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

aerOS

EUROPEAN IOT-EDGE-CLOUD

D4.1 - Software for delivering intelligence at the edge preliminary release

Deliverable No.	**D4.1**	**Due Date**	**31-AUG-2023**
Type	**Other**	**Dissemination Level**	**Public**
Version	**1.0**	**WP**	**WP4**
Description	*Initial software components, relationships, building blocks in relationship with 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
COSMOTE KINITES TILEPIKOINONIES AE	EL
EIGHT BELLS LTD	CY
INQBIT INNOVATIONS SRL	RO
FOGUS INNOVATIONS & SERVICES P.C.	EL
L.M. ERICSSON LIMITED	IE
SYSTEMS RESEARCH INSTITUTE OF THE POLISH ACADEMY OF SCIENCES IBS PAN	PL
ICTFICIAL OY	FI
INFOLYSIS P.C.	EL
PRODEVELOP SL	ES
EUROGATE CONTAINER TERMINAL LIMASSOL LIMITED	CY
TECHNOLOGIKO PANEPISTIMIO KYPROU	CY
DS TECH SRL	IT
GRUPO S 21SEC GESTION SA	ES
JOHN DEERE GMBH & CO. KG*JD	DE
CLOUDFERRO SP ZOO	PL
ELECTRUM SP ZOO	PL
POLITECNICO DI MILANO	IT
MADE SCARL	IT
NAVARRA DE SERVICIOS Y TECNOLOGIAS SA	ES
SWITZERLAND INNOVATION PARK BIEL/BIENNE AG	CH

Disclaimer

Authors

Name	Partner	e-mail
Prof. Carlos E. Palau Ignacio Lacalle Úbeda Rafael Vaño Andreu Belsa Pellicer Raúl San Julián Salvador Cuñat	P01 UPV	caplau@upv.es iglaub@upv.es ravagar2@upv.es anbelpel@upv.es rausanga@upv.es salcuane@upv.es
Harilaos Koumaras Vasilis Pitsilis George Makropoulos Anastasios Gogos Constantinos Vassilakis Thanos Papakyriakou Dimitris Davazoglou	P02 NCSRD	koumaras@iit.demokritos.gr vpitsilis@iit.demokritos.gr gmakropoulos@iit.demokritos.gr angogos@iit.demokritos.gr cvassilakis@iit.demokritos.gr thpap@iit.demokritos.gr d.davazoglou@iit.demokritos.gr
Ignacio Dominguez Martinez-Casanueva	P07 TID	ignacio.dominguezmartinez@telefonica.com
Ioannis Makropodis Panagiotis Bountakas Georgios Petichakis	P10 IQB	giannis.makropodis@inqbit.io panagiotis.bpountakas@inqbit.io giorgos.petihakis@inqbit.io
Joseph McNamara	P12 LMI	joseph.mcnamara@ericsson.com
Katarzyna Wasielewska-Michniewska Wiesław Pawłowski Przemysław Hołda Kajetan Rachwał	P13 SRIPAS	Katarzyna.wasielewska@ibspan.waw.pl Wieslaw.Pawlowski@ibspan.waw.pl Przemyslaw.Holda@ibspan.waw. pl Kajetan.Rachwal@ibspan.waw.pl
Nikolaos Gkatzios George Koumaras	P15 INF	ngkatzios@infolysis.gr gkoumaras@infolysis.gr

History

Date	Version	Change
14-JUN-2023	0.1	Initial ToC
30-JUN-2023	0.2	First round of contributions
20-JUL-2023	0.5	Second round of contributions
21-JUL-2023	0.9	Version ready for internal review
4-AGO-2023	1.0	Final submitted version

Key Data

Keywords	IoT, aerOS, data homogenization, data governance, AI, analytics, trustworthiness, management services
Lead Editor	P13 SRIPAS – Katarzyna Wasielewska-Michniewska
Internal Reviewer(s)	Alexander Wagner, P21 JD , Michalis Michaelides, P18 CUT

Executive Summary

This document is the first of three deliverables summarizing the outcomes of WP4 which, along with WP3, comprise the two technical work packages in aerOS project. D4.1 presents results at the M12 of the aerOS project execution, (8 months deep of WP's timeframe). In terms of proposed solutions, this deliverable exposes the technical approaches selected in the various areas covered by the different WP4 tasks. The WP4 tasks cover the following aspects of aerOS' meta Operating System (metaOS) for the computing continuum: (1) data autonomy of homogenization, semantic interoperability; (2) data governance, traceability, provenance and lineage policy engine; (3) decentralized frugal AI; (4) real-time embedded multiplane analytics; (5) trustworthiness; (6) management services and aerOS management portal. For each task a specification/design of the solution is described according to a common structure including: description and main functionalities, mapping onto requirements, structure diagram of the solution, candidate technologies foreseen for the realization.

For data autonomy of homogenization and semantic interoperability, a solution based on semantic annotation and translation has been proposed. Semantic annotation will allow to ingest "raw data" from external sources and enrich it with appropriate semantic information, to enable further processing. Semantic translation will offer a highly-scalable, efficient translation architecture, with two main interfaces: REST and reactive streaming.

Moreover, the aerOS Data Fabric shall include mechanisms supporting data governance procedures for a traceable and responsible usage of data in the continuum. Keeping track of the provenance of the data, knowing how and where data are generated in the continuum, and who are consuming the data, are essential towards maintaining full control over the data. The main building blocks of the aerOS Data Fabric that implement mentioned mechanisms are: Data Catalog (a tool for organizing and finding data available within an enterprise), Data Security (to provide access control over the data that are exposed through the Data Fabric), and Data Product (the conceptual representations of (physical) data assets).

aerOS will support execution of decentralized AI tasks expressed as workflows that can be commissioned to run over different Infrastructure Elements (IEs) in the continuum with optional use of frugality techniques (for model training: one/few-shot learning, heterogenous FL; for model inference: TinyML, model reduction) and inclusion of explainability/interpretability.

Furthermore, the main objective of the trustworthiness in aerOS is to provide a high level of security in the whole aerOS ecosystem guaranteeing that malicious actions will be avoided at the highest-level degree possible. Towards this direction, trustworthiness is guaranteed in aerOS by deploying a distributed trust management approach that collects several attributes from IEs to calculate their trust scores as well as the trust score of the whole aerOS domain.

Finally, the aim of management services and the aerOS management portal (tasks started in M10) is the development of the aerOS Basic Service in the architecture "aerOS Management Framework", which is composed of two independent components: the aerOS Management Portal (a single entry point for end users to manage and interact with the continuum) and the aerOS Federator (a management service responsible for controlling the establishment and maintenance of federation mechanisms among the multiple aerOS domains that form the continuum).

The finalization of this deliverable will boost the on-going implementation of the respective software components. Within next iterations of the deliverable (D4.2 in M18, D4.3 in M30) specifications of proposed solutions may be modified and extended due to ongoing work in WP2 and WP3, and further analysis of needs coming from pilot use cases and needed integrations between other products of WP4 and products of WP3.

Table of contents

List of tables

List of figures

List of acronyms

Acronym	Explanation
AAA	Authentication, Authorization and Accounting
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
CSV	Comma Separated Values
DaaP	Data-as-a-Product
DevOps	Development and Operations
DevSecOps	Development, Security and Operations
DevPrivSecOps	Development, Privacy, Security and Operations
DLT	Distributed Ledger Technology
DSBA	Data Spaces Business Alliance
DSSC	Data Spaces Support Centre
DXWG	Dataset Exchange Working Group
EDP	European Data Portal
FaaS	Function-as-a-Service
FL	Federated Learning
FOAF	Friend Of a Friend (ontology)
HA	High Availability
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure-as-a-Service
ICOS	IoT Cloud Operating System
IdM	Identity Management
IE	Infrastructure Element
IoT	Internet of Things
iPaaS	Integration-Platform-as-a-Service
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
JVM	Java Virtual Machine
K8s	Kubernetes
LD	Linked Data
MQTT	MQ Telemetry Transport
ML	Machine Learning
NaaS	Network-as-a-Service

NGSI-LD	Next Generation Service Interface – Lined Data
PaaS	Platform-as-a-Service
PAP	Personal Auto Policy
PDP	Policy Decision Point
PEP	Primary Entry Point
RML	RDF Mapping Language
SaaS	Software-as-a-Service
SKOS	Simple Knowledge Organization System
XAI	Explainable AI
XML	Extensible Markup Language

1. About this document

The main goal of this deliverable is to provide specifications of solutions and identification of technical components that are going to be developed under the scope of WP4 to deliver applications intelligence at the edge. These solutions along with the solutions proposed in WP3, are the technological backbone of the aerOS project and they will enable the deployment of an aerOS architecture.

It should be highlighted that this deliverable corresponds to the first out of three documents, and therefore its content will be expanded and adapted as the project evolves. This is motivated by different reasons, including the fact that both the requirements and the architecture produced by the work of WP2 are still evolving (and therefore modifications may be needed), and as a result the interactions between components from WP3 and WP4 may also require adaptations (in the form of new interfaces, methods, components, etc.).

1.1. Deliverable context

Item	Description
Objectives	**O3** (Definition and implementation of decentralized security, privacy and trust): Specification of mechanisms for trustworthiness and decentralized trust management. **O4** (Definition and implementation of distributed AI components with explainability): Specification 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). Specification of mechanisms for data homogenization, governance, traceability, provenance and lineage.
Work plan	D4.1 takes input from: • T2.1 (state-of-the-art): Novel components and technologies research for further design choices • T2.2 (use cases and requirements): To be evaluated and fulfilled with the proposed solutions • T2.5 (architecture): Design principles and high-level functionalities to cover D4.1 influences: • WP5 (integration, use case deployment, validation, evaluation, assessment): To later materialize solutions in pilot deployments D4.1 must be in line with: • WP2 (requirements, architecture, DevPrivSecOps): To develop according to DevPrivSecOps methodology and in line with the architecture design • WP3 (infrastructure components): To define functional boundaries and interactions
Milestones	This deliverable does not mark any specific milestone; still, it contributes to the realization of *MS3 – Components defined*, that will be achieved in M12. Although far in time (M30), it is also central part of *MS7 – Final software release.*
Deliverables	This deliverable receives inputs from D2.1 (State-of-the-Art and market analysis report), D2.2 (Use cases manual, requirements, legal and regulatory analysis (1)). It is prepared in parallel to D2.6 (aerOS architecture definition (1)) and synchronized with its content. Once components from WP4 are being delivered, they will feed the deliverables of WP5 related to integration, evaluation and use case deployment and evaluation.

1.2. The rationale behind the structure

This deliverable consists of 5 sections. It starts with general information about the document followed by an introduction that outlines focus areas covered in WP4 and relation to outcomes of WP3, all contextualized against the global aerOS stack and architecture. Next section includes specification of solutions (with respective technical components identified) divided by tasks that they belong to. Tasks denote the division into "focus areas" within WP4 and, consequently, corresponding subsections describe proposed approaches in aerOS. Each task-specific subsection is structured in the same way i.e., it includes: main functionalities, relation to requirements, high-level structure diagram of the solution, description of components and candidate technologies for realization. Some subsections are further divided into more specific parts of the proposed solution e.g., for data homogenization, annotation and translation are described separately. Finally, the two last sections of this document concludes with a summary of the future work carried out in the work package that will be included in the second version of the deliverable.

1.3. Outcomes of the deliverable

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

Briefly, data autonomy for homogenization will address functionalities such as semantic data annotation and semantic translation. Data governance, provenance, traceability, and lineage will be covered by three building blocks: Data Catalog, Data Security, Data Product with implementation supported by the concept of knowledge graphs. Outcomes from these data-related tasks shall follow the concept of Data Fabric.

Decentralized frugal AI will provide components to enable execution of AI tasks e.g., federated learning or inference on trained models with the possibility to include support of frugality and explainability when needed. Embedded analytics components will allow to deploy functions on aerOS infrastructure according to FaaS approach and provide output for analytics or building blocks for applications or workflows deployed on aerOS.

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

Work on management services and aerOS management portal has started in M10 so it will be only briefly outlined in this deliverable, however, it will be included in the second iteration of this deliverable (D4.2 Software for delivering intelligence at the edge intermediate release).

2. Introduction

WP4 together with WP3 constitute the two technical work packages of the aerOS project. The goal of WP4 is to enable delivering intelligence at the edge of the aerOS infrastructure by optimizing usage of data in aerOS without sacrificing control over it. To that end, solutions for data autonomy and semantic interoperability will be provided, as well as concepts related to the "Data Fabric" will be exploited to drive processing of data in the aerOS continuum. Moreover, the usage of decentralized frugal AI and analytics will be supported towards executing internal processes and external applications. Emerging security and data sovereignty challenges will also be addressed. Specifically, the objectives are to:

- propose standards and mechanisms for data annotation in order to enable other functionalities in this WP,
- define ontologies and semantic interoperability mechanisms for data autonomy,
- introduce data homogenization solutions for interoperability,
- provide mechanisms for trust and decentralized trust management,
- deliver tools for multiplane real-time data analytics and visualization,
- enable decentralized frugal AI on the edge,
- integrate elements and services of this WP under a management framework.

