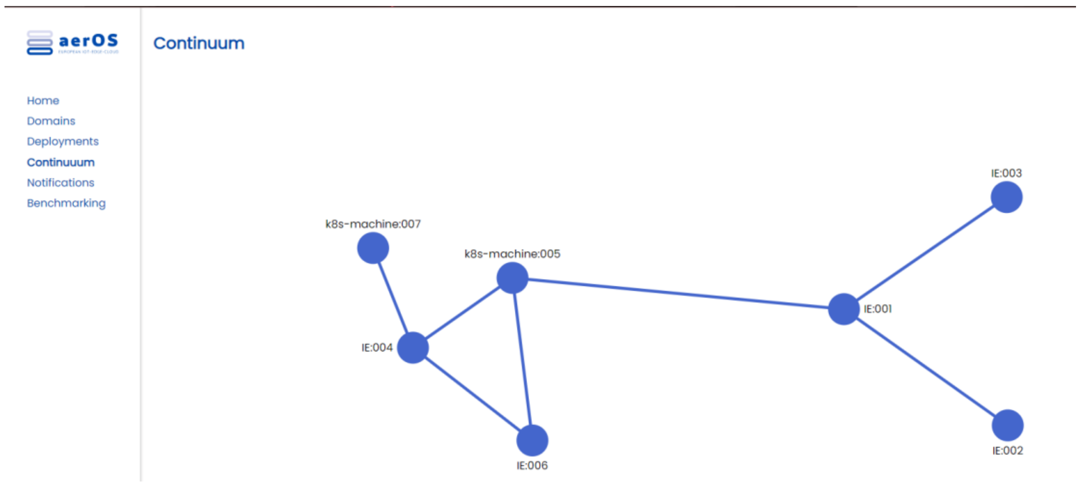


Figure 31. aerOS Management Portal continuum view

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.

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 accomplish 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 [10], [11], 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 [12], [13] 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) [14]) 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) [15], [16], which assigns tasks according to a circular list, or Opportunistic Load Balancing (OLB) [17], 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 [18]). The more complex ones are, for example, Resource-based Load Balanced Min-Min (RBLMM) 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 would be the LC (or one of its modifications). The proposed algorithm meets all established criteria and yet is simple enough that it would not require major changes in existing architecture. After the selection of LC algorithms, the next action points will be to develop and test a set of variations of LC algorithms, in a more mature state of aerOS composed of the components ready for the MVP, to select the one with best performance and a better fit in the aerOS continuum.

3.6.1.4. Benchmarking tool

The benchmarking tools are conceived to complement the aerOS Management Portal. These tools provide two main functionalities:

1) Benchmarking and comparison of IE/domains performance against known standards and methodologies such as TPCx-IOT [19] and RFC 2544 [20].
2) Internal Technical KPIs dashboard, which provides a snapshot of these KPIs defined for aerOS technical components in deliverable D5.2 and which are directly measurable from the aerOS stack.

In both cases, these tools will ingest data from the Data Fabric, meaning that at least integration with the Orion-LD broker is necessary, as well as support for semantics defining all the data to be used. The following diagram sketches the relationship between these tools and other services deployed with the aerOS stack.

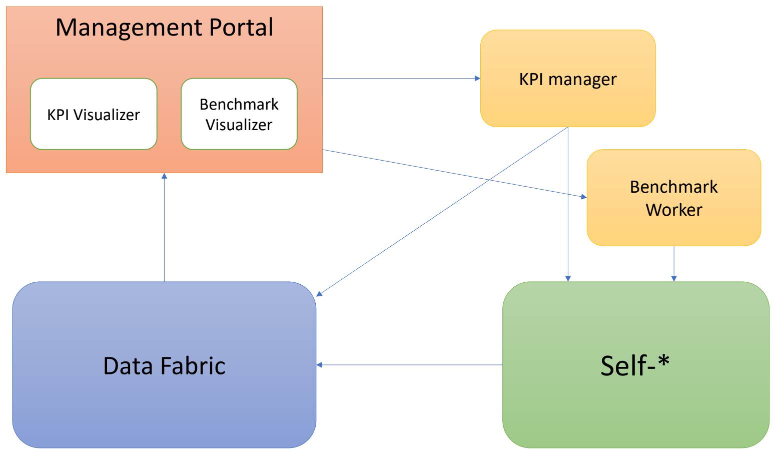


Figure 32. Benchmark tools architecture

Using as data source the data of the Data Fabric already inserted by Self-* modules and additional queries to the Self-API to gather additional information, the Management Portal will be extended with two components to visualize both technical KPIs (KPI visualizer) and benchmark results (Benchmark visualizer).

Since data needs to be prepared before the visualization, two backend workers, “KPI Manager” and “Benchmark Worker”, will periodically gather source data from the different IEs and insert in the Data Fabric the information to be displayed by the Management Portal. Finally, it will be explored if these workers will be included as part of the backend component of the portal.

3.6.1.5. Candidate technologies and standards

Table 8. Candidate technologies and standards for aerOS Management Portal

Technology/ Standard	Description	Justification
Vue.js	Vue.js is an open-source JavaScript framework for	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

	building web user interfaces.	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.
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 is a strong candidate to show this information following a network graph visualization.
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 defacto 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.
Least connection 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.

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. This capability is 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 only focused on the data related with the management of the continuum, which follows the ontology introduced in section 3.1.3.1. It also acts as the building block for the aerOS distributed domain repository. Therefore, the main efforts for the MVP have been put in testing Orion-LD federation to deliver a methodology to create the needed Context Source Registrations to achieve domains federation, always having into consideration the aerOS ontology for the IoT-Edge-Cloud computing continuum. Without this work, it will not be possible to federate aerOS domains in a secure and resilient way.

