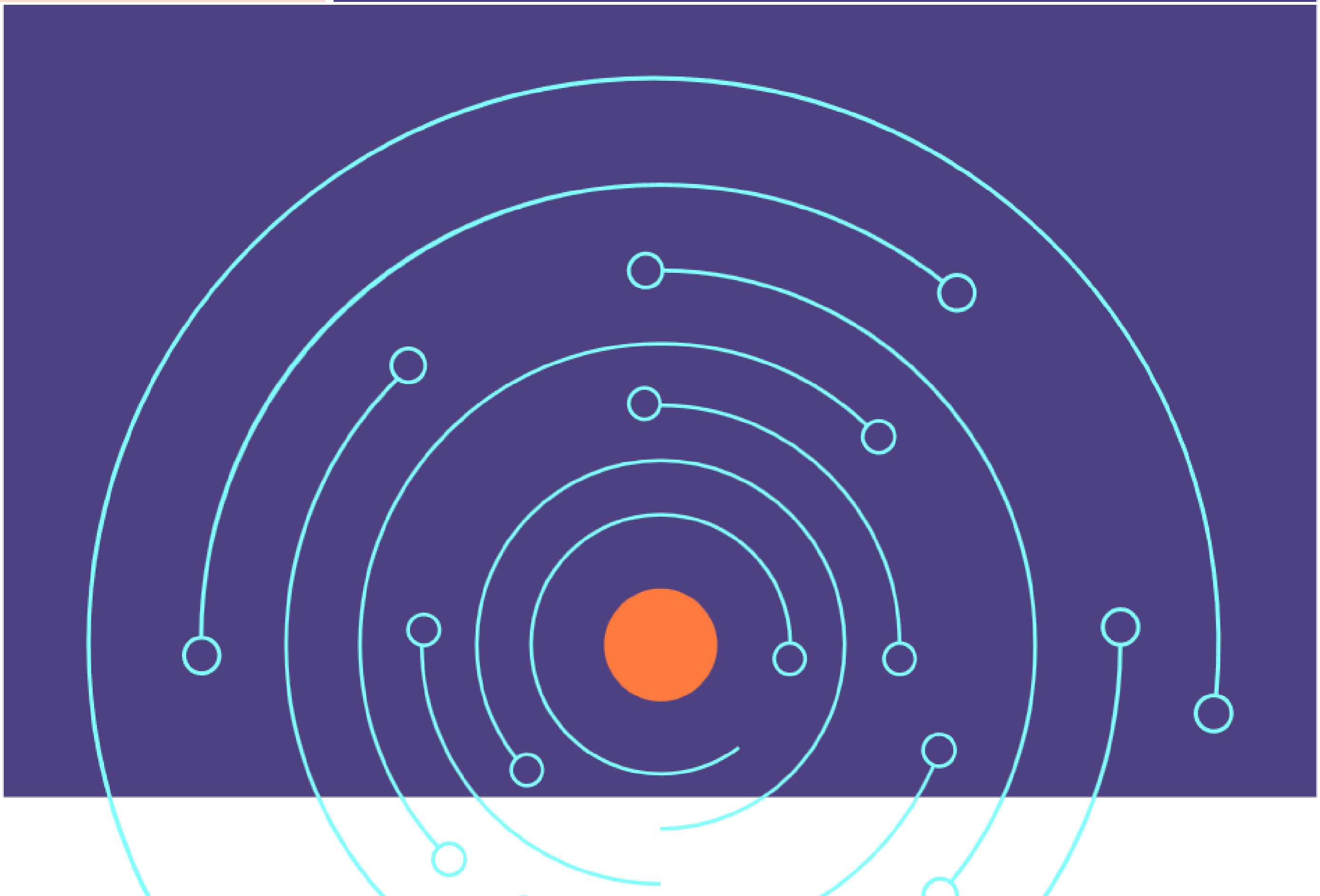
RE4DY
MANUFACTURING DATA NETWORKS

Title	D3.2: First generation digital continuum 4.0 open toolkit
Document Owners	INTRA
Contributors	ATOS, UPV, CERTH, UNI, ATLANTIS, CORE, ENG, NOVA, ICF, S21SEC, CNR, SIEMENS
Dissemination	Public
Date	22/12/2023
Version	V.1.0