WP4 will complement and build upon the outcomes of WP3 that will provide a set of infrastructure components to enable secure, scalable IoT edge-cloud continuum, supporting resources and services orchestration to boost operation of autonomous systems, based on the aerOS architecture.

Furthermore, the present document (D4.1) focuses on defining the initial software components, relationships and building blocks in relationship with the aerOS architecture. aerOS will be developed following a metaOS approach, encompassing and addressing the project objectives. From the point of view of aerOS architecture, the system will be designed around several main building blocks and fundamental concepts. For that reason, the deliverable, D2.6 offers a comprehensive overview of aerOS, consolidating the main concepts and building blocks. Additionally, it analyses the core elements, services offered, and functionalities that will be offered. D4.1 was prepared in parallel to D2.6 and synchronized with its content.

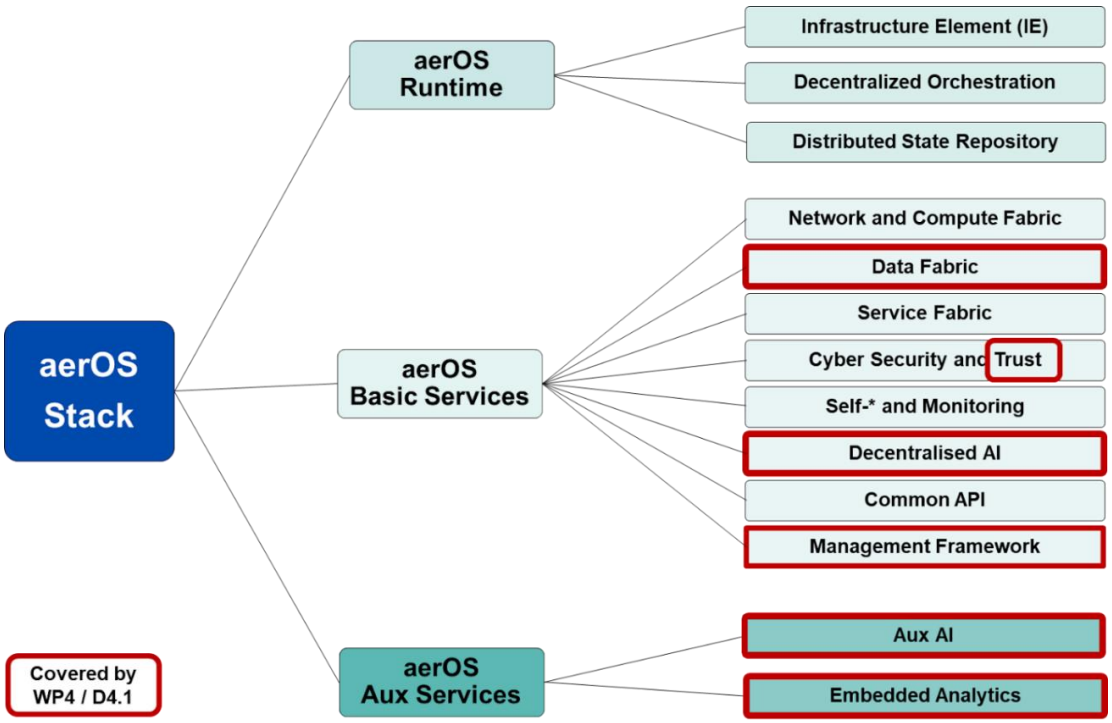

Figure 1. WP4 components in the aerOS stack

D2.6 outlines several principles that aerOS builds upon, such as the Infrastructure Element (IE) and the aerOS domain. On the one hand, the IE, as the most granular computing entity, serves as the fundamental building block. It enables the deployment and management of workloads through a flexible and adaptable physical or virtual entity, supporting containerised workloads, providing network connectivity, storage capacity, and a well-defined API to expose its state. On the other hand, the aerOS domain is formed by one or more IEs, creating a complete aerOS domain that facilitates the hosting and sharing of essential aerOS basic services across all IEs. In addition, D2.6 defines an architectural stack that provides a structured overview of runtime, basic, and auxiliary aerOS services. Figure 1 explicitly shows the location on this aerOS stack of the WP4 proposed solutions described in D4.1.

3. Preliminary proposal of software solutions

This section contains the specification of the preliminary proposal of solutions for WP4, along with the identification of their respective technical components. Each subsection corresponds to a task within WP4 and presents the identified solutions related to it. These subsections are structured in the same way i.e., it includes: main functionalities and general description, relation to requirements, high-level structure diagram of its components, a table with the components description, and candidate technologies for implementation. Some subsections are further divided into specific parts to provide more granular details.

3.1. Data autonomy for homogenization

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

3.1.1. Semantic annotation

The aerOS infrastructure utilizes semantic annotation to ingest "raw data" from external sources and enrich it with appropriate semantic information, to enable further processing. Currently, there are several existing open-source tools that can be accommodated/extended and applied for this purpose in aerOS. One of the more promising is the Semantic Annotator, developed within the ASSIST-IoT project[1]. It can annotate arbitrary JSON, XML, and CSV data, based on transformation rules expressed in YARRML/RML, JsonPath and XPath. The tool can perform "single-message" annotation tasks – using REST API, as well as "streaming" annotation with the help of Kafka/MQTT message brokers as mediators.

3.1.1.1. Related requirements

- TR-37 Semantic data annotation
- TR-38 Streaming semantic annotation
- TR-39 Semantic interoperability

3.1.1.2. Structure diagram

The following figure illustrates the structure diagram of the Semantic Annotator, including the relevant components that will be considered.

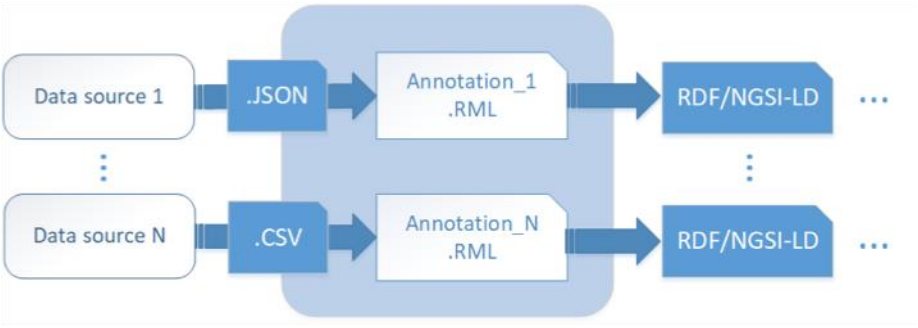

Figure 2. Semantic Annotator – data processing workflow

Such components are identified and further described in the table below:

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

Table 1. Components for Semantic Annotator

Component	Description	Interactions
Data source	A streaming (backed by Apache Kafka or Eclipse Mosquitto) or REST-API client submitting the "raw data" to the Semantic Annotator.	Raw data → semantically annotated data
Annotation in .RML	The annotation logics/rules expressed in the RDF Mapping Language.	Provides the "annotation logic" for the raw data.

3.1.1.3. Candidate technologies and standards

Table 2. Candidate technologies for Semantic Annotator

Technology/Standard	Description	Justification
RML/CARML	RDF Mapping language	RML is a (standard) mapping language defined to express custom mapping rules for transforming heterogeneous data structures to RDF. CARML is a Java library for performing transformations expressed in RML.
Scala	A modern "multi-paradigm" programming language, targeting (among others) the JVM	Scala is a "hybrid" programming language that supports both object-oriented and functional paradigms. It offers modern, concise and expressive syntax, high efficiency and excellent interoperability with the Java ecosystem.
Akka/Pekko HTTP	An Akka/Apache Pekko based library for handling the HTTP protocol requests	The semantic annotation component will offer REST API for configuration and management.
Apache Kafka	A high performance, distributed streaming platform	A solution that can be used for large-scale deployments, where the semantic annotation component will need to handle vast amounts of data.
Eclipse Mosquitto	An MQTT broker	A messaging solution that can be used for low power environments, in particular, close to the edge, where data annotation will usually take place.

3.1.2. Semantic translation

The semantic translation within aerOS will be realized via the Semantic Translator component, which is based on IPSM (Inter-Platform Semantic Mediator), a generic streaming semantic translation software developed by INTER-IoT[2], and further enhanced within the ASSIST-IoT European projects. The translation "rules" are expressed in the IPSM Alignment Format (IPSM-AF). IPSM offers a highly-scalable, efficient translation architecture, with two main interfaces: REST and reactive streaming. The first option offers quick, one-message-at-a-time, service, while the latter is meant for handling large asynchronous streams of data. In both cases, translation is done transactionally "per message" by applying rules defined in *alignment* files.

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

The *reactive stream* processing in IPSM, rests on semantic *translation channels*, and a *central ontology*. A translation *channel* is a one-way stream that accepts messages described with a given ontology and translates them into a different ontology. In IPSM, translation is performed "on the fly", as messages pass the channel, and is configured by two one-way alignments per channel – mediated by the central ontology. An important aspect of the central ontology is its *modularity*.

3.1.2.1. Related requirements

- TR-36 Reactive data streams handling
- TR-39 Semantic interoperability
- TR-40 Core data models/ontologies

3.1.2.2. Structure diagram

The following figure illustrates the structure diagram of the Semantic Translator, including the relevant components that will be considered.

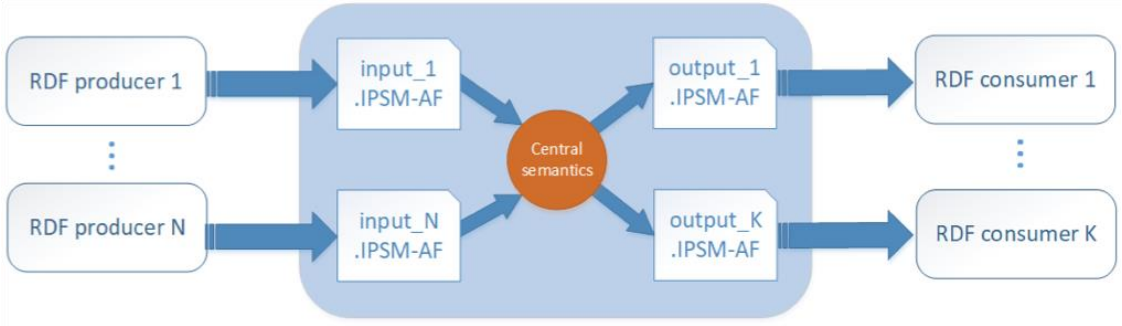

Figure 3. Semantic Translator – data processing workflow

Such components are identified and further described in the table below:

Table 3. Components for Semantic Translator

Component	Description	Interactions
RDF producer	A source of RDF data for the Semantic Translator. It can submit the data for translation in two ways: • via REST API call, to perform a single RDF graph translation • via streaming interface (backed by Apache Kafka or Eclipse Mosquitto) Semantic Translator can handle many RDF producers concurrently.	RDF producer → input of the translation channel An arbitrary source of RDF data can submit it to the Semantic Translator via one of the available interfaces (REST or streaming – MQTT/Kafka).
Central semantics	A deployment-dependent modular ontology chosen as a "mediating point" for the semantic translation process.	Provides the target and source semantics for the translation expressed via input and output alignments, respectively.
Input alignment in IPSM-AF	Translation "rules", expressed in IPSM-AF format, provided by a data producer, and used by the component for transforming the input data and expressing it in the central semantics/ontology.	After "compilation" used by translation channel for expressing input data in terms of the central semantics.

Output alignment in IPSM-AF	Client-provided semantic translation rules for transforming the selected subset of the central semantics/ontology to the output/consumer semantics/ontology.	After "compilation" used by translation channel for expressing data represented within central semantics, in terms of the target semantics.
RDF consumer	A target client interested in receiving the output of the translation. The translation result can be received/consumed in two ways: • via REST API call, to perform a single RDF graph translation • via streaming interface (backed by Apache Kafka or Eclipse Mosquitto) Semantic Translator can handle many RDF consumers concurrently.	Output of the translation channel → RDF consumer An arbitrary target for RDF data, that can consume it from the Semantic Translator via one of the available interfaces (REST or streaming – MQTT/Kafka).

3.1.2.3. Candidate technologies and standards

Table 4. Candidate technologies for Semantic Translator

Technology/Standard	Description	Justification
Apache Jena	A mature, JVM-based RDF processing framework to be used for RDF graph transformations.	Chosen as being the de facto standard for building JVM-based semantics-targeted applications, offering efficient and flexible RDF handling mechanisms.
Scala	See Section 3.1.1.3	See Section 3.1.1.3
Akka / Apache Pekko	An actor-based toolkit for creating highly concurrent, resilient, message-driven applications.	Thanks to the nature of actor-based solutions Akka/Pekko allows to create dynamically scalable applications, using high-level, expressive and abstract components.
Akka/Pekko HTTP	See Section 3.1.1.3	Within the Semantic Translation component, HTTP requests will be used for both configuration management and for non-streaming "single-message" translation processing.
Apache Kafka	See Section 3.1.1.3	A perfect solution for large-scale deployments, where the Semantic Translation component will need to handle vast amounts of messages/data.
Eclipse Mosquitto	See Section 3.1.1.3	A messaging solution that can be used for the low power environments, in particular, close to the edge.