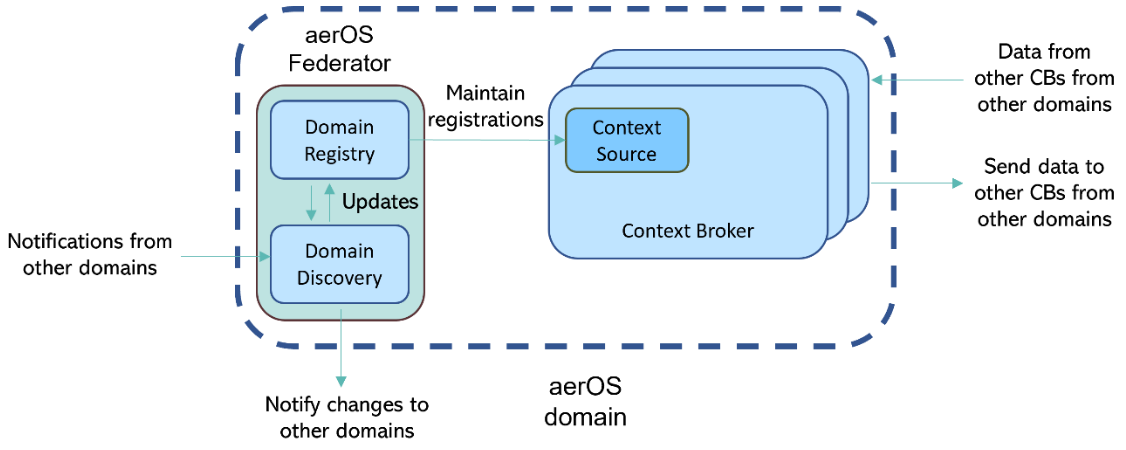
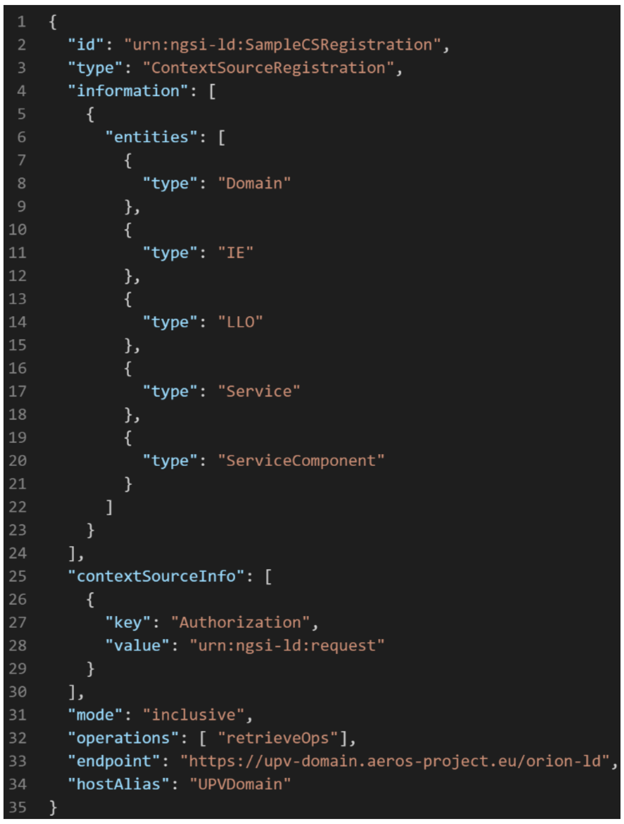

aerOS
Federator
Domain Registry
Updates
Domain Discovery
Maintain registrations
Context Source
Context Broker
Data from other CBs from other domains
Send data to other CBs from other domains
Notifications from other domains
Notify changes to other domains
aerOS domain

Figure 33. 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.



```json
{
  "id": "urn:ngsi-ld:SampleCSRegistration",
  "type": "ContextSourceRegistration",
  "information": [
    {
      "entities": [
        {
          "type": "Domain"
        },
        {
          "type": "IE"
        },
        {
          "type": "LLO"
        },
        {
          "type": "Service"
        },
        {
          "type": "ServiceComponent"
        }
      ]
    }
  ],
  "contextSourceInfo": [
    {
      "key": "Authorization",
      "value": "urn:ngsi-ld:request"
    }
  ],
  "mode": "inclusive",
  "operations": [ "retrieveOps"],
  "endpoint": "https://upv-domain.aeros-project.eu/orion-ld",
  "hostAlias": "UPVDomain"
}
```

Figure 34. aerOS Federator NGSI-LD Context Source Registration example

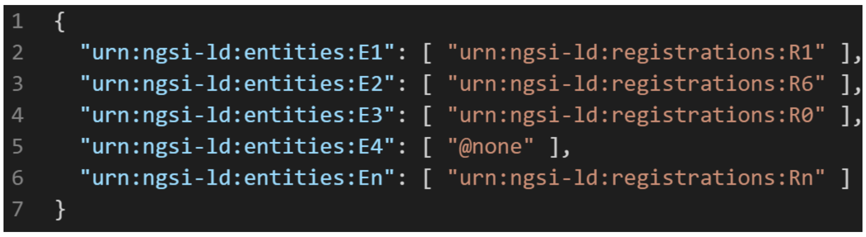
Regarding the code development of Orion-LD in the scope of aerOS, queries for retrieving contextual data as NGSI-LD entities actually stored in different context brokers play a main role in the aerOS federation. Thus, in the domain of distributed systems, particularly within the context of entity federation, 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 has been adopted for 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. This approach is slated for inclusion in the forthcoming v1.8.1 of the NGSI-LD API specification, expected in March 2024, and is already being incorporated into Orion-LD version 1.5.1. The Snapshots method remains under discussion in ETSI ISG CIM and is anticipated to be part of NGSI-LD API version 1.9.1, targeted for release in September 2024.

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 35. Orion-LD Entity Map example

3.6.2.2. aerOS Federator custom component

aerOS Federator custom component depends on the work described above. This 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). 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. For that reason, gRPC stands as the most promising option compared to traditional REST APIs. gRPC is an open-source implementation of remote procedure calls (RPC) developed by Google, which uses a binary protocol for data exchange over HTTP/2 and implements bi-directional communications. Moreover, messages are defined following Protocol Buffers (Protobuf) that will also be used for asynchronous exchange of messages among aerOS orchestrator microservices (see section 4.3.2 of D3.2). The design of this component follows the microservices pattern, which allows to split traditional monolithic applications into different small pieces of software with a particular and dedicated functionality. Finally, as explained before, the main efforts for the MVP have been put in the Orion-LD component, so more detailed description of the custom aerOS Federator component will be provided in next deliverables.