Document Status

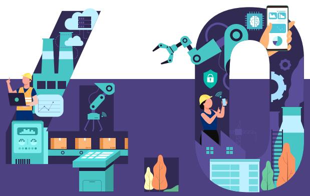
Deliverable Leader	INTRA
Internal Reviewer 1	INNO
Internal Reviewer 2	KUL
Work Package	WP3: Continuity Management Toolkit for Industrial Digital Thread & Cognitive Twin Fabrics
Deliverable	D3.2: First generation digital continuum 4.0 open toolkit
Due Date	31/11/2023
Delivery Date	04/01/2024
Version	V1.0

Version history

10/05/2023	Table of Contents
15/11/2023	Contributions on Technical Descriptions
20/11/2023	Contributions on Experiment Plans
11/12/2023	Version for Review
22/12/2023	Final Version

Project Partners

#	Acronym	Participant organisation name
1	INNO	ASOCIACIÓN DE EMPRESAS TECNOLÓGICAS INNOVALIA
2	CHAL	CHALMERS TEKNISKA HOGSKOLA AB
3	IDSA	INTERNATIONAL DATA SPACES EV
4	VWAE	VOLKSWAGEN AUTOEUROPA, LDA
5	CEIT	ASSECO CEIT AS
6	UNI	UNINOVA-INSTITUTO DE DESENVOLVIMENTO DE NOVAS TECNOLOGIAS-ASSOSIACAO
7	FILL	FILL GESELLSCHAFT MBH
8	AVL	AVL LIST GMBH
9	VIS	VISUAL COMPONENTS OY
10	UMH	UNIVERSIDAD MIGUEL HERNANDEZ DE ELCHE
11	ATLANTIS	ATLANTIS ENGINEERING AE
12	DATA	DATAPIXEL SL
13	CORE	CORE KENTRO KAINOTOMIAS AMKE
14	UiO	UNIVERSITETET I OSLO
15	AVIO	GE AVIO
16	ENG	ENGINEERING-INGENIERIA INFORMATICA SPA
17	POLIMI	POLITECNICO DI MILANO
18	AtoS	ATOS IT SOLUTIONS AND SERVICES IBERIA SL
18.1	AtoS-ES	ATOS SPAIN SA
19	KU	KATHOLIEKE UNIVERSITEIT LEUVEN
20	INTRA	NETCOMPANY-INTRASOFT SA
21	NOVA	NOVA ID FCT - ASSOCIACAO PARA A INOVACAO E DESENVOLVIMENTO DA FCT
22	ICF	INDUSTRY COMMONS FOUNDATION (INSAMLINGSSTIFTELSE)
23	CERTH	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS
24	S21SEC	GRUPO S 21SEC GESTION SA
25	UPV	UNIVERSITAT POLITECNICA DE VALENCIA
26	CNR	CONSIGLIO NAZIONALE DELLE RICERCHE
27	SANDETEL	SOCIEDAD ANDALUZA PARA EL DESARROLLO DE LAS TELECOMUNICACIONES SA
28	SSF	SWITZERLAND INNOVATION PARK BIEL/BIENNE AG
29	GFMS ADVMAN	GF MACHINING SOLUTIONS AG
30	Fraisa SA	FRAISA SA
31	SIE	SIEMENS SCHWEIZ AG



Executive Summary

Deliverable 3.2, First generation digital continuum 4.0 open toolkit is the second deliverable of work package 3 ("Continuity Management Toolkit for Industrial Digital Thread & Cognitive Twin Fabrics"). This document will report on the development of the first generation of the digital 4.0 continuum open toolkit.

Following the initial description of the different tools (commercial and open source) brought to the project by partners and their mapping to the IDSA building blocks provided in the previous deliverable of WP3, namely D3.1, this document focuses mainly on two aspects. Firstly, the detailed technical description of the current version of the tools that comprise the first-generation digital continuum 4.0 open toolkit and secondly the experimentation plan for the qualification of the toolkit on the network of TEFs and pilot sites of the RE4DY project.

The technical description of the different tools serves as a basis for the integration activities that are ongoing within WP3. The result of these activities will be an integrated digital 4.0 continuum open toolkit. Currently, the toolkit consists of 26 tools that fulfil the specific WP3 objects.

The experimentation plan includes a set of 5 experiments and 14 scenarios to be executed in the relevant TEFs or pilot sites. Each experiment includes one or more scenarios, each one involving a set of tools. The described scenarios test not only the functionalities of the independent tools but also the effectiveness of the interactions between them and their integration.

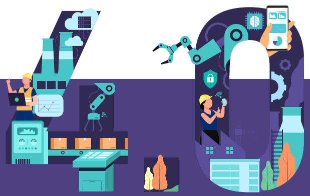
Finally, each experimentation scenario includes a number of KPIs that will enable an effective qualification of the toolkit.