3.2. Data governance, traceability, provenance, and lineage

The Data Fabric in aerOS is a metadata-driven architecture that automates the integration of data from heterogeneous sources and exposes the data through a standard interface. The Data Fabric transforms the raw data of the providing domain into a data product that follows a standard data model; and the resulting data product can be shared with consuming domains through the standard interface of the Data Fabric.

The aerOS Data Fabric shall include mechanisms supporting data governance procedures for a traceable and responsible usage of data in the continuum. Keeping track of the provenance of the data, knowing how and where data are generated in the continuum, and who are consuming the data, are essential towards maintaining full control over the data. In addition, it covers the conception of all data available in a continuum as a single box that can be queried and will forward the proper information.

The main building blocks of the aerOS Data Fabric that implement the mechanisms mentioned above, are described in detail in this section. As described in the aerOS Data Fabric architecture (see D2.6), these building blocks are the following: Data Catalog, Data Security, and Data Product.

3.2.1. Data Catalog

In a few words, a data catalog is a tool for organizing and finding data available within an enterprise. The data catalog promotes data democratization through a unified, standardized searching system that helps data users (e.g., data scientists) to find interesting data for their business cases. Precisely, the data catalog optimizes data exploitation by capturing related metadata such as the physical structure, the meaning, location, or ownership.

On the other hand, the data catalog also facilitates data governance activities as it provides a global view of what data is available, who is responsible for the data, and who is consuming the data. Classification of data as sensitive or confidential can help the chief data office (CDO) and the cybersecurity team to ensure that data is secured and used only by authorized consumers for a specific purpose.

To manage all the datasets and any related information, the data catalog builds upon a metamodel (i.e., a data model for the metadata) that is designed based on the requirements of the enterprise.

3.2.1.1. Conceptual design of the Data Catalog metamodel

In the case of aerOS, the data catalog aligns with the principles of the data mesh paradigm, thus, concepts such as data domains and data products must be captured. A high-level overview of the metamodel envisioned by aerOS is depicted in Figure 4.

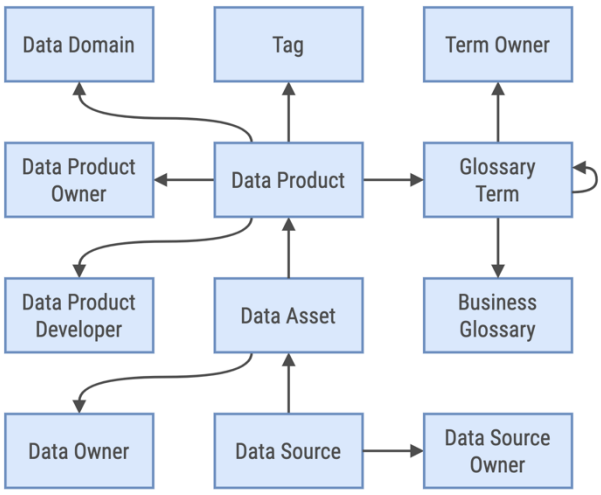

Figure 4. Data catalog metamodel defined by aerOS

The elements captured in the aerOS data mesh metamodel are described in the following table:

Table 5. Elements that compose the aerOS Data Catalog metamodel

Element	Description
Data Source	Application that exposes some data. There are three different types of data sources: 1. Batch (e.g., MySQL, Postgres) 2. Real-time (e.g., Kafka, MQTT) 3. API (e.g., REST). **Note:** APIs represent a special case. Managed as batch and/or

	streaming data source depending on the implementation.
Data Asset	Collection of data that follows a particular structure and format and is available through a data source. For example, a table in a relational database, a topic in Kafka, a file in a filesystem or an object storage platform like AWS S3.
Data Owner	Person that produced the dataset in its respective data source.
Data Product	Conceptual representation of data as understood at the business level. Linked with data assets that represent the physical representation for the data.
Data Domain	Maps to a business capability. Represents a bounded context that encapsulates data used to solve a business problem. Provides a grouping of data products that are logically related.
Data Product Owner	Person accountable for managing the life cycle of data products within a particular data domain. Interacts with data product consumers to collect their feedback and identify their requirements based on their use cases.
Data Product Developer	Person with practical experience on the data and responsible for making data products within a data domain. Collaborates with the data source owner in the creation of the data product.
Data Source Owner	Person responsible for access control and maintenance of a data source. This differentiation from data owner and data steward is needed as the same data source can expose data from different domains.
Business Glossary	Controlled vocabulary that has been agreed within a particular data domain or globally across the enterprise.
Glossary Term	Business term that has been defined as part of a glossary. A term may link with other terms composing a taxonomy (hierarchical) or thesaurus (synonyms).
Term Owner	Person responsible for curating a term within a business glossary and its relationship with other glossary terms.
Tag	Free-form terms that data users can propose and link to data products.

3.2.1.2. Data Catalog metamodel implementation

The proposed metamodel determines how the aerOS Data Catalog organizes metadata internally. In this sense, the aerOS Data Catalog leverages the knowledge graph based on the NGSI-LD standard for integrating and exposing the metadata. In fact, note that the conceptual metamodel presented before already has the shape of a graph. Since the NGSI-LD information model enables referencing existing ontologies, the standard DCAT ontology has been pinpointed as the baseline for implementing the metamodel. By making the aerOS metamodel compliant with DCAT, the aerOS Data Catalog would interoperate with public data catalogs such as the European Data Portal (EDP)[3], which uses the DCAT-AP extension for managing metadata.

[3] https://data.europa.eu/en

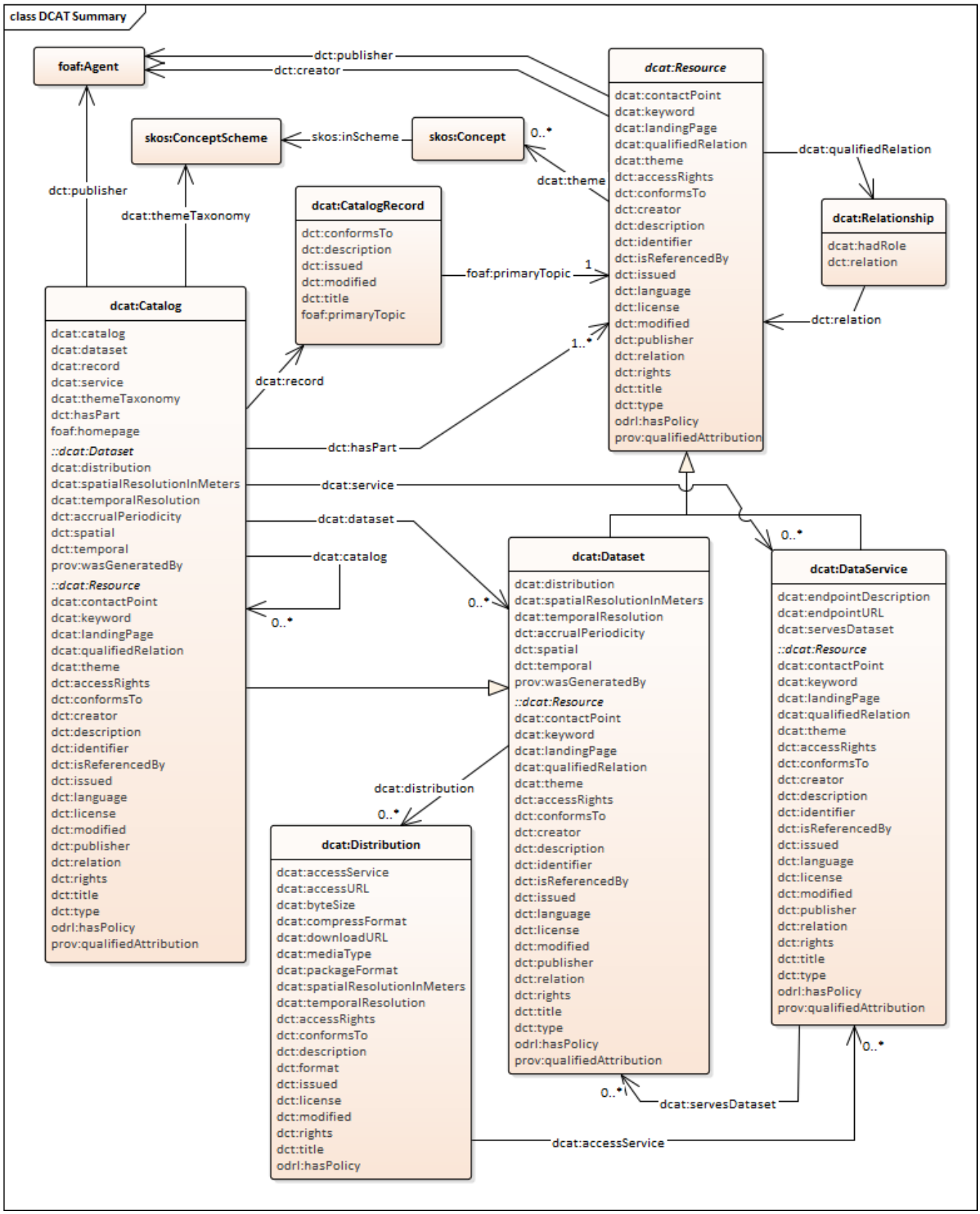

Figure 5. Model diagram of DCAT ontology[4]

[4] https://www.w3.org/TR/vocab-dcat-2/

The latest release DCAT 2.0 has been analyzed to determine to which extent the ontology covers the requirements identified in the envisioned data mesh metamodel. During this analysis, the following gaps have been identified:

Table 6. Gap analysis between aerOS metamodel and DCAT 2.0 ontology

aerOS data catalog metamodel	DCAT 2.0 ontology	Remarks
Data Source	dcat:DataService	The "DataService" class is mainly designed to support data sources that serve file-based distributions, which can be downloaded from a URL, such as filesystems or object storage platforms like AWS S3. However, support for other types of data sources like streaming platforms, relational databases, NoSQL databases, or REST APIs, will require adjustments in the model.
Data Asset	dcat:Distribution	The "Distribution" class conveys the physical manifestation of a collection of data. A Distribution can be exposed through a DataService.
Data Product	dcat:Dataset	The ontology defines the "Dataset" class as "the conceptual dataset" that can have "different manifestations or distributions". This definition partially fits the aerOS vision of the data product as the conceptual representation of some data from the perspective of the business. Additionally, the current draft of the next release DCAT 3.0 introduces the class "DatasetSeries" as "a dataset that represents a collection of datasets that are published separately but share some characteristics that group them". This definition may apply to the definition of a data product by a data mesh as a combination of conceptual models. For this reason, the new "DatasetSeries" could be the basis for defining a new class "DataProduct" in the ontology.
Data Domain	dcat:theme	The "theme" property of a "Resource" is defined as the main category of the resource. In this case, the category could map to a data domain as defined by data mesh.
Data Product Owner Data Product Developer Data Owner Data Source Owner	foaf:Person	DCAT 2.0 ontology allows for subclasses of a Resource to draw relationships with agents through the prov:qualifiedAttribution. The range of the relationship can be Person as defined by the FOAF ontology.
Business Glossary	skos:ConceptSchem e	A "Catalog" class can be linked to "ConceptSchema" class as defined by the SKOS ontology.
Glossary Term	skos:Concept	DCAT 2.0 ontology allows for subclasses of a Resource to be linked with Concept individuals as defined by SKOS ontology.
Term Owner	foaf:Person	Out of scope of DCAT as this applies to the "skos:Concept" class.
Tag	dcat:keyword	The "keyword" property of a DCAT "Resource" allows for adding labels or tags as literal values to resources registered in the catalog.

In summary, the results from this analysis show that the mapping the DCAT 2.0 ontology with data mesh principles requires a few adjustments. Essential concepts such as data domain or data product are neither modelled in the ontology nor mentioned in the description of the existing classes. The Dataset Exchange Working Group (DXWG), which is responsible for the DCAT ontology, has identified this need as well[5]. Drawing from the gaps identified in Table 6 and syncing with the efforts done in the W3C DXWG, future releases of aerOS will work on the alignment of the DCAT with the data mesh principles. The resulting data mesh ontology will serve as the final ontology that will be mapped to NGSI-LD for the implementation of the aerOS Data Catalog.