3.6.2.3. Candidate technologies and standards

Table 9. Candidate 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 the testing of these new functionalities will be performed 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 will provide a layer of automatization and fault tolerance mechanisms on top of Orion-LD. By using a microservices architecture, the development of this component will be 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.

4. Conclusions and future work

This deliverable has presented the intermediate release of the WP4 software components for delivering intelligence at the edge. The document has provided an overview of the aerOS MVP from the standpoint of WP4, detailing how each of the building blocks has contributed to the realization of core features of the aerOS MVP such as the intelligent orchestration of the continuum.

Regarding data homogenization, on the one hand, the Semantic Annotator components have been integrated into the aerOS ecosystem, extended with capabilities to accommodate NGSI-LD. Additionally, the Semantic Translator has been ported Scala 3, and undergone several improvements. Lastly, the LOT methodology for ontology development has been introduced in the project and applied during the creation of the aerOS continuum 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 future release of the Data Fabric will focus on the data security and data catalogue features.

When it comes to decentralized frugal AI, the AI Local Executor and AI Task Controller components, including the FL Training Collector and FL Repository, have been implemented for AI workflow management. In addition, research on explainability methods for an aerOS use case based on reinforcement learning has been conducted, with an approach designed and implemented. Requirements for AI Explainability Service input data were collected and analysed.

A first release of the Embedded Analytics Engine has been implemented, including 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. In future releases, the Embedded Analytics Engine will be integrated with the aerOS Data Fabric and other components of the aerOS stack.

The future actions for finalizing the development of the aerOS Trust Management component include the identification of how each attribute affects the trust of IEs to define the appropriate weight for the trust score calculation algorithm. Afterwards, the implementation of the trust score calculation function will take place in the IEs of the continuum while the final version of the Trust Management is expected to be presented in D4.3.

Finally, the management services of aerOS have made notable progress, including a first release of the aerOS management for registration of new aerOS domains, as well as with the implementation of the domain federation mechanism and the extended capabilities of the Orion-LD for sharing data across aerOS domains.

The final release of the WP4 components, incorporating extensions delivered in months M19-M30 will be documented in D4.3, targeted for month M30 of the project.

References

[1] 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.

[2] 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.

[3] ‘BIMERR Ontologies’. Accessed: Feb. 18, 2024. [Online]. Available: https://bimerr.iot.linkeddata.es/

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

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

[6] M. D. Wilkinson *et al.*, ‘The FAIR Guiding Principles for scientific data management and stewardship’, *Sci. Data*, vol. 3, no. 1, p. 160018, Mar. 2016, doi: 10.1038/sdata.2016.18.

[7] J. Arenas-Guerrero, D. Chaves-Fraga, J. Toledo, M. S. Pérez, and O. Corcho, ‘Morph-KGC: Scalable knowledge graph materialization with mapping partitions’, *Semantic Web*, pp. 1–20, Aug. 2022, doi: 10.3233/SW-223135.

[8] ‘PROV-O: The PROV Ontology’. Accessed: Feb. 02, 2024. [Online]. Available: https://www.w3.org/TR/prov-o/

[9] ‘The Organization Ontology’. Accessed: Feb. 18, 2024. [Online]. Available: https://www.w3.org/TR/vocab-org/

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] ‘TPCx-IoT Homepage’. Accessed: Feb. 18, 2024. [Online]. Available: https://www.tpc.org/tpcx-iot/

[20] ‘Benchmarking Methodology for Network Interconnect Devices’, Internet Engineering Task Force, Request for Comments RFC 2544, Mar. 1999. doi: 10.17487/RFC2544.