1 Contents

List of Figures

Acronyms

3D	Three Dimensional
AAS	Asset Administration Shell
ABAC	Attribute-Based Access Control
ACL	Access Control List
AF	Application Function
AI	Artificial Intelligence
API	Application Programming Interface
BART	Bidirectional and Auto-Regressive Transformers
BDA	Big Data Analytics
BERT	Bidirectional Encoder Representations from Transformers
BM25	Best Match 25
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CNC	Computer Numerical Control
CNN	Convolutional Neural Network
CO2	Carbon Dioxide
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
CSW	Catalog Services for the Web
DAG	Directed Acyclic Graph
DB	Database
DC	Data Container
DCP	Data Connection Profile
DIDI	Dataspace for Industrial Data Intelligence
DIN	Director Identification Number
DL	Deep Learning
DOI	Digital Object Identifier
DSS	Data Security Standard
DaaP	Data as a Product
EDC	Eclipse Data Connector
eIDAS	electronic Identification, Authentication and Trust Services
ERP	Enterprise Resource Planning
ETL	Extract, Transform, Load
FAIR	Findability, Accessibility, Interoperability and Reusability
FEDMA	Federated Maintenance for Milling Machines
FL	Federated Learning
FML	Federated Machine Learning



FPdM	Federated Predictive Maintenance
FTP	File Transfer Protocol
GPSI	General Purpose Serial Interface
GPT	Generative Pre-trained Transformer
GUI	Graphical User Interface
HDFS	Hadoop Distributed File System
HMI	Human Machine Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
I/O	Input/Output
ICCID	Integrated Circuit Card ID
IDSA	International Data Spaces Association
IDSA RAM	International Data Spaces Association Reference Architecture Manual
IIoT	Industrial Internet of Things
IMSI	International Mobile Subscriber Identity
IPC	Inter-Process Communication
IoT	Internet of Things
JS	JavaScript
JSON	JavaScript Object Notation
K8S	Kubernetes
KPI	Key Performance Indicator
LLM	Large Language Model
MES	Manufacturing Execution System
MK	Machine Kit
ML	Machine Learning
MPD	Media Presentation Description
MQTT	Message Queuing Telemetry Transport
MRD	Meta Repository Demonstrator
MRL	Manufacturing Resource Library
MS	Microsoft
NC	Numerical Control
NCK	Numerical Control Kernel
NEF	Network Exposure Function
NOVAAS	NOVA Asset Administration Shell
OAI-PMH	Open Archives Initiative Protocol for Metadata Harvesting
OCES	Ontology Commons EcoSystem
OMP	On Machine Probing
OPC-UA / OPCUA	Open Platform Communications United Architecture
OS	Operating System

OSS	Open-Source Software
P2P	Peer-to-Peer
PCF	Policy Control Function
PCS7	Process Control System 7
PDU	Packet Data Unit
PIN	Personal Identification Number
PLC	Programmable Logic Controller
PP	Post Processor
QoS	Quality of Service
REST	Representational State Transfer
RFC	Request for Comments
RMVM	Run MyVirtual Machine
ROS	Read-Only Storage
RPC	Remote Procedure Call
RRM	Radio Resource Management
SCADA	Supervisory Control And Data Acquisition
SDL	Session Description Language
SDK	Software Development Kit
SDP	Session Description Protocol
SEAL	Service Enabler Architecture Layer
SFC	Shop Floor Connect
SFTP	Secure File Transfer Protocol
SIM	Subscriber Identity Module
SMF	Session Management Function
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SPN	Service Provider Name
SQL	Structured Query Language
SSL	Secure Sockets Layer
STL	Standard Triangle Language
SVM	Support Vector Machine
TC	Teamcenter
TCP	Transmission Control Protocol
TDS	Trusted Data Sources
TEF	Testing and Experimentation Facilities
TIA	Totally Integrated Automation
TLS	Transport Layer Security
UE	User Equipment
UI	User Interface
UML	Unified Modeling Language

UPF	User Plane Function
URL	Uniform Resource Locator
UX	User Experience
VC	Verified Credential
VCP	Visual C++ Project
WP	Work Package
XAI	Explainable Artificial Intelligence
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

1 Introduction

1.1 Purpose and scope

The main scope of WP3 ("Continuity Management Toolkit for Industrial Digital Thread & Cognitive Twin Fabrics") is to establish a network of Testing and Experimentation Facilities (TEFs) and European Data Spaces for qualifying digital 4.0 continuum open systems & OSS tools.