3.2.1.3. Management of Data Catalog metadata

Based on the NGSI-LD data model derived from the aerOS Data Catalog metamodel, the aerOS Data Catalog includes mechanisms for collecting and managing metadata. The types of mechanisms envisioned by the Data Catalog are the following:

- **Data source scanning:** The connector interacts with a registered data source to discover related metadata. For example, OpenAPI has become the standard for modelling REST APIs, describing the available operations and the data models used for each operation. Many implementations of REST APIs expose an endpoint through which their OpenAPI specification can be retrieved. A data source connector can leverage this endpoint to discover the data exposed by the REST API data source.
- **Crowdfunding:** Users of the data catalog provide metadata through the Data Catalog's interface. For example, an application owner registers the application as a new data source in the Data Catalog. The registration information can later be leveraged to scan the source and automatically retrieve additional metadata.

3.2.1.4. Related requirements

- TR-38 Steaming semantic annotation
- TR-46 AI-related data models adaptability and extendibility
- TR-48 Support for data frugality
- TR-49 AI job orchestration
- TR-53 Internal and external AI support

3.2.1.5. Structure diagram

The following figure illustrates the structure diagram of the Data Catalog in the Data Fabric, including the relevant components that will be considered.

[5] https://github.com/w3c/dxwg/issues/1493

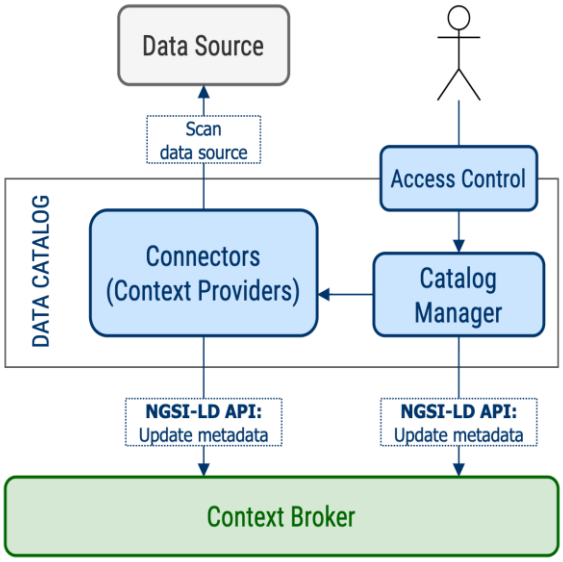

Figure 6. Architecture of aerOS Data Catalog

The following table describes in detail the components involved in the architecture of the aerOS Data Catalog:

Table 7. Components for aerOS Data Catalog

Component	Description	Interactions
Connectors	Containerized microservices that scan a registered data source to automatically discover related metadata such as available datasets.	Connector → Data Source: The connector interacts with the data source through its native protocol to discover metadata. Connector → Context Broker: The connector sends the discovered metadata to the context broker.
Catalog Manager	Component responsible for deploying and configuring connectors upon data source registration.	Catalog Manager → Connector: Deployment and configuration of connectors Context Broker → Catalog Manager: Context broker sends notifications about registered data sources.

3.2.1.6. Candidate technologies and standards

Table 8. Candidate technologies and standards for the aerOS Data Catalog

Technology/Standard	Description	Justification
DCAT	OWL ontology for building data catalogs, standardized by the W3C. DCAT ontology supports extensions through application profiles which introduce constraints for specific use cases e.g., DCAT-AP.	Standard ontology that will serve as the baseline for defining the metamodel of the aerOS Data Catalog. DCAT will be aligned with data mesh principles, with focus on the concepts of data product and data domain. Additionally, DCAT will be analysed to pinpoint any gaps for incorporating the different data sources used in aerOS.
Orion-LD	Open-source implementation of an NGSI-LD Context Broker.	This component represents the core of the knowledge graph.

Docker/Kubernetes	Docker is the de-facto standard for containers while Kubernetes has become the standard for managing containers.	Catalog connectors and the Catalog Manager will be implemented as containerized microservices orchestrated through Kubernetes.

3.2.2. Data Security

The purpose of the Data Security building block is to provide access control over the data that are exposed through the Data Fabric. This building blocks includes a set of services that leverage aerOS cybersecurity tools to ensure only authenticated and authorized data consumers can access the requested data from the Data Fabric. A clear set of roles and permissions will be established based on the AAA block of aerOS. This will determine the capabilities and actions available to each user of the Data Fabric. In addition, owners of data products and data governance team members will be able to define access control policies, based on different patterns like roles or attributes, and the Data Security services will take care of registering these policies in the pertinent aerOS cybersecurity tools.

aerOS is focused on addressing the requirements to support data sovereignty in cooperation with European edge and cloud data initiatives. The main European data space initiatives play a pivotal role, offering specifications, tools, and procedures to achieve sovereign, interoperable, and reliable data-sharing practices. This becomes a notable strength of aerOS and its Data Fabric, as it relies on core FIWARE components (NGSI-LD and Smart Data Models) that effectively address data interoperability within data spaces. By leveraging these building blocks, aerOS aligns its work with the data sovereignty requirements outlined by prominent European data space initiatives like the Data Spaces Business Alliance (DSBA) and Data Spaces Support Centre (DSSC). Additionally, it is worth emphasizing that the data product approach represents a fundamental principle in the data fabric concept. However, in the context of data space development, the role of data products is currently in its early stages of growth. It is possible that data fabric and data space approaches could converge in the future, leading to the discovery of synergies and the definition of novel data-sharing approaches, embracing the data product conception as well.

3.2.2.1. Related requirements

- TR-17 Cross-layer cybersecurity
- TR-50 Context-aware data access
- TR-57 Cybersecurity policies
- TR-61 Authentication and authorization accounting

3.2.2.2. Structure diagram

The following figure illustrates the structure diagram of the Data Security in the Data Fabric, including the relevant components that will be considered.

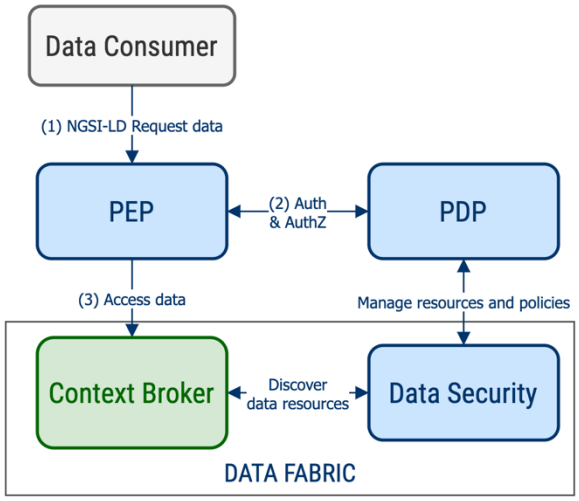

Figure 7. Overview of aerOS Data Security

Such components are identified and further described in the table below:

Table 9. Components for aerOS Data Security

Component	Description	Interactions
Data Security	Implements services responsible for managing access control policies defined by owners of data products and data governance team. Includes services that discover resources of the Context Broker to be protected and registers them in the Policy Decision Point (PDP).	Data Security → Context Broker Data Security → PDP

3.2.2.3. Candidate technologies and standards

Table 10. Candidate technologies and standards for the aerOS Data Security

Technology/Standard	Description	Justification
KrakenD	Open-source implementation of an API GW.	Performs the Primary Entry Point (PEP) role in the data access control architecture.
Keycloak	Open-source implementation of an Identity Provider (IdP) that provides functionalities for authentication and authorization.	Performs the PDP role in the data access control architecture.
Orion-LD	Open-source implementation of an NGSI-LD Context Broker.	As mentioned in previous sections, this component represents the core of the knowledge graph.

3.2.3. Data Product

The aerOS Data Fabric implements an architecture where data products are managed as the conceptual representations of (physical) data assets, where the conceptual representation is modelled in a knowledge graph using the NGSI-LD standard. In this sense, the Data Fabric facilitates data providing domains with all the mechanisms needed to build and share data products based on the NGSI-LD standard.

The set of mechanisms supported by the Data Fabric allows for flexible creations of data products, depending on the requirements of the data source from which data assets are ingested and the characteristics of the data, such as performance, privacy, or quality. These mechanisms and their interactions are described in detail in Figure 8.

3.2.3.1. Related requirements

- TR-34 Composable data topologies
- TR-35 Data streams handling
- TR-36 Semantic data annotation
- TR-38 Core data models/ontologies
- TR-47 Data collection
- TR-50 Context-aware data access
- TR-51 Distributed data management
- TR-52 Data integration
- TR-53 Data-as-a-product

3.2.3.2. Structure diagram

The following figure illustrates the creation of the Data Product in the Data Fabric, including the relevant phases that will be considered.

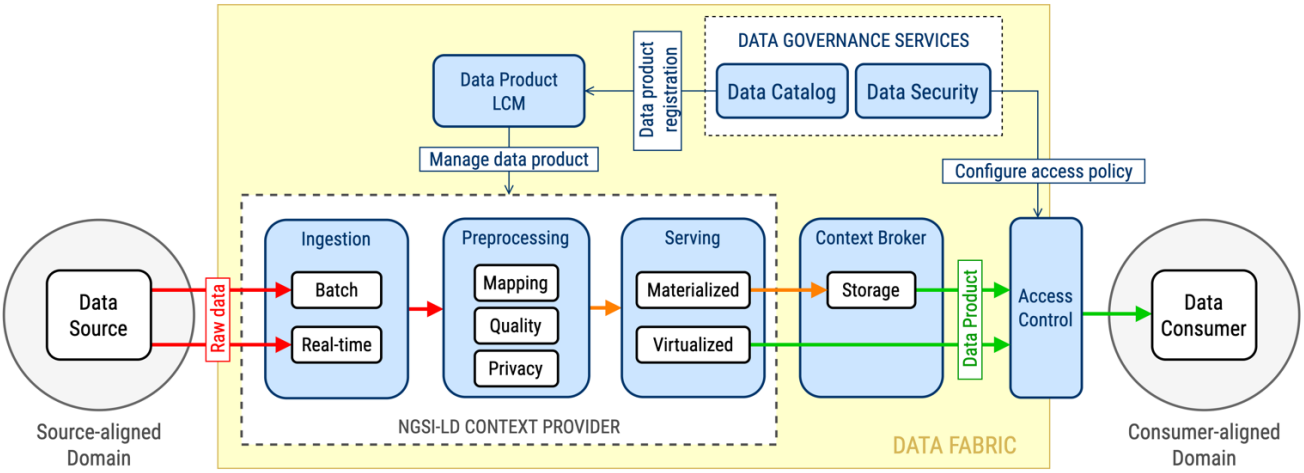

Figure 8. Creation of a Data Product

The following table describes in detail the components involved in the creation of data products within the aerOS Data Fabric:

Table 11. Components involved in the creation and consumption of a Data Product

Component	Description	Interactions
Data Product LCM	Manages the life cycle of a data product in the Data Fabric. Receives notifications from the Data Catalog containing the information and resources required for the creation of the data product such as NGSI-LD schema or mapping code.	Data Product LCM → Context provider Data Catalog → Data Product LCM
Data source	Represents an application that contains raw data which can be turned into a data product. This application belongs to a source-aligned data domain.	Data source → Ingestion
Ingestion	First component of the context provider. Component responsible for collecting raw data from a data source and	Data source → Ingestion

	its delivery to the following components of the context provider.	Ingestion → Preprocessing
Preprocessing	Second component of the context provider. Receives data from Ingestion component and then performs three main tasks: 1. **Mapping**: Input raw data is translated to the structure defined by the NGSI-LD data model(s) specified in the data product. This stage leverages the mappings mechanisms supported by the aerOS semantic annotation block, as described in section 3.1.1. Other mapping techniques like code-specific will also be evaluated. 2. **Quality**: The NGSI-LD schema will validate the data after being mapped to NGSI-LD in the previous step. 3. **Privacy**: For those NGSI-LD properties labelled as sensitive, privacy mechanisms like obfuscation or anonymization will be applied.	Ingestion → Preprocessing Preprocessing → Serving
Serving	Last component of the context provider. Receives data formatted in NGSI-LD from the Preprocessing component. This component has two different flavours: • **Materialization**: Default flavour. Data is sent to the context broker for persistence. Requires implementing an NGSI-LD client for interacting with the context broker. In this setup, the context provider adopts the role of context producer. • **Virtualization**: For virtual data products, data is directly exposed towards consumers. Requires implementing an NGSI-LD server for exposing the data. In this setup, the context provider adopts the role of context source.	Preprocessing → Serving Serving → Access Control
Context Broker	Stores data received from the Serving component. Additionally, includes a context registry that maintains a record of serving components that provide virtualized data products (i.e., context sources).	Serving → Context Broker Context Broker → Access Control
Access Control	Implements access control mechanisms to allow data product consumption only to those authorized data consumers. Refer to aerOS cybersecurity framework (described in the parallel deliverable D3.1) for more details.	Context Broker → Access Control Access Control → Data consumer Data security → Access Control
Data consumer	Application that consumes the resulting data product from the data fabric.	Access Control → Data consumer

3.2.3.3. Candidate technologies and standards

Table 12. Candidate technologies and standards for the aerOS Data Product block

Technology/Standard	Description	Justification
Kafka	Open-source streaming platform	Facilitates exchange of data between

	that has become the de-facto standard in the industry. Kafka provides high performance for high volume and low frequency data.	components of the context provider. Enables decoupling of components within the context provider, following a microservice-based approach.
Orion-LD	Open-source implementation of the NGSI-LD context broker.	As mentioned in previous sections, this component represents the core of the knowledge graph.
Docker/Kubernetes	Docker is the de-facto standard for containers while Kubernetes has become the standard for managing containers.	The Data Product LCM and components of each context provider will be implemented as containerized microservices orchestrated through Kubernetes.
FastAPI[6]	Python framework for building REST APIs. Provides high performance and easy code based on the popular pydantic.	Implementation of Serving component in this virtualization role.
JSON Schema	Standard data modelling language for JSON format.	Since it is based on JSON-LD, NGSI-LD schemas can be defined using this standard.