A. Complete workflow of LOT methodology

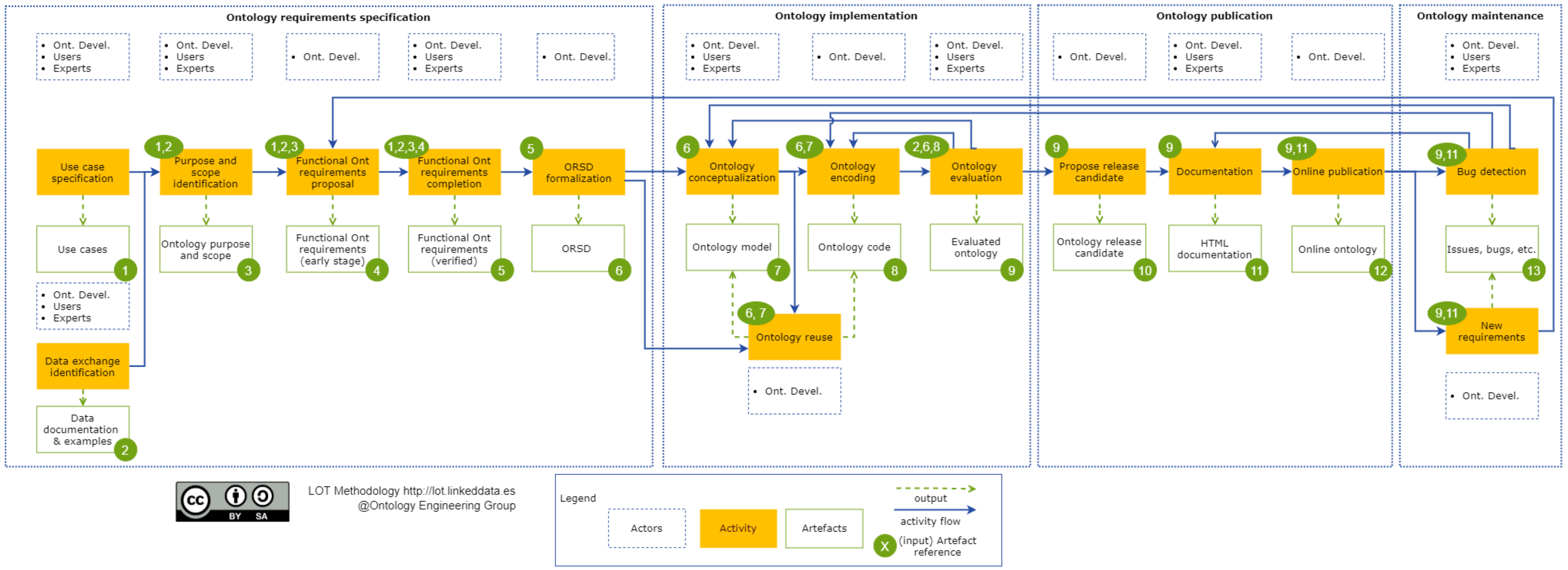

Ontology requirements specification
Ontology implementation
Ontology publication
Ontology maintenance
Ont. Devel.
Users
Experts
Use case specification
Purpose and scope identification
Functional Ont requirements proposal
Functional Ont requirements completion
ORSD formalization
Use cases
Ontology purpose and scope
Functional Ont requirements (early stage)
Functional Ont requirements (verified)
ORSD
Data exchange identification
Data documentation & examples
Ontology conceptualization
Ontology encoding
Ontology evaluation
Ontology model
Ontology code
Evaluated ontology
Ontology reuse
Propose release candidate
Documentation
Online publication
Ontology release candidate
HTML documentation
Online ontology
Bug detection
Issues, bugs, etc.
New requirements
LOT Methodology http://lot.linkeddata.es
@Ontology Engineering Group
Legend
Actors
Activity
Artefacts
output
activity flow
(input) Artefact reference

Figure 36. Detailed view of the complete LOT methodology. Source [1]

B. Published aerOS continuum methodology

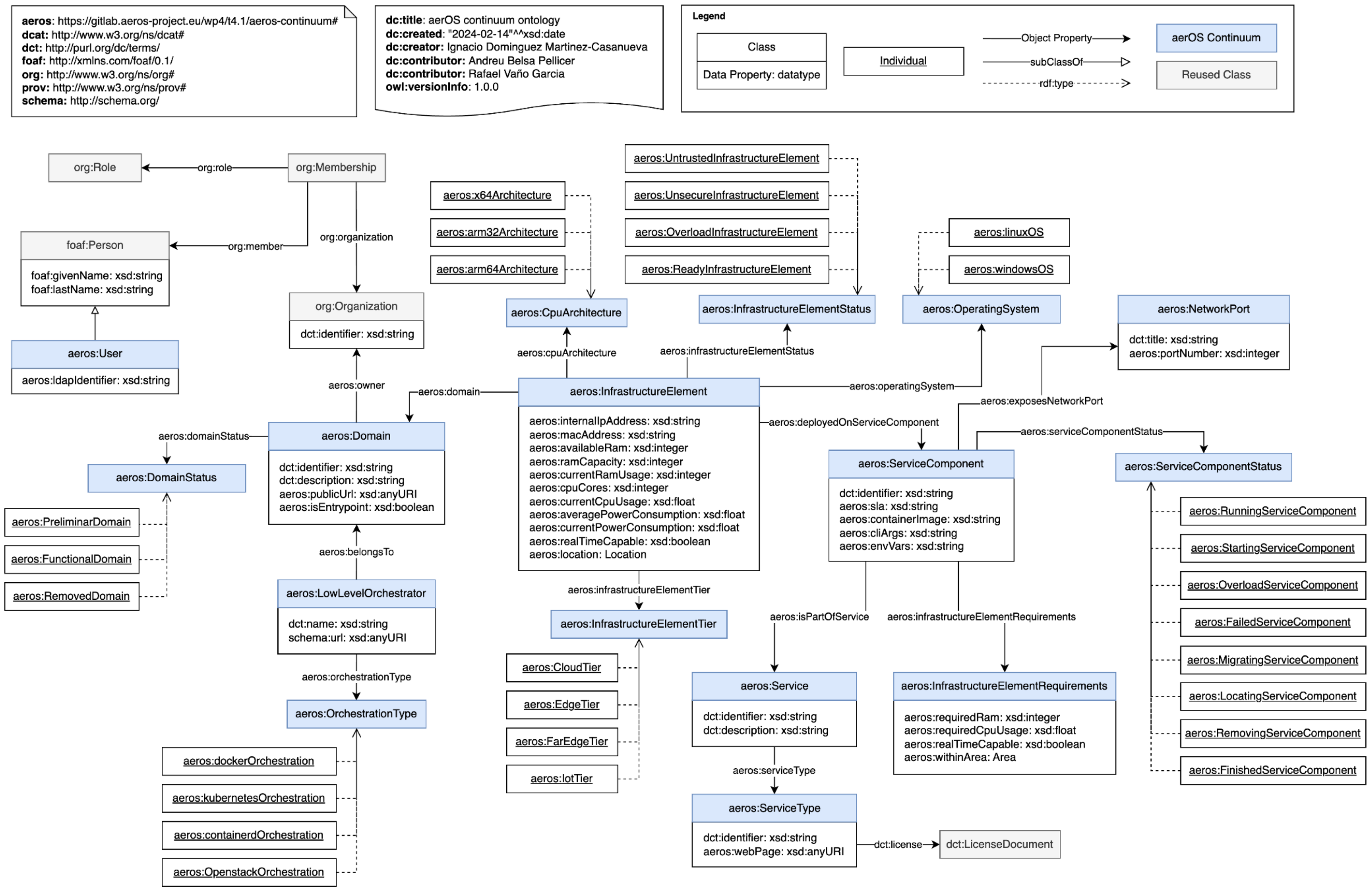
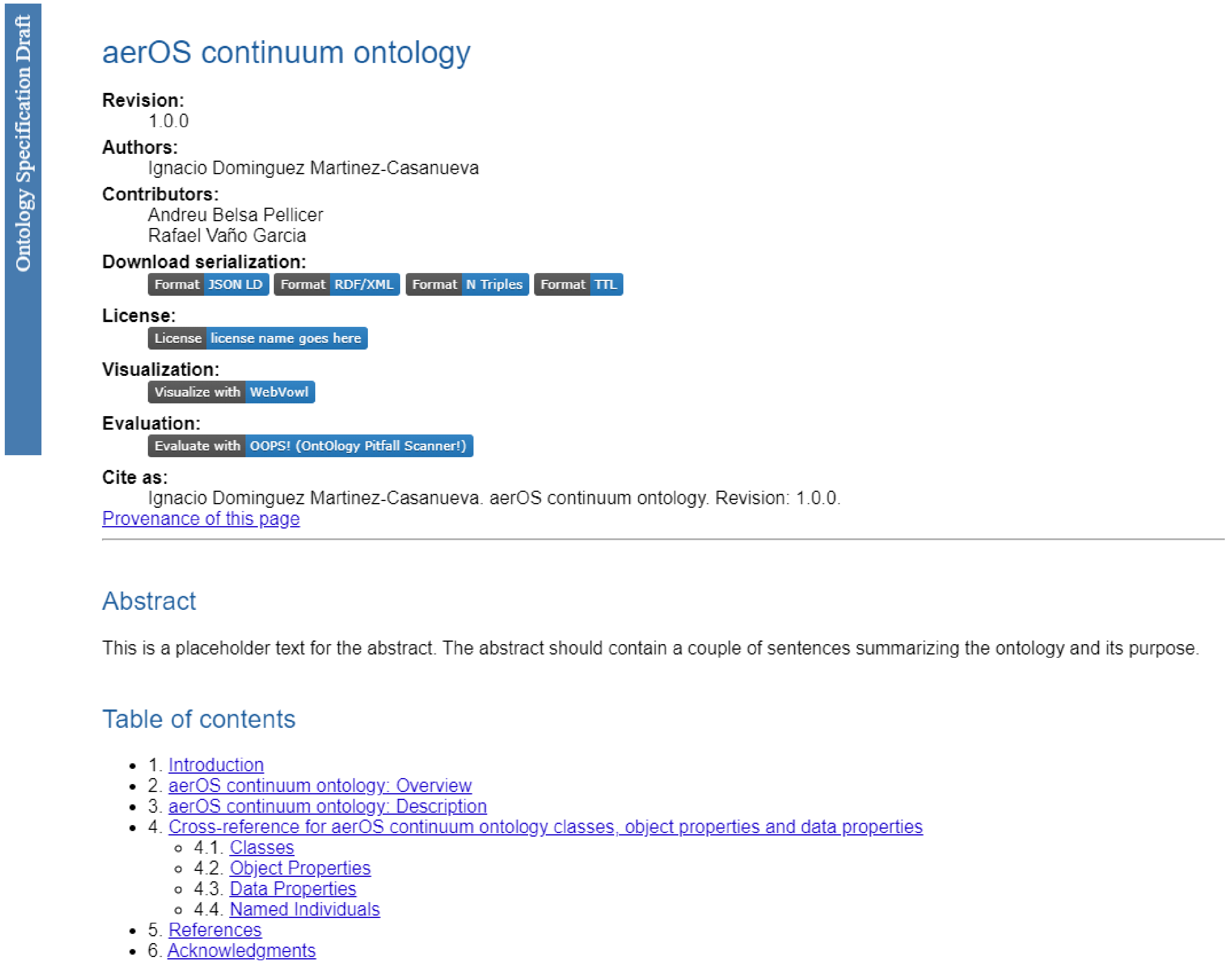