The current document is the second deliverable of WP3, following the submission of D3.1 "Resilient & sustainable "Data as a Product" computing and data space", where the process of TEF network establishment was presented together with the methodology describing how the RE4DY Project will benefit from this network in order to foster reaching a resilient and sustainable "Data as a Product" computing and data space.

Building on the results presented in D3.1, the current document paves the way for the delivery of the qualified the digital 4.0 continuum open toolkit and its final assessment in the pilot sites and TEFs and the RE4DY value network.

The deliverable provides a detailed technical description of the current version of the tools that comprise the first-generation digital continuum 4.0 open toolkit. The detailed technical description of the different tools includes details like the type of inputs and outputs of each tool, its main functionalities, the internal architecture and the APIs they provide that are going to serve as integration points.

Also, the deliverable introduces an experimentation plan for the qualification of the toolkit on the network of TEFs and pilot sites of the RE4DY project. Each experiment includes a set of experimentation scenarios accompanied with the definition of the related components and datasets/AI models, the proposed added value and a set of KPIs.

Finally, an OpenAPI definition of a set of tools included in the toolbox has been collected in the form of YAML files to support the integration activities of WP3.

1.2 Relation to other project WPs and Tasks

First generation digital continuum 4.0 open toolkit, as described in the current document will serve as an input to WP4 "Large-Scale Trials of Resilient Smart Connected Factory 4.0 Process Engineering" and WP5 "Large-Scale Trials of Resilient Smart Connected Factory 4.0 Process Operations" which perform the early assessment of RE4DY data pipelines and toolkit on business impact.

1.3 Structure of the document

The document is structured as follows:

Section 1 serves as an introduction, providing the purpose and scope of the current deliverable, its structure and its relationship with other deliverables and other project work packages and tasks.

Section 2 provides a detailed technical description of each tool included in the first version of the open toolkit.

Section 3 presents the experimentation plan for the qualification of the toolkit on the network of TEFs and pilot sites.

Finally, Section 4 summarizes the results presented in this deliverable and provides the final conclusions.

2 RE4DY Data Space and Toolware Catalogue: Technical Description

This section offers a comprehensive description of the technical specifications of each tool incorporated within the first-generation digital continuum 4.0 open toolkit. Each entry in this collection provides a short description of the tool, information related to its input and output, its main information flows and its internal architecture. Furthermore, it describes any APIs that are or will be made available and provides information regarding the implementation of the tool, such as the programming languages, platforms and libraries used. This information serves as a basis for the integration actions that will turn this set of tools into a qualified digital continuum 4.0 open toolkit.

2.1 Component Name: Knowledge Graph Visualization Environment

2.1.1 Description

The knowledge graph visualization environment aims to visualize knowledge graphs of different types and supports the presentation of flows and interconnected data (an example is supply chains). The JS/TypeScript components can present significant data volumes through interactive graphs.

2.1.2 Input

The input to the JS/TypeScript components is data formatted as JSON text and CSV.

2.1.3 Output

The output is an interactive graphical model describing knowledge graphs and/or flows the input data represents.