3.3. Decentralized frugal AI

Frugal AI is the optimal way to bring AI close to the edge. The research in aerOS with regards to Frugal AI focuses on a relaxed need of abundant well-labelled data in order to perform smart, distributed orchestration of workloads (internal AI) and on the deployment of specific AI services for stakeholders over the continuum (external AI). Thus, in T4.3 the topic of efficient implementation and orchestration of selected distributed frugal and/or explainable AI methods on resource constrained devices is touched upon. In this section, possible approaches to decentralized frugal AI in aerOS are described. The proposed solutions are based on the concept of AI workflows and orchestration, frugality and explainability support.

3.3.1. AI workflows and orchestration

The proliferation of IoT devices and rising popularity of edge-cloud infrastructure deployments enable a new approach to prepare, use and maintain AI solutions. First of all, model training can be done in a decentralized fashion without moving the data to a central location (e.g., cloud). This approach, called Federated Learning (FL), is attractive as it allows to reduce the computational load of a single infrastructure element and scale the system dynamically. Moreover, it helps mitigate some privacy issues that may arise from data owners.

The idea behind FL is to iteratively train a model on several clients (in the case of aerOS, they can span over several IEs belonging to the continuum) with local data. After each round the parameters obtained for a local model are aggregated by an AI Task Controller and an updated global model is distributed to the clients.

In aerOS, the training of models will be defined as AI workflows where FL tasks will be divided into sub-tasks and delegated for execution to IEs matching requirements for each sub-task. Sub-tasks can include an execution of a local ML training round or data preparation (consuming, transformation, etc.). aerOS general mechanisms for service/task orchestration and reliability will allow to dynamically react and adopt to potential changes in the continuum. Moreover, aerOS will choose IEs to be involved in the workflow execution based on requirements and in a way abstract to the end user. aerOS should enable the concurrent use of heterogenous IEs with differing capabilities, while at the same time allowing to specify tasks/sub-task restrictions e.g., how many epochs should there be, what is the stopping criteria, what is the minimum number of clients.

aerOS, besides supporting FL, will allow to define workflows for deploying services with already trained models in the continuum. Such deployment will also be guided by requirements that will allow to choose the best

[6] https://fastapi.tiangolo.com

place for deployment using general aerOS orchestration mechanisms. The deployment of the trained model is a variation of AI workflow in which separate sub-tasks can exist e.g., data preparation or inclusion of explainability. In the simplest case the AI workflow can contain only one step corresponding to one AI task with one AI sub-task.

3.3.1.1.1. Frugality

Mechanisms proposed for the realization of AI workflows can, if needed, due to the execution environment characteristics, support frugality. Note that, decentralized AI e.g., federated learning, is not equivalent to frugal AI. 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. On the other hand, 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, having in mind also the fact, that frugality of the AI solutions is often related to lower accuracy of the resulting model. Frugality is something that needs to be considered per use case for selected techniques and acceptable loss of model quality. However, aerOS will provide access to the most popular approach to be utilized when deploying workflows on the continuum.

In aerOS several design choices have been made to support frugality of decentralized AI in both model training and deployment of trained models as services. Model training and inference on the trained model (exposed as a service) are two separate and independent design decisions.

For training, aerOS will support techniques such as:

- One-/few-shot learning - training with limited data samples
- Heterogenous FL - using different architectures of local models (based on client characteristics) to train a global model

These techniques will influence the definition of workflow and its interpretation by the aerOS.

For inference of trained models under resource-constrained conditions e.g., close to the network edge:

- Utilization of low-resource tools according to the paradigm of TinyML (e.g., Keras2C, TensorFlow Lite, PyTorch Mobile) to deploy AI-related services on IEs. TinyML is defined as an embedded ML technique that enables ML applications on multiple cheap, resource- and power-constrained devices.
- Reduction of models' techniques - specifically model compression techniques will be available:
    - Binarization/quantization - converts model weights from high-precision floating-point representation to low-precision floating-point (FP) or integer (INT) representations
    - Pruning - discarding the weights that do not improve a model's performance

These techniques will be realized as: (1) preparation of "minimized" AI-related services to be deployed on aerOS, (2) preparation of services that will expose functionality to reduce an AI model.

Finally, in case of lack of data for model training then techniques for data augmentation will be available. Augmentation is artificially increasing the training set by creating modified copies of a dataset using existing data.

Depending on the use case, support for frugality will influence AI workflows by either including additional logic related to sub-tasks distribution or calling of additional services e.g., to generate missing data. Therefore, application of frugality techniques should be dependent on user or system requirements, that should determine the need between "full" or "minimized" deployment.

Note, that application of frugality is independent of the inclusion of explainability/interpretability in the designed AI workflows.

3.3.1.2. Related requirements

- TR-42 AI jobs execution in the continuum
- TR-43 Support for non-centralized data processing

- TR-44 Models deployed in the continuum
- TR-45 User requirements for AI jobs
- TR-47 Compliance with frugal AI paradigm
- TR-48 Support for data frugality
- TR-53 AI job description
- TR-54 Internal and external AI support

3.3.1.3. Structure diagram

Decentralized AI will be realized as AI workflows that can be executed on the aerOS infrastructure. It will require dedicated AI-related components that will interact with other aerOS components. Specially, it has been very crucial to analyze the evolution of the global aerOS architecture design, specifically the orchestration artifacts. aerOS has defined a two-level orchestration schema where the workloads (containerized, virtualized software achieving a functionality) are first assigned to a particular IE (HLO) and, then, deployed over that specific IE (LLO). Thus, as the AI workflows will be nothing but a combination of single workloads, put together in a way to achieve a specific functionality (AI service) of AI. It has been designed that the orchestration of AI workflows will make use of the two previous (HLO and LLO commodities), adding an extra layer on top of those. Here, the specific relations among the workloads that form the workflow will be the aspect that will be managed by the AI Service Controller. Such relations will be strongly directed by the necessities expressed by the user / IoT service provider wishing to deploy such a workflow.

The AI task (expressed as AI workflow definition) to be executed, can be treated as the extension of concept of an *App Definition Intention*. Moreover, as mentioned, the deployment and orchestration of workloads corresponding to sub-tasks from the AI workflow will be done by an aerOS orchestrator. The AI Service Controller will accept AI task/workflow definitions and translate them into a format understandable by the aerOS orchestration that will deploy a set of interacting services - AI Task [n] Controller and AI Local Executors.

The following figure illustrates the structure diagram of the AI workflow execution, including the relevant components that will be considered.

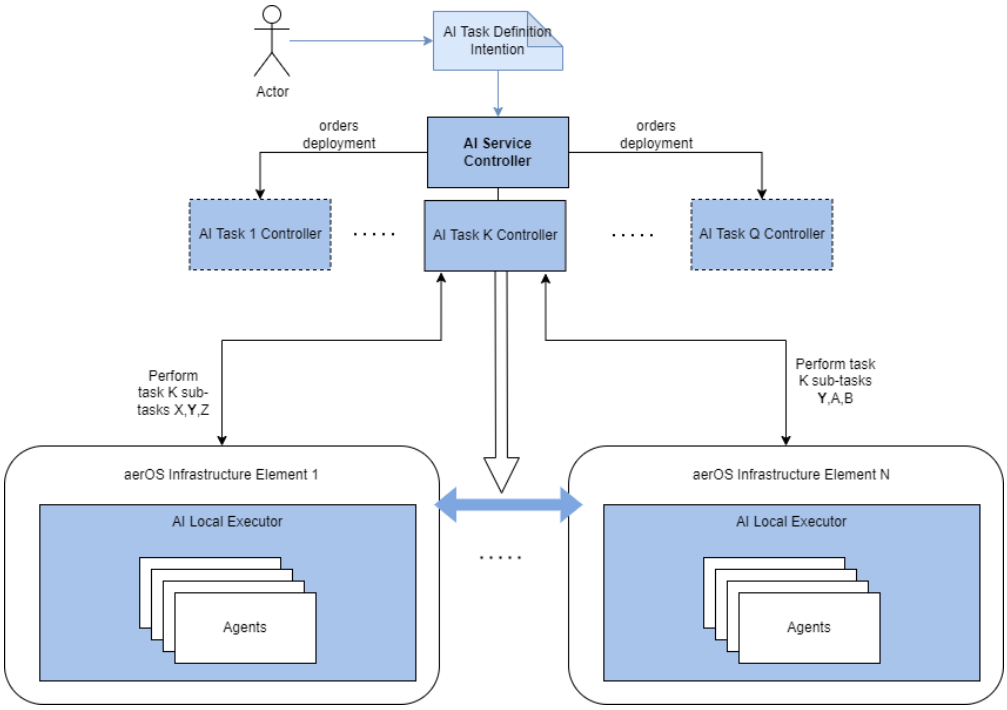

Figure 9. Overview of decentralized AI execution structure

Such components are identified and further described in the table below:

Table 13. Components for decentralized frugal AI

Component	Description	Interactions
AI Service Controller	This component accepts AI Task Definition Intentions being definitions of AI workflows to be deployed on aerOS. It is responsible for a correct interpretation of related intensions and their translation into input acceptable by the aerOS orchestrator. It exposes an API accessible for external actors.	Interacts with general aerOS components for API exposure and orchestration.
AI Task [n] Controller	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 Task [n] Controller ↔ AI Local Execution - synchronization of partial results for AI workflow AI Task [n] Controller ↔ AI Model Reduction Service - optional utilization of frugal techniques as part of AI workflow
AI Local Execution	A service deployed on aerOS infrastructure to execute workload for an AI sub-task i.e., to execute a granular "step in the workflow".	AI Task [n] Controller ↔ AI Local Execution - synchronization of partial results for AI workflow
Agent	The internal component in AI Local Execution services that is responsible for an actual execution of a sub-task.	Internal to AI Local Execution.
AI Model Reduction Service	An auxiliary service exposing functionalities related to model reduction. The service, if needed, can be deployed on aerOS and included in AI workflow definition.	AI Task [n] Controller ↔ AI Model Reduction Service - optional utilization of frugal techniques as part of AI workflow

3.3.1.4. Candidate technologies and standards

Table 14. Candidate technologies for decentralized frugal AI

Technology/Standard	Description	Justification
Python	Programming language	Well-suited for ML/AI applications with a lot of libraries.
FastAPI	REST API library for Python	Allows to build REST API for Python applications.
Docker/Kubernetes	Docker is the de-facto standard for containers while Kubernetes has become the standard for managing containers.	Components will be implemented as containerized micro-services.

TensorFlow	An open-source library for ML and AI particularly focused on training and inference of deep neural networks.	Functionality related to execution of different ML/AI-related tasks.
TensorFlow Lite	A mobile library for deploying models on mobile, microcontrollers and other edge devices.	Can be part of a minimal deployment of AI-related services in accordance with TinyML concepts. Functionalities related to frugality support.
PyTorch	An open-source ML framework used for applications such as computer vision and natural language processing.	Functionality related to execution of different ML/AI-related tasks.
PyTorch Mobile	A version of PyTorch framework dedicated for edge and resource-restricted devices.	Can be part of a minimal deployment of AI-related services in accordance with TinyML concepts. Functionalities related to frugality support.
Keras	An open-source framework providing interface to neural networks e.g., using TensorFlow as backend.	High-level, deep learning API for execution of different ML/AI-related tasks. Functionalities related to data augmentation.
Keras2C	A library for deploying Keras neural networks in C99, using only standard libraries. It is designed to be as simple as possible for real time applications.	Can be part of a minimal deployment of AI-related services in accordance with TinyML concepts. Functionalities related to frugality support.
Scikit-learn	A Python ML library for simple and efficient tools for predictive data analysis.	Functionality related to execution of different ML/AI-related tasks.
Larq	An open-source deep learning library for training neural networks with extremely low precision weights and activations, such as Binarized Neural Networks (BNNs).	Functionality related to model binarization.
Larq Compute Engine	A highly optimized inference engine for deploying extremely quantized neural networks, such as Binarized Neural Networks (BNNs).	Functionality related to model binarization on resource-restricted devices.
AugLy	Data augmentation library that supports four modalities (audio, image, text, video) and over 100 augmentations.	Functionality related to data augmentation in model training or evaluating robustness gaps of the model.