aeros: https://gitlab.aeros-project.eu/wp4/t4.1/aeros-continuum#
dcat: http://www.w3.org/ns/dcat#
dct: http://purl.org/dc/terms/
foaf: http://xmlns.com/foaf/0.1/
org: http://www.w3.org/ns/org#
prov: http://www.w3.org/ns/prov#
schema: http://schema.org/
dc:title: aerOS continuum ontology
dc:created: "2024-02-14"^^xsd:date
dc:creator: Ignacio Dominguez Martinez-Casanueva
dc:contributor: Andreu Belsa Pellicer
dc:contributor: Rafael Vaño Garcia
owl:versionInfo: 1.0.0
Legend
Class
Data Property: datatype
Individual
Object Property
subClassOf
rdf:type
aerOS Continuum
Reused Class
org:Role
org:role
org:Membership
foaf:Person
org:member
foaf:givenName: xsd:string
foaf:lastName: xsd:string
org:organization
org:Organization
dct:identifier: xsd:string
aeros:User
aeros:ldapIdentifier: xsd:string
aeros:owner
aeros:domain
aeros:domainStatus
aeros:Domain
dct:identifier: xsd:string
dct:description: xsd:string
aeros:publicUrl: xsd:anyURI
aeros:isEntrypoint: xsd:boolean
aeros:DomainStatus
aeros:PreliminarDomain
aeros:FunctionalDomain
aeros:RemovedDomain
aeros:belongsTo
aeros:LowLevelOrchestrator
dct:name: xsd:string
schema:url: xsd:anyURI
aeros:orchestrationType
aeros:OrchestrationType
aeros:dockerOrchestration
aeros:kubernetesOrchestration
aeros:containerdOrchestration
aeros:OpenstackOrchestration
aeros:x64Architecture
aeros:arm32Architecture
aeros:arm64Architecture
aeros:CpuArchitecture
aeros:cpuArchitecture
aeros:UntrustedInfrastructureElement
aeros:UnsecureInfrastructureElement
aeros:OverloadInfrastructureElement
aeros:ReadyInfrastructureElement
aeros:InfrastructureElementStatus
aeros:infrastructureElementStatus
aeros:linuxOS
aeros:windowsOS
aeros:OperatingSystem
aeros:NetworkPort
dct:title: xsd:string
aeros:portNumber: xsd:integer
aeros:operatingSystem
aeros:exposesNetworkPort
aeros:InfrastructureElement
aeros:internalIpAddress: xsd:string
aeros:macAddress: xsd:string
aeros:availableRam: xsd:integer
aeros:ramCapacity: xsd:integer
aeros:currentRamUsage: xsd:integer
aeros:cpuCores: xsd:integer
aeros:currentCpuUsage: xsd:float
aeros:averagePowerConsumption: xsd:float
aeros:currentPowerConsumption: xsd:float
aeros:realTimeCapable: xsd:boolean
aeros:location: Location
aeros:deployedOnServiceComponent
aeros:serviceComponentStatus
aeros:ServiceComponent
dct:identifier: xsd:string
aeros:sla: xsd:string
aeros:containerImage: xsd:string
aeros:cliArgs: xsd:string
aeros:envVars: xsd:string
aeros:ServiceComponentStatus
aeros:RunningServiceComponent
aeros:StartingServiceComponent
aeros:OverloadServiceComponent
aeros:FailedServiceComponent
aeros:MigratingServiceComponent
aeros:LocatingServiceComponent
aeros:RemovingServiceComponent
aeros:FinishedServiceComponent
aeros:infrastructureElementTier
aeros:InfrastructureElementTier
aeros:CloudTier
aeros:EdgeTier
aeros:FarEdgeTier
aeros:IotTier
aeros:isPartOfService
aeros:infrastructureElementRequirements
aeros:Service
dct:identifier: xsd:string
dct:description: xsd:string
aeros:InfrastructureElementRequirements
aeros:requiredRam: xsd:integer
aeros:requiredCpuUsage: xsd:float
aeros:realTimeCapable: xsd:boolean
aeros:withinArea: Area
aeros:serviceType
aeros:ServiceType
dct:identifier: xsd:string
aeros:webPage: xsd:anyURI
dct:license
dct:LicenseDocument