3.3.2. Explainability support

The rising popularity of AI-based applications, specifically in critical areas such as medicine or energy, triggered research and implementation of mechanisms that should increase the trustworthiness of the system also by means of enhancing accountability and enabling explainability of models and decisions taken.

In aerOS, AI will be 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 definition of AI workflow can include a step (an execution of e.g., FAAS) in which explainability/interpretability is used.

The objective for explainability support in aerOS is to prepare mechanisms to optionally "plug-in" such functionality in the AI workflow for an AI task execution. The popular and widely used metrics and tools have been selected to provide respective functions and to study trade-offs between explainability, cost in the edge (explainability tools vs IE configuration) and acceptable model accuracy (which is proved to be lower for better explainable methods).

Note that there are many methods and levels of detail possible that need to be matched to a model specification and needs of the end users. Interpretable models can be understood by humans without information, just by looking at summary / parameters with the ability to predict what is going to happen based on the input parameters. These include: decision trees, linear regression, rule-based. Explainable models require additional techniques to be understood, and include black-box models, e.g., ensembles, random forest, (deep) neural networks. Consequently, the aerOS design should allow in the future to use other metrics/tools in the same way, to enhance definitions of AI workflows.

The core methods considered in aerOS are the following:

- Local Interpretable Model-agnostic Explanations (LIME) - a technique that approximates any black box machine learning model with a local, interpretable model to explain each individual prediction
- Shapley Additive Explanations (SHAP) - a game theoretic approach to explain the output of any machine learning model

aerOS is planned to support both local (explain/interpret a single prediction) and global explainability/interpretability (explain/interpret the whole model).

3.3.2.1. Related requirements

- TR-46 AI-related data models adaptability and extendibility
- TR-55 Explainability support

3.3.2.2. Structure diagram

The following figure illustrates the structure diagram of the decentralized AI execution with explainability, including the relevant components that will be considered.

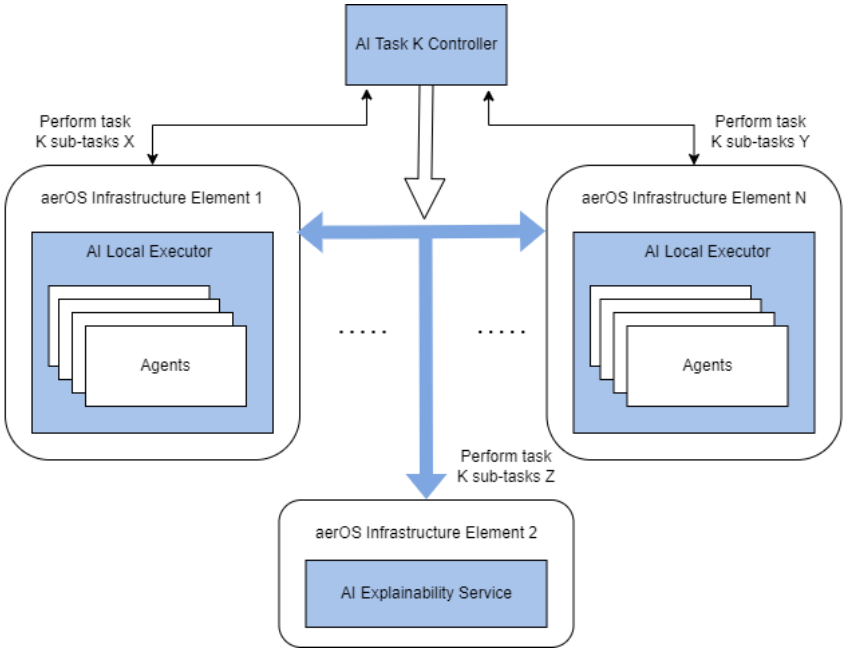

Figure 10. Overview of decentralized AI with explainability execution structure

Such components are identified and further described in the table below:

Table 15. Components for explainability support

Component	Description	Interactions
AI Explainability Service	Provides functionalities related to model explainability/interpretability on local and global level.	AI Task [n] Controller - can be included as a step in an AI workflow definition for execution of AI task to be executed on aerOS infrastructure.

3.3.2.3. Candidate technologies and standards

Table 16. Candidate technologies for explainability

Technology/Standard	Description	Justification
InterpretML	An open-source package that incorporates state-of-the-art machine learning interpretability techniques under one roof.	Glass-box models are interpretable due to their structure e.g., Explainable Boosting Machines (EBM), Linear models, and decision trees. Glass box models can provide explanations on both a global (overall behavior) and local (individual predictions) level. Black-box models are challenging to understand e.g., deep neural networks. Black-box explainers can analyze the relationship between input features and output predictions to interpret models e.g., LIME and SHAP.
AIX360	An open-source library that supports interpretability and explainability of datasets and machine learning models.	Includes a comprehensive set of algorithms that cover different dimensions of explanations along with proxy explainability metrics.
Dalex	X-rays any model and helps to explore and explain its behaviour,	Model agnostic. Compatible with many libraries (keras, scikit-learn, xgboost, etc.).

	helps to understand how complex models are working. The main function creates a wrapper around a predictive model. Wrapped models may then be explored and compared with a collection of local and global explainers.	Includes: LIME, SHAP, Break Down, Permutaion-based algorithms, Partial dependance profiles, Cetris Paribus profiles. A comprehensive tool, with well documented API. Allows for many different types of analysis and explainability algorithms.
Python	Programming language	Well-suited for data analysis and ML applications.
FastAPI	REST API library for Python	Allows to build REST API for Python applications.
Docker/Kubernetes	Docker is the de-facto standard for containers while Kubernetes has become the standard for managing containers.	Components will be implemented as a containerised micro-service.

3.4. Embedded multiplane analytics

Embedded multiplane analytics task offers a real-time data analysis approach at the edge, utilizing stratified sampling for monitoring, observing algorithm input data, and detecting anomalies and data drift, enhancing the edge's data analysis capabilities in the IoT edge-cloud continuum.

3.4.1. Description and main functionalities

The main objective of the Embedded Analytics Tool within aerOS is to provide an agile, customized framework for the creation, deployment, and execution of scripts for the purpose of incorporating functionalities such as advanced analytics, workflow models and the generation of aggregated data. These functionalities are intended to provide aerOS services with specialized data processing and decision-making capabilities to support services in their respective roles. The Embedded Analytics Tool must also support visualization features for executing of functions and exposing of data processed on the tool. This data should be accessible to non-technical users through dashboards supported by open-source community driven user guides for ease of use. It is agreed that the usage of the framework resulting from T4.4 will be consistent and frequent across the whole aerOS meta Operating System (metaOS). It would have created a very flexible and handy tool for applying quick processing over data for the sake of federated, decentralized decision-making in the overall functioning of the continuum.

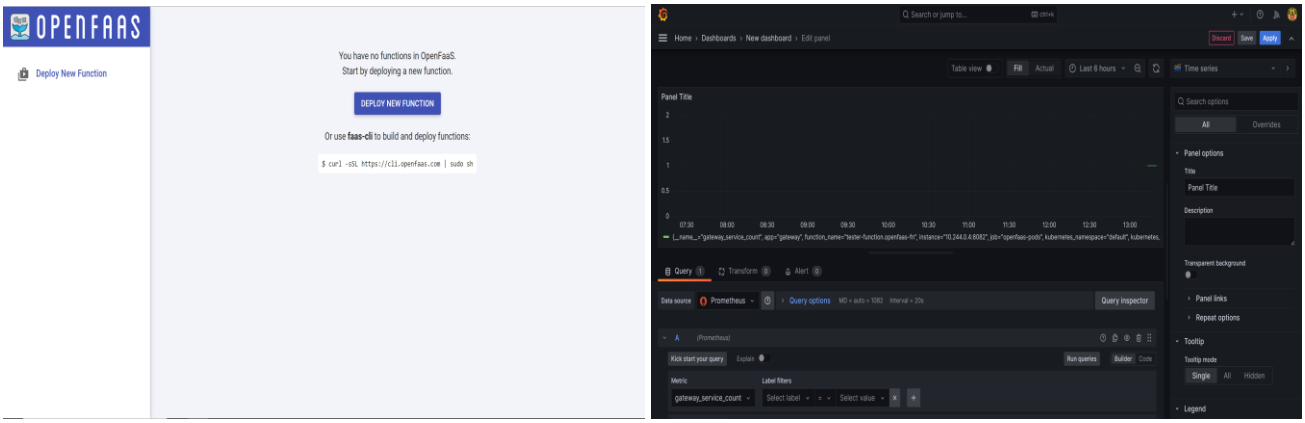

Figure 11. The OpenFaaS and Grafana dashboards

An example of these dashboards is shown in Figure 11, OpenFaaS (shown on the left) provides a Function as a Service (FaaS) framework to the Embedded Analytics Tool. The deployment and invoking of functions can be performed through the dashboard. Grafana (shown on the right) is used to visualize metrics and in-function data through graphs and descriptions. Both features are packaged with the Embedded Analytics Tool. The Embedded

Analytics Tool supports HTTP interfaces for gateway, Prometheus and Grafana components. The creation, deployment and execution of functions can be performed through the *faas-cli* application.

The *faas-cli* application provides a mechanism for the creation of OpenFaaS functions. The creation of functions utilizes a template, which specifies the scripting language, function interface and pre-packaged libraries/files to be accessible as part of the function. For the aerOS Embedded Analytics Tool, a specialized template has been created to allow the integration of new functionalities such as, the Grafana visualization component and the Prometheus *pushgateway*. Scripts have been provided which utilize pre-packaged libraries to support the visualization features of the Embedded Analytics Tool. Implementation has been provided for the generation of models which can be populated with data by users and exposed to Grafana through the *pushgateway*. Given the variety of data that may be generated by a function we refer to this data generally as in-function metrics. In-function metrics may be monitoring information, the results of AI/ML models or aggregated data. The exposer of this information is at the discretion of the function author as the aerOS template provides these capabilities as default. Specialized or additional templates may be required in the future to provide specific functionalities, such as data streaming or lightweight latency sensitive execution.

3.4.1.1. Related requirements

- TR-40 Support for non-centralised data processing
- TR-43 Support for data frugality and compliance with frugal AI paradigm
- R-P1-8 Real time dashboarding of processed and/or collected data
- R-P5-9 Data Analytics & Decision Making at the Edge

3.4.2. Structure diagram

The following figure illustrates the structure diagram of the Embedded Analytics Tool, including the relevant components that will be considered.

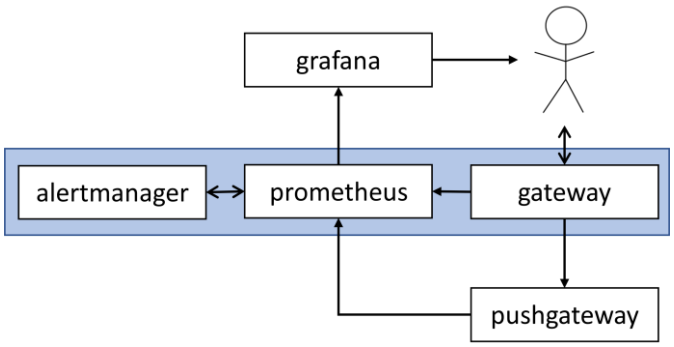

Figure 12. aerOS Embedded Analytics Tool architecture

Such components are identified and further described in the table below:

Table 17. Components for Embedded Analytics Tool

Component	Description	Interactions
gateway	The gateway hosts the deployed functions on the aerOS Embedded Analytics Tool. It exposes the functions through an HTTP API. The gateway also hosts a dashboard GUI.	The gateway provides connectivity for the deployed functions. Technical users will establish a HTTP connection to invoke functions and retrieve the results Non-technical users can utilise a GUI to invoke functions
prometheus	Prometheus provides monitoring of the aerOS Embedded Analytics Tool.	Prometheus monitors the gateway and pushgateway components through a scraping process on a regular interval. Metrics may be exposed through the Prometheus dashboard.

alertmanager	Alertmanager is a feature component of Prometheus. This allows for the creation of alert criteria around metrics gathered from the gateway and pushgateway component.	Alertmanager interacts directly with the Prometheus component. Alerts can be configured and communicated to external components when they are triggered.
pushgateway	Pushgateway is a feature component of Prometheus. This allows for the exposing of in-function metrics to Prometheus. As Prometheus scrapes metrics at an interval it is possible that several functions may execute and close within that time frame. Pushgateway hosts this information beyond the lifecycle of the function so it may be gathered.	Pushgateway communicates directly with executing functions on the aerOS Embedded Analytics Tool exposing their in-function metrics to Prometheus.
grafana	Grafana provides visualization features to the aerOS Embedded Analytics Tool.	Grafana connects directly to Prometheus as a data source. Exposing all metrics to visualization by Grafana. The Grafana dashboard may be used to visualise metrics from the aerOS Embedded Analytics Tool and previously executed functions.

3.4.3. Candidate technologies and standards

Table 18. Candidate technologies for aerOS Embedded Ananlytics Tool

Tech./Standard	Description	Justification
OpenFaaS	OpenFaaS is an open-source Function-as-a-Service platform which utilises containerised docker images as functions to be deployed, executed and managed.	Previous experience with OpenFaaS and extensive documentation of the platform reduced the learning curve when engaging with new software technologies. The features of OpenFaaS met the requirements of the aerOS Embedded Analytics Tool and when compared to alternative technologies such as jupyter notebooks, OpenFaaS proved a better fit to use as the basis for the aerOS Embedded Analytics Tool.
Prometheus	Prometheus is a popular open-source monitoring component in industry used to expose metrics of internal components and processes.	Prometheus is a well-recognised and proven monitoring solution in industry. Extensive documentation with the support of a strong development community. Prometheus is a leader in its domain.
Grafana	Grafana is an open-source interactive visualization tool. When linked to a data source Grafana provides diagrams and tables to represent the data.	Grafana like Prometheus is a well-recognised visualization solution. A strong community involvement continues the development of the Grafana tool and the utilization of dashboards and panels allow for flexible representation of data.

3.5. Trustworthiness and decentralized trust management

Figure 1 places the task T4.5 in the aerOS stack as an aerOS Basic Service named Cybersecurity and Trust. It is worth mentioning that this basic service will be addressed through two project tasks: Task T3.4 will focus on the cybersecurity components that support secure communications in the continuum, the global identity management and the cybersecurity approach aligned with DevPrivSecOps methodology, including authorization and authentication, while the part related to trust in aerOS corresponds to Task T4.5, which will be described in this section.

For that reason, this section describes the potential ways to ensure trustworthiness and decentralized trust management in aerOS. The proposed solutions are covered by the aerOS trust management approach and IOTA distributed ledger technology.

3.5.1. Trustworthiness of IEs in the continuum

The main objective of the trustworthiness functions that will be implemented in aerOS is to provide a high level of security in the whole aerOS ecosystem guaranteeing that malicious actions (e.g., unauthorized access, data breach, etc.) will be avoided at the highest-level degree. While the management of identities, authorization ad authenticated access is deployed under task T3.4 in WP3, trustworthiness is guaranteed in aerOS by deploying a distributed trust management approach that collects several attributes from IEs to calculate their trust scores as well as the trust score of the whole aerOS domain.

The aerOS Trust Manager is responsible for calculating the trust scores of the IEs and the aerOS domain using a set of pre-defined attributes. More specifically, to calculate the trust score, the Trust Manager retrieves attributes from the IEs via the NGSI-LD protocol, including but not limited to security events (from self-security), health score (from the health diagnose of self-*), running services, and firmware version. Moreover, the Trust Manager is an integral part of the aerOS basic/auxiliary services, therefore it is a critical component within the aerOS infrastructure that plays a vital role in managing trust relationships between the aerOS domains. Its primary responsibility involves employing a trust score calculation algorithm at two distinct orchestration levels: a) at the low orchestration level, where the Trust Manager calculates the trust score of all the IEs within the aerOS domain and b) at the high level, where the Trust Manager employs the IEs' trust scores to calculate the trust of the entire aerOS domain. In case that the trust score is below a threshold at either level, the corresponding IEs or aerOS domains are labelled as untrusted, resulting in restricted capabilities. Additionally, the Trust Manager handles the management of trust relationships and facilitates the sharing of trust information for IEs/domains using technologies such as the IOTA Distributed Ledger Technology (DLT) and other essential infrastructure components of aerOS (e.g., NGSI-LD). The trust score of each device with regards to security and privacy threats, and by extend of the whole aerOS domain, will be recalculated at run time, whenever necessary, in order to reflect the actual level of trustworthiness at any point in time.

Key features: i) Trust score calculation algorithm: Trust Manager calculates the trust of neighbouring IEs based on predefined attributes using the NGSI-LD protocol. ii) Retrieve IE attributes of the Neighbour IE: Communicate via NGSI-LD with the trust agent of a neighbour IE to retrieve predefined attributes.

3.5.1.1. Related requirements

- TR-17: Cross-layer cybersecurity
- TR-64: Cybersecurity tools
- TR-66: Trust Establishment
- TR-70: Distributed Trust Management

3.5.2. Trustful decentralized exchange: IOTA

Trust relationships in data circulation are of utmost importance. Once bonds of trust in a data sharing network are formed and malicious actors are excluded, confidentiality, integrity and accessibility are feasible. One of the ways leveraged in aerOS to achieve trust is by taking advantage of open-source edge technologies such as IOTA's distributed ledger Tangle framework.

IOTA is a distributed ledger technology (DLT) which shall allow for secure and trustworthy feeless data transactions between different nodes in the Tangle. The Tangle is a data structure replicated across a network of nodes (IE's in the continuum) that contains all the information necessary to track messages and ensure traceability of the payloads distributed across the network. While the Tangle and the more traditional blockchains have the same function of maintaining a traceable ledger state, the Tangle goes beyond and solves many of the difficulties blockchains face. The Tangle succeeds the blockchain as its next evolutionary step as it offers features suited to establish more efficient and scalable distributed ledger solutions. The Tangle describes a novel leaderless, probabilistic consensus protocol that enables parallel validation of transactions without requiring total ordering. This parallelization, along with being leaderless, the absence of intermediaries and the capability of working in an asynchronous setting is what makes IOTA an attractive solution, offering a highly performant consensus and ledger solution.

When an application that uses the protocol issues information payloads to a node, the node verifies the message and broadcasts it through the network if they are considered valid and follow the standard protocol specifications. When a node receives a valid message, it will send it to its direct neighbours using the gossip protocol, who in turn, send it to their own neighbours. Very quickly, every other node in the network sees the message and has the same information and the same knowledge of the "state" of the network at a given time. The messages sent by the nodes are meant to carry the most relevant data regarding the state of the network, this data could be, among others, the trust score of the different elements in the continuum. With this in mind, the deployment of IOTA's nodes will be done accordingly with both the domain and continuum requirements in mind.

3.5.2.1. Related requirements

- TR-17: Cross-layer cybersecurity
- TR-64: Cybersecurity tools
- TR-66: Trust Establishment
- TR-70: Distributed Trust Management

3.5.3. Structure diagram

The following figure illustrates the structure diagram of the trustworthiness and decentralized trust management, including the relevant components that will be considered.

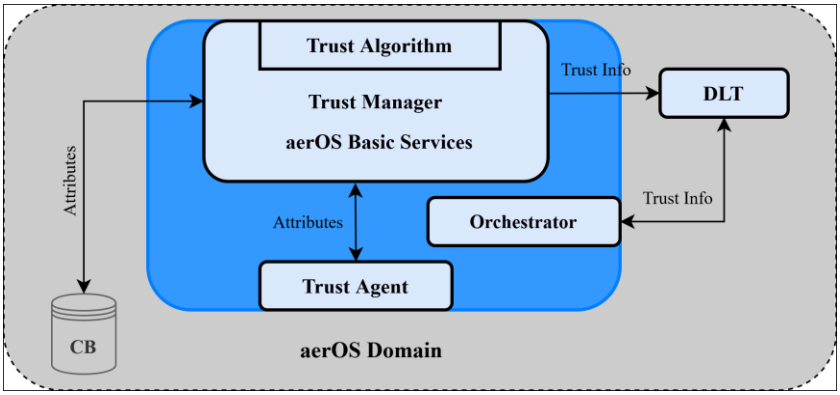

Figure 13. Overview of trust management structure

Such components are identified and further described in the table below:

Table 19. Components for trust management

Component	Description	Interactions
Trust Manager	Receiving the attributes from the Trust Agent, initiating the trust score calculation process and communicating with the orchestrator.	Communication with: Trust Agent to receive the attributes, IOTA to send/receive trust scores, context broker to collect attributes, and orchestrator to inform regarding whether the trust score is below a threshold.

Trust Algorithm	Implements a weighted function to calculate the trust scores based on the collected attributes.	It is located in the Trust Manager.
Trust Agent	Responsible for collecting the trust scores from the IEs and sending them to Trust Manager.	Interacts with the Trust Manager to send the collected attributes.
DLT	Distributing trust information	Interacts with Trust Manager to receive and distribute the trust information

3.5.4. Candidate technologies and standards

Table 20. Candidate technologies and standards for trust management

Technology/Standard	Description	Justification
IOTA	DLT that allows for secure and trustworthy feeless data transactions between different devices.	IOTA ensures traceability of the most relevant data transactions taking place between elements in the continuum, as well as making sure every node knows the state of the network.
X.509 certificates	A standard that defines the format of public-key certificates	Validating certificates during a TLS handshake for Kubernetes clusters
MQTTs	Exchanging messages between IoT devices and trust management	Establishing a secure communication between the IoT devices and trust manager to receive attributes for trust score calculation

3.6. Management services and aerOS management portal

The main aim of this task is the development of the aerOS Basic Service depicted in Figure 1 as aerOS Management Framework. This framework is composed by two independent components: the aerOS Management Portal and the aerOS Federator. These components are described separately in their respective subsections.

It should be pointed out that the task T4.6, which involves both the management services and aerOS management portal, has started in M10 of the project. As of the time of writing this document, it is currently in its third month of execution. Moreover, this task has a strong dependence on the other technical tasks of the project. Thus, the components described in this section are in a preliminary stage and are more open to further changes during its development process.

3.6.1. aerOS Management Portal

The aerOS Management Portal main purpose is to act as a single entry point for end users to manage and interact with the continuum. The dashboard will present valuable real-time information collected by the aerOS services, providing users with updates on the status of the continuum. This includes displaying the topology graph of added domains, deployed services, and the current state of domains and their IEs. It's important to note that the portal does not introduce new functionalities to aerOS but utilizes the existing capabilities of aerOS to fulfill user requests in terms of functionality.

In terms of the portal security, a clear set of roles and permissions will be established based on the AAA block of aerOS. This will determine the capabilities and actions available to each user of the portal. As a result, the content displayed in the dashboard will change depending on the role of the logged-in users. For instance, an administrator will have the rights and permissions to deploy specific services within their designated domains.

The portal will be deployed in a single domain of the continuum, also referred to as the "entrypoint domain". It will provide migration capabilities, allowing it to be relocated to a different domain or IE based on user needs

or unforeseen failures. This approach aligns with the principles of aerOS, emphasizing flexibility and decentralization, ensuring that the loss of the unique management entrypoint is avoided while maintaining a decentralized approach.

In terms of the Management Portal architecture, the dashboard, developed as a modern web application, acts as the single point of interaction with end users, acting as a frontend for managing operations related to the metaOS. It sends the user requests to the backend for accessing the requested information and executing desired actions, while also providing a web view to end users, presenting data obtained from the backend. The backend serves as the brain of the Management Portal. It translates user actions into specific processes and directly interacts with the frontend through appropriate communication protocols such as HTTP requests and WebSockets. This backend component also interacts with the aerOS Runtime and Basic Services to perform the requested actions. In order to maintain the decentralized nature of aerOS, an entry point balancer is required to distribute backend requests to specific aerOS Basic Services across different domains.

3.6.1.1. Structure diagram

The following figure illustrates the structure diagram of the aerOS Management Portal, including the relevant components that will be considered.