Figure 37. Conceptual model for the aerOS continuum ontology.



Ontology Specification Draft

aerOS continuum ontology

Revision:
1.0.0

Authors:
Ignacio Dominguez Martinez-Casanueva

Contributors:
Andreu Belsa Pellicer
Rafael Vaño Garcia

Download serialization:
Format JSON LD | Format RDF/XML | Format N Triples | Format TTL

License:
License license name goes here

Visualization:
Visualize with WebVowl

Evaluation:
Evaluate with OOPS! (OntOlogy Pitfall Scanner!)

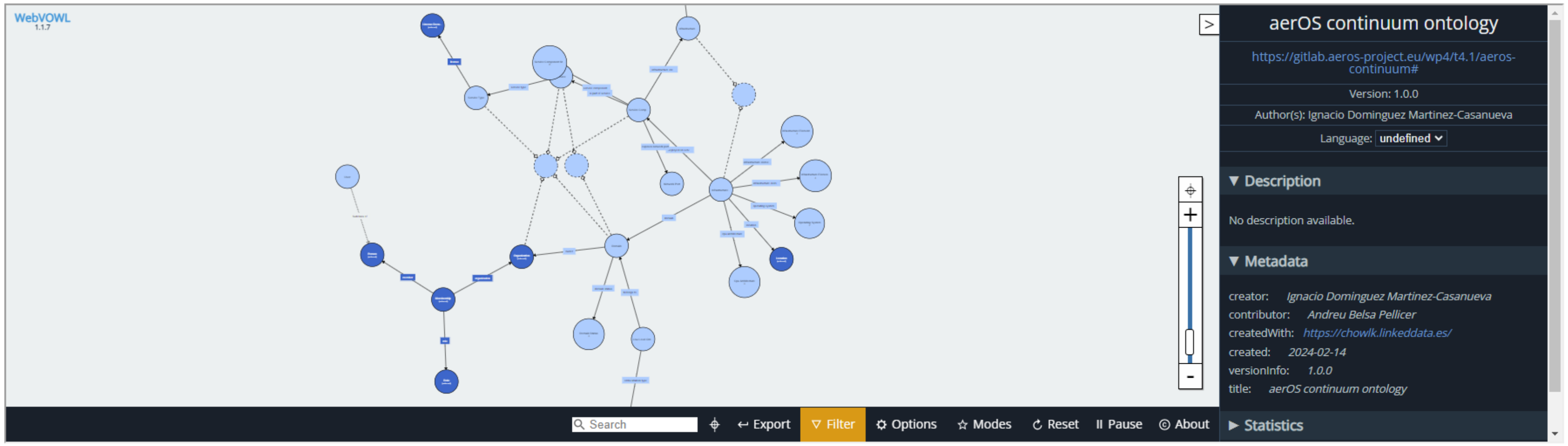
Cite as:
Ignacio Dominguez Martinez-Casanueva. aerOS continuum ontology. Revision: 1.0.0.
Provenance of this page

Abstract

This is a placeholder text for the abstract. The abstract should contain a couple of sentences summarizing the ontology and its purpose.