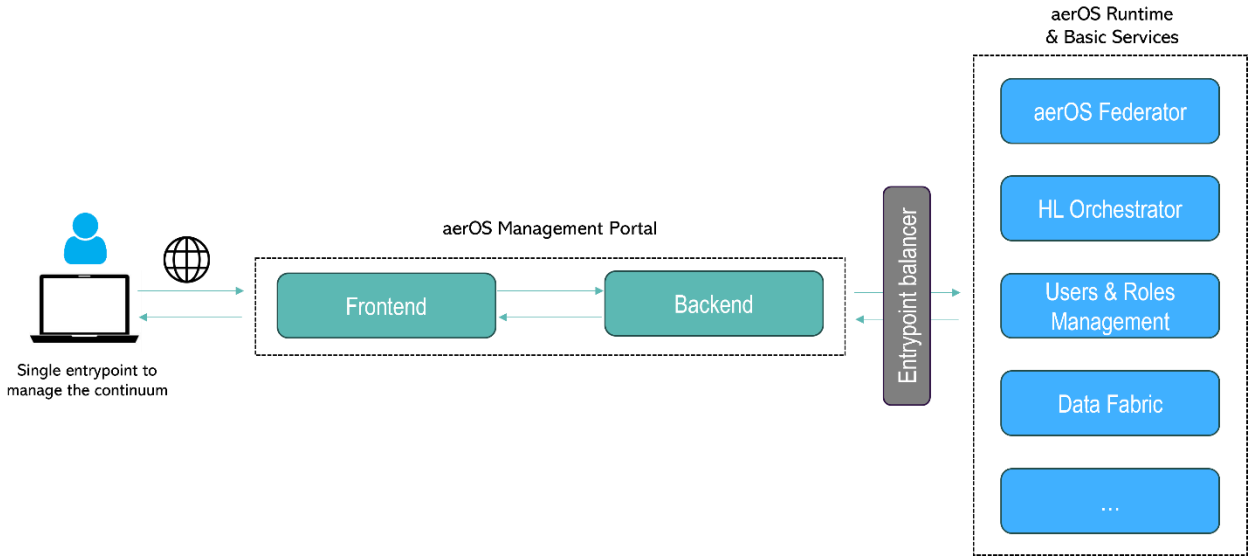

Figure 14. aerOS Management Portal architecture

Such components are identified and further described in the table below:

Table 21. Components for aerOS Management Portal

Component	Description	Interactions
Frontend	Frontend component of the Management Portal, it serves the web views, filled with data from the backend, to end users.	Single interaction point with end users (UI), it sends requests to the backend to prompt requested information and to perform requested actions.
Backend	Backend component of the web application, which acts as the brain of the Management Portal: translates the actions performed by users into specific processes.	Direct interaction with the frontend using appropriate communication protocols (HTTP requests, WebSockets, …) as part of the aerOS Management portal, and with the aerOS Runtime and Basic Services, also through the entrypoint balancer when needed.
aerOS Runtime & Basic services	The aerOS Runtime and Basic Services offered by the metaOS as	The backend component interacts with these components to perform the requested actions

	described in the architecture.	by users in the UI.
Entrypoint balancer	The aerOS Management Portal acts as a single entrypoint for managing the continuum, but it needs to avoid centralization because of the decentralized nature of aerOS. For that reason, a component is needed to distribute the requests to certain aerOS Basic Services (e.g., HLO) between different domains.	This component is only called by the backend to forward its request to a Basic Service instance of a domain.

3.6.1.2. Candidate technologies and standards

Table 22. Candidate technologies and standards for the aerOS Management Portal

Technology/Standard	Description	Justification
Vue.js	Vue.js is an open-source JavaScript framework for building web user interfaces.	Vue.js is currently one of the most popular frameworks to build web applications along with React, so it was decided to take advantage of previous experience in the technology by the partners involved in the task. In addition, a UI library (Vuetify, Element, Vue Material Kit, …) should also be used to polish the visual appearance of the Management Portal.
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.
Load balancing algorithm	Load balancers try to efficiently distribute network traffic across multiple final endpoints, following certain configurable rules.	Certain requests from the portal's backend must be forwarded to one instance in a domain of the needed aerOS Basic Service following a distributed approach. For instance, a service deployment request must be forwarded to the HLO of a domain but avoiding to always send the request to the same domain. It is envisaged to use a state-of-the-art open-source algorithm as a starting point to develop a specific component for aerOS needs.

3.6.2. aerOS Federator: domain registration and discovery

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. It is crucial to emphasize that aerOS operates on a decentralized model, and thus, by leveraging the distributed state repository mechanisms (described in detail in D2.1) the aerOS Federator will be capable to establish the appropriate mechanisms required to achieve a federated architecture across the aerOS domains. Specifically, the aerOS Federator encompasses the functionalities related to domain registration and discovery.

In the context of management services, the operations performed by the aerOS Federator are not directly initiated by end users, as the ones described in the previous section regarding the use of the Portal. Instead, the

aerOS Federator performs smart and automatically conducted actions, triggered by specific user actions like creating new domains or modifying IEs that belong to an existing domain.

To accomplish the intended process, there are two elements that engage with the aerOS Federator: (i) the Context Broker, which manages real-time contextual information for entities in aerOS, representing and monitoring the continuum, and (ii) the Context Sources, which allow an NGSI-LD broker to access location information and extract additional context information from other registered context brokers or context sources.

Considering the previously mentioned aspects, the architecture of the aerOS Federator comprehensively covers the functionalities related to domain registration and discovery. The Domain Discovery component of the aerOS Federator ensures consistency in the federation of context brokers by reacting to changes in the continuum, updating their corresponding Domain Registry, and then spreading registration update messages to other domains. Meanwhile, the Domain Registry oversees the maintenance of registrations in the Context Sources, constantly interacting with them based on indications from the Domain Discovery.

3.6.2.1. Structure diagram

The following figures illustrate the structure diagram of the aerOS Federator, both within a single domain and across multiple domains, including the relevant components that will be considered.

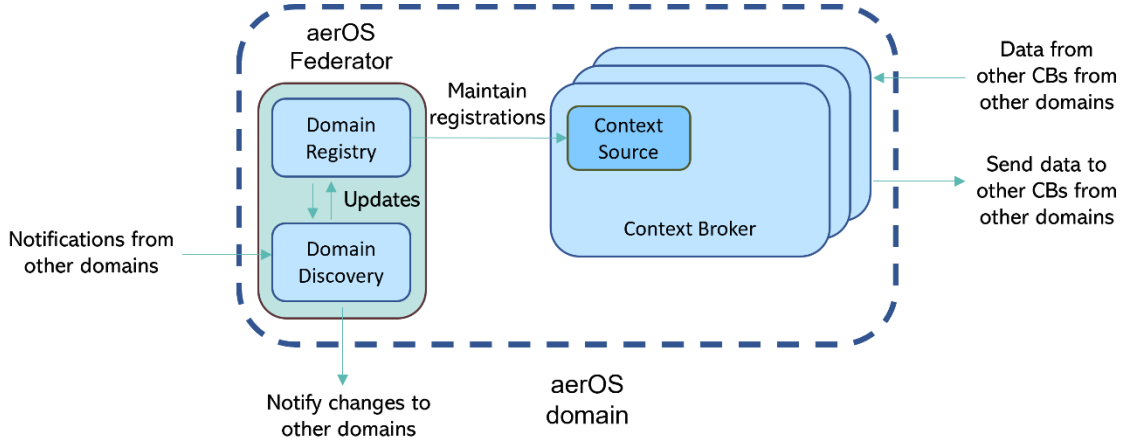

Figure 15. aerOS Federator architecture in a single domain

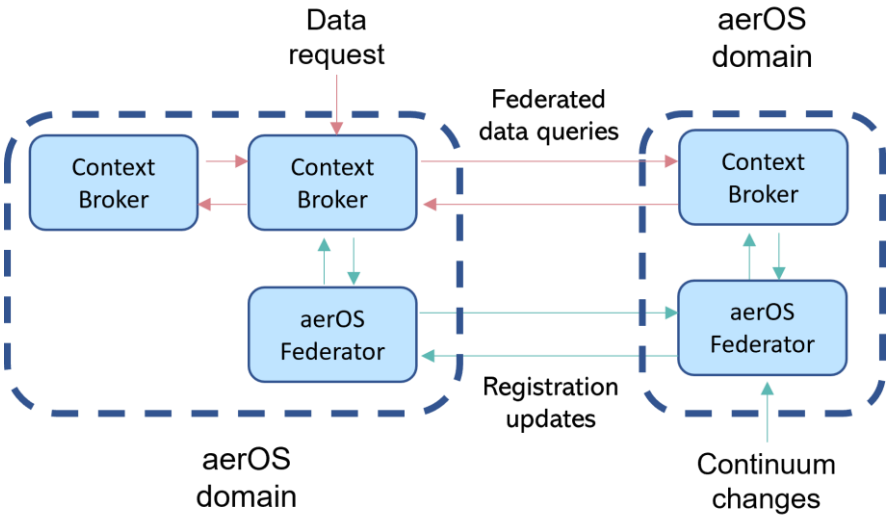

Figure 16. aerOS Federator architecture in multiple domains

Such components are identified and further described in the table below:

Table 23. Components for aerOS Federator

Component	Description	Interactions
aerOS Federator	Architectural block of aerOS which is in charge of establishing the federation among the existing aerOS domains. In this case, only its features related to domain registration and discovery are covered.	Its interactions are described in depth in the following Domain Discovery and Domain Registry entries in this table.
Domain Discovery	The Domain Discovery component of the aerOS Federator reacts to the notified changes in the continuum. First in a local way by updating the Domain Registry, and then from a global point of view by spreading update messages to other domains. Its main purpose is to maintain consistency in the federation of all the context brokers along the continuum.	This component listens to other Domain Discoveries from other domains and also sends requests to them. In addition, translates the received updates to specific instructions to be sent to the Domain Registry of its domain.
Domain Registry	The Domain Registry component of the aerOS Federator oversees the maintenance of registrations in the Context Sources of the Context Brokers.	The aerOS Federator constantly interacts with the Context Brokers to maintain (add, update and delete) the registrations of their Context Sources, always reacting to the indications of the Domain Discovery.
Context Broker	Component responsible for managing and providing real-time contextual information about various entities and their environments. In aerOS, the continuum will be represented and monitored through this contextual information.	The Context Broker interacts with other Context Brokers and with the aerOS Federator of its domain. Moreover, it receives data requests. The interactions regarding data are described in depth in the section 3.2.
Context Source	The Context Source is the element that enables an NGSI-LD broker to be informed about the location of additional (non-local) entities and extract additional context information from other registered brokers or context sources.	The Context Source works together with the Context Broker to properly retrieve the requested entities.

3.6.2.2. Candidate technologies and standards

Table 24. Candidate technologies and standards for the aerOS Federator

Technology/Standard	Description	Justification
Orion-LD	Open-source implementation of the NGSI-LD context broker.	Orion-LD provides additional features to the Orion-NGSIv2 implementation, which are needed in the project. Furthermore, partners have expertise on the use of Orion, and its main developer and maintainer is a partner of

		aerOS (FIWARE Foundation).
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 wide range of key capabilities that are not fully identified at this moment. 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 programming language in which they are most confident, or they think it is more appropriate. In addition, some capabilities would not be required in each domain or at every point in time.

4. Conclusions

During the past months, the partners of the Consortium have focused their effort on developing the design specifications that will facilitate the realization of the aerOS architecture. The preliminary proposed solutions for each task have been formulated and respective technical components have been identified. Moreover, candidate technologies have been selected and investigated with respect to technical requirements that have been identified so far. From the work within WP4, the following (more specific) insights have been extracted:

- AI in the project needs to be analysed from both external and internal perspective. External should met the requirements coming from pilots and possible aerOS usage scenarios triggered by external actors, whereas internal should be more focused on utilising AI in intelligent decision-making to manage the continuum. Both may require mechanisms to address frugality and support explainability.
- For embedded analytics OpenFaaS turned out to be a good match according to the requirements also providing options for visualization. However, moving OpenFaaS from Docker Swarm distribution to Kubernetes would require a substantial effort, since interfaces between components would require reconfiguration and testing to ensure connectivity in different domains.
- The flexibility offered by FaaS frameworks can be challenging to manage, this has led to several internal discussions (to be continued guided by experiences of use cases) on the topic of best practices for function authoring and how we expect functions to behave as part of the aerOS.

5. Future Work

During this project stage, aerOS has mainly focused on the design phase, which involves the analysis of the requirements, definition of the use cases, and selection of technologies. The successful completion of this phase has resulted in the definition of the initial version of the aerOS architecture and the specification and initial proposal of the aerOS WP3 and WP4 software components. Consequently, this influence is reflected in the work carried out in WP4 during this period, as well as in the results presented in this document. Specifically, D4.1 is centred around the preliminary release of WP4 proposed solutions. WP4 aims to optimize the usage of data in aerOS without sacrificing control over it. To achieve this goal, solutions for data autonomy and semantic interoperability has been defined to drive the processing of data, utilizing decentralized frugal AI and embedded multiplane analytics. In addition, the WP4 addresses emerging data sovereignty and trust challenges and the definition of an aerOS management framework.

Deliverable D4.1 is part of an iteration of living deliverables of WP4. This document is the first version, which is due to M12 of the project and presents work done so far. Each of these deliverables aim at releasing the outcomes related to the software for delivering intelligence at the edge, which occurs during intermittent time periods. The second iteration of the deliverable is planned to deliver the intermediate release in M18, and the third iteration is planned to deliver the final release in M30. Since this deliverable will have two more subsequent iterations, the specifications included here may be extended or updated as the project evolves. Solutions presented were classified according to WP4 tasks division and described using a template to maintain a common structure of information given. It covers description and main functionalities, a high-level component diagram, a component description table, and candidate technologies for their implementation. In the next iteration (D4.2) of the deliverable, the templates of each solution will be updated to include additional information, such as usage stories. In addition, the next iteration will already include actual software as first outcomes of WP4 tasks.

Regarding the future work, the finalization of this deliverable will boost the already initiated development of related software. Next tasks will be to:

- Fill in missing information in the design of specific components (e.g., communication interfaces),
- Conduct any necessary adjustments (e.g., slight modification in provided functionalities, change in structure, refinement of selected technologies),
- Establish needs for interactions between WP3 and other WP4 components,
- Preparation of the backlog of tasks, distribution of work and kick-off of implementation activities,
- First software results (MVP) ready and available by M18 (however, the degree of development is foreseen not be homogeneous for all the components).