Table of contents

Figure 38. Published aerOS continuum ontology



aerOS continuum ontology

https://gitlab.aeros-project.eu/wp4/t4.1/aeros-continuum#

Version: 1.0.0

Author(s): Ignacio Dominguez Martinez-Casanueva

Language: undefined

▼ Description

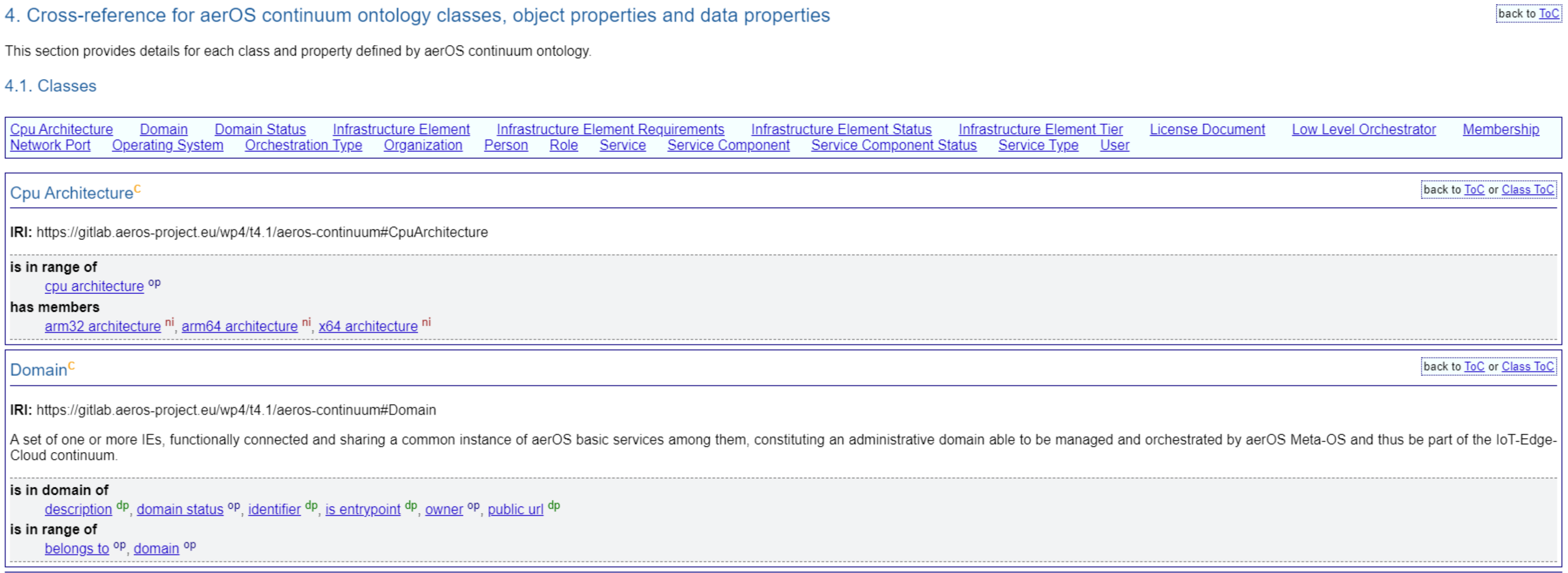
No description available.

▼ Metadata

creator: *Ignacio Dominguez Martinez-Casanueva*
contributor: *Andreu Belsa Pellicer*
createdWith: *https://chowlk.linkeddata.es/*
created: *2024-02-14*
versionInfo: *1.0.0*
title: *aerOS continuum ontology*

▶ Statistics

Figure 39. Neural network graph of the published aerOS ontology



4. Cross-reference for aerOS continuum ontology classes, object properties and data properties

back to ToC

This section provides details for each class and property defined by aerOS continuum ontology.

4.1. Classes

Cpu Architecture   Domain   Domain Status   Infrastructure Element   Infrastructure Element Requirements   Infrastructure Element Status   Infrastructure Element Tier   License Document   Low Level Orchestrator   Membership   Network Port   Operating System   Orchestration Type   Organization   Person   Role   Service   Service Component   Service Component Status   Service Type   User

Cpu Architecture^c

back to ToC or Class ToC

IRI: https://gitlab.aeros-project.eu/wp4/t4.1/aeros-continuum#CpuArchitecture

is in range of
cpu architecture ^op
has members
arm32 architecture ^ni, arm64 architecture ^ni, x64 architecture ^ni

Domain^c

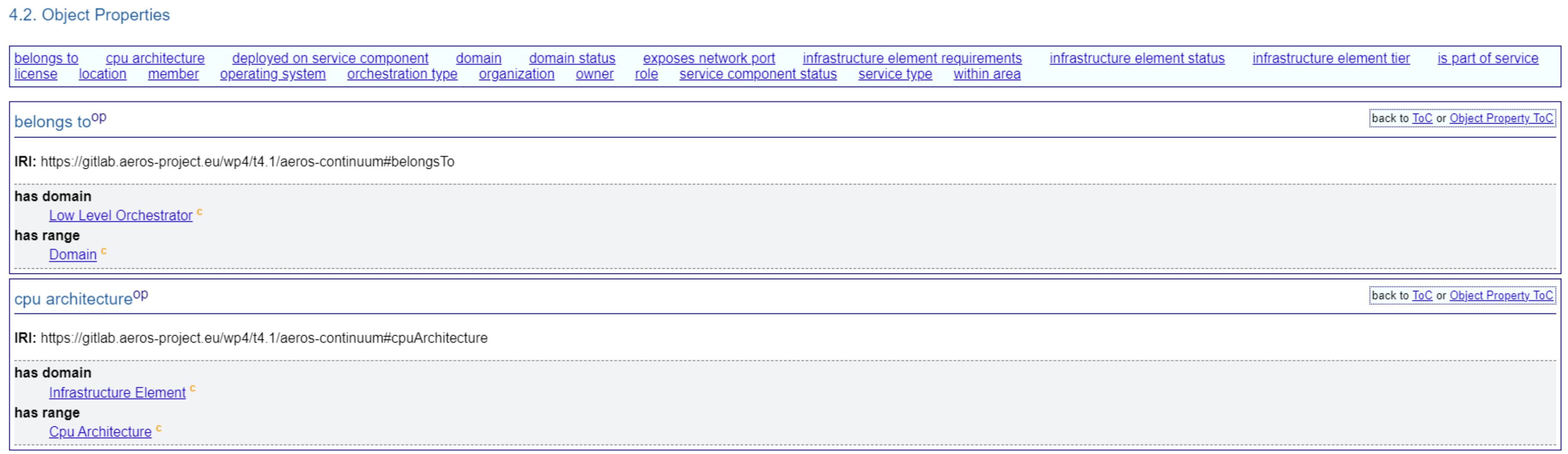
back to ToC or Class ToC

IRI: https://gitlab.aeros-project.eu/wp4/t4.1/aeros-continuum#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.

is in domain of
description ^dp, domain status ^op, identifier ^dp, is entrypoint ^dp, owner ^op, public url ^dp
is in range of
belongs to ^op, domain ^op

Figure 40. Class definition of the published aerOS continuum ontology



4.2. Object Properties

belongs to cpu architecture deployed on service component domain domain status exposes network port infrastructure element requirements infrastructure element status infrastructure element tier is part of service license location member operating system orchestration type organization owner role service component status service type within area

belongs to^{op} — back to ToC or Object Property ToC

IRI: https://gitlab.aeros-project.eu/wp4/t4.1/aeros-continuum#belongsTo

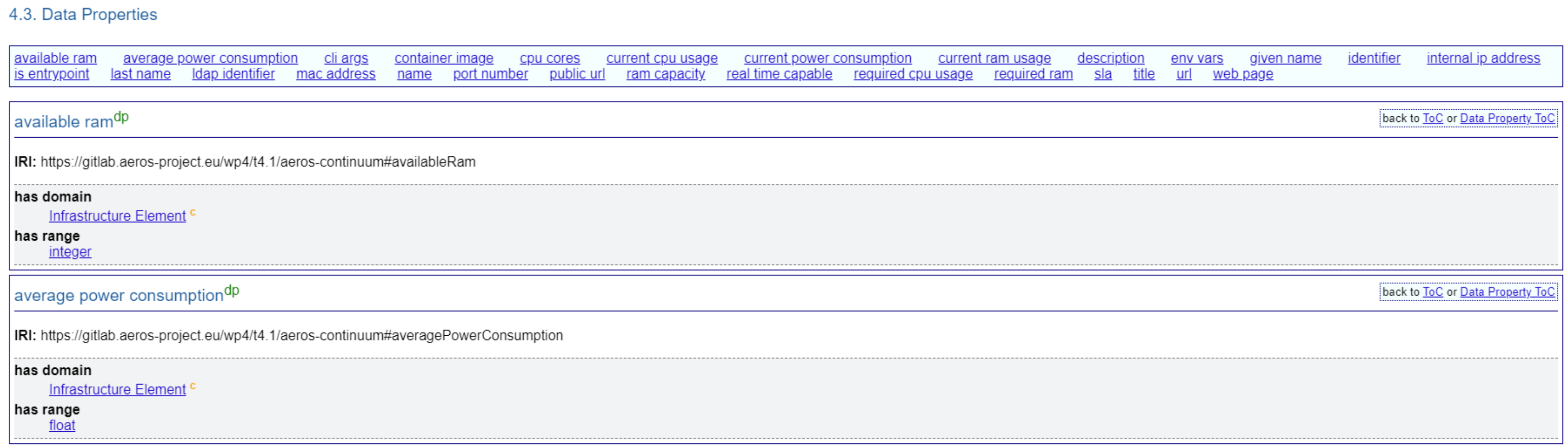
has domain
Low Level Orchestrator c
has range
Domain c

cpu architecture op — back to ToC or Object Property ToC

IRI: https://gitlab.aeros-project.eu/wp4/t4.1/aeros-continuum#cpuArchitecture

has domain
Infrastructure Element c
has range
Cpu Architecture c

Figure 41. Object properties of the published aerOS continuum ontology



4.3. Data Properties

available ram average power consumption cli args container image cpu cores current cpu usage current power consumption current ram usage description env vars given name identifier internal ip address is entrypoint last name ldap identifier mac address name port number public url ram capacity real time capable required cpu usage required ram sla title url web page

available ram dp — back to ToC or Data Property ToC

IRI: https://gitlab.aeros-project.eu/wp4/t4.1/aeros-continuum#availableRam

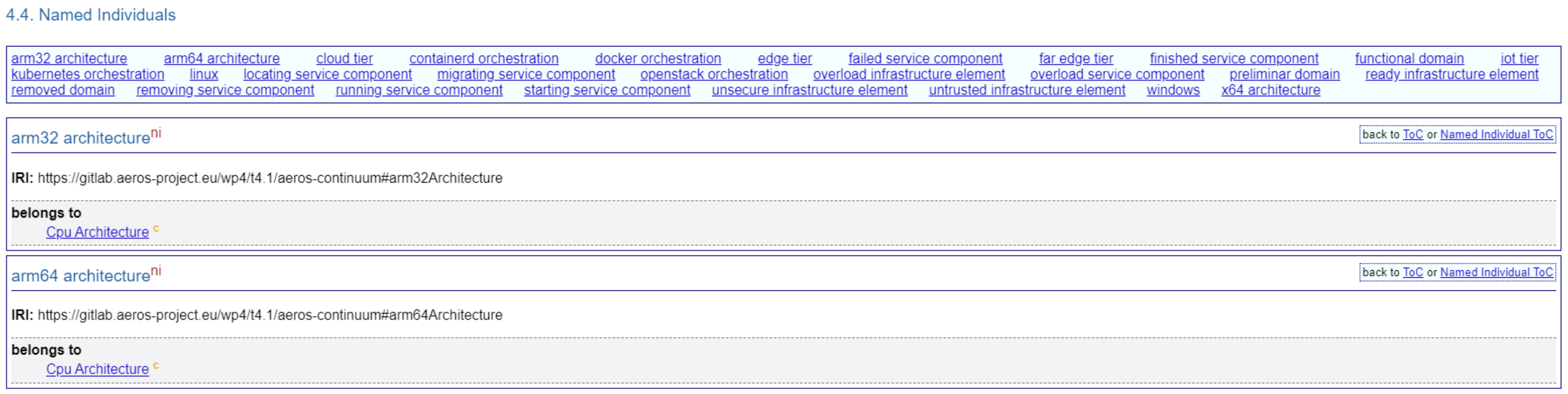
has domain
Infrastructure Element c
has range
integer

average power consumption dp — back to ToC or Data Property ToC

IRI: https://gitlab.aeros-project.eu/wp4/t4.1/aeros-continuum#averagePowerConsumption

has domain
Infrastructure Element c
has range
float

Figure 42. Data properties of the published aerOS continuum ontology



4.4. Named Individuals

arm32 architecture | arm64 architecture | cloud tier | containerd orchestration | docker orchestration | edge tier | failed service component | far edge tier | finished service component | functional domain | iot tier | kubernetes orchestration | linux | locating service component | migrating service component | openstack orchestration | overload infrastructure element | overload service component | preliminar domain | ready infrastructure element | removed domain | removing service component | running service component | starting service component | unsecure infrastructure element | untrusted infrastructure element | windows | x64 architecture

arm32 architecture ni

back to ToC or Named Individual ToC

IRI: https://gitlab.aeros-project.eu/wp4/t4.1/aeros-continuum#arm32Architecture

belongs to

Cpu Architecture c

arm64 architecture ni

back to ToC or Named Individual ToC

IRI: https://gitlab.aeros-project.eu/wp4/t4.1/aeros-continuum#arm64Architecture

belongs to

Cpu Architecture c

Figure 43. Named individuals of the published aerOS continuum ontology