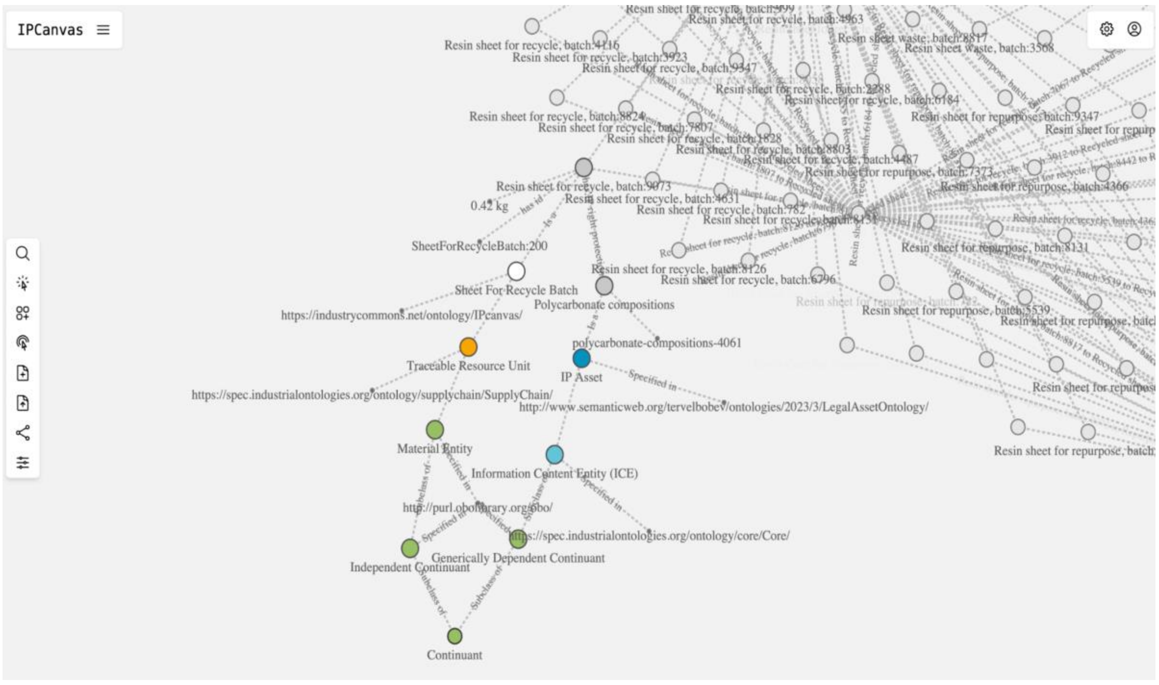

IPCanvas
SheetForRecycleBatch:200
Sheet For Recycle Batch
Polycarbonate compositions
polycarbonate-compositions-4061
https://industrycommons.net/ontology/IPcanvas/
Traceable Resource Unit
IP Asset
https://spec.industrialontologies.org/ontology/supplychain/SupplyChain/
http://www.semanticweb.org/tervelbobev/ontologies/2023/3/LegalAssetOntology/
Material Entity
Information Content Entity (ICE)
http://purl.obolibrary.org/obo/
https://spec.industrialontologies.org/ontology/core/Core/
Independent Continuant
Generically Dependent Continuant
Continuant
0.42 kg
Resin sheet for recycle, batch:4116
Resin sheet for recycle, batch:3923
Resin sheet for recycle, batch:9347
Resin sheet for recycle, batch:8824
Resin sheet for recycle, batch:9073
Resin sheet for recycle, batch:8126
Resin sheet for recycle, batch:6796
Resin sheet waste, batch:8817
Resin sheet waste, batch:3568
Resin sheet for repurpose, batch:8131
Resin sheet for repurpose, batch:5539

Figure 1: Interactive Graphical Model of the Knowledge Graph Environment

2.1.4 Information Flow

To use the environment after installation, the user loads and prepares the data from external sources and adapts it to JSON or CSV files/variables. These are sent to the JS/TypeScript component that uses the data to create the knowledge or flow graphs. In the resulting component graph window, the user can instruct the component to print or copy the graph for external documentation. The graph allows for interactive operation – allowing the user to interact with data points in the graph to connect to other data sources – systems for storing attributes etc.

2.1.5 Internal Architecture

The knowledge graph visualization environment consists of a JS/TypeScript application with a defined API for data and results management and presentation.

2.1.6 API

The environment is a visualization environment that provides visualization components in different solutions – it does not contain an API for external systems access. Still, it has only an API for local use.

2.1.7 Implementation technology

GUI: JavaScript, https://react.dev/, https://vuejs.org/. https://headlessui.com/, https://tailwindcss.com/

Backend: https://nextjs.org/, https://expressjs.com/

Graphics Programming: https://threejs.org/, https://deck.gl/, https://luma.gl/

2.1.8 Comments

None

2.2 Component name: Meta Repository Demonstrator

2.2.1 Description

The Meta Repository Demonstrator (MRD) is a management infrastructure for related data and meta-data for objects, knowledge resources, interconnectivity resources and relationships. MRD allows the management of a large type of information resources of different types and contents that can be extended over time to meet future challenges. The MRD can be used to demonstrate a simple Digital Twin management system during the RE4DY project.

2.2.2 Input

As a Digital Twin, the resources during the demonstration are manually added (option if needed - import of existing data) and linked to other relevant objects in the installation. Beyond industry-specific information types, data can be entered in the form of media, external links to web-based data, and external databases and services in specific cases. External data storage (measurement and time-series data) is an example of an external service.

2.2.3 Output

End-user web-based GUI for desktop systems (optionally, if time allows also mobile device GUI). Access to data via REST calls and direct database access calls is optional. The end-user environment can act as an interface to other externally connected systems. Report and export functionalities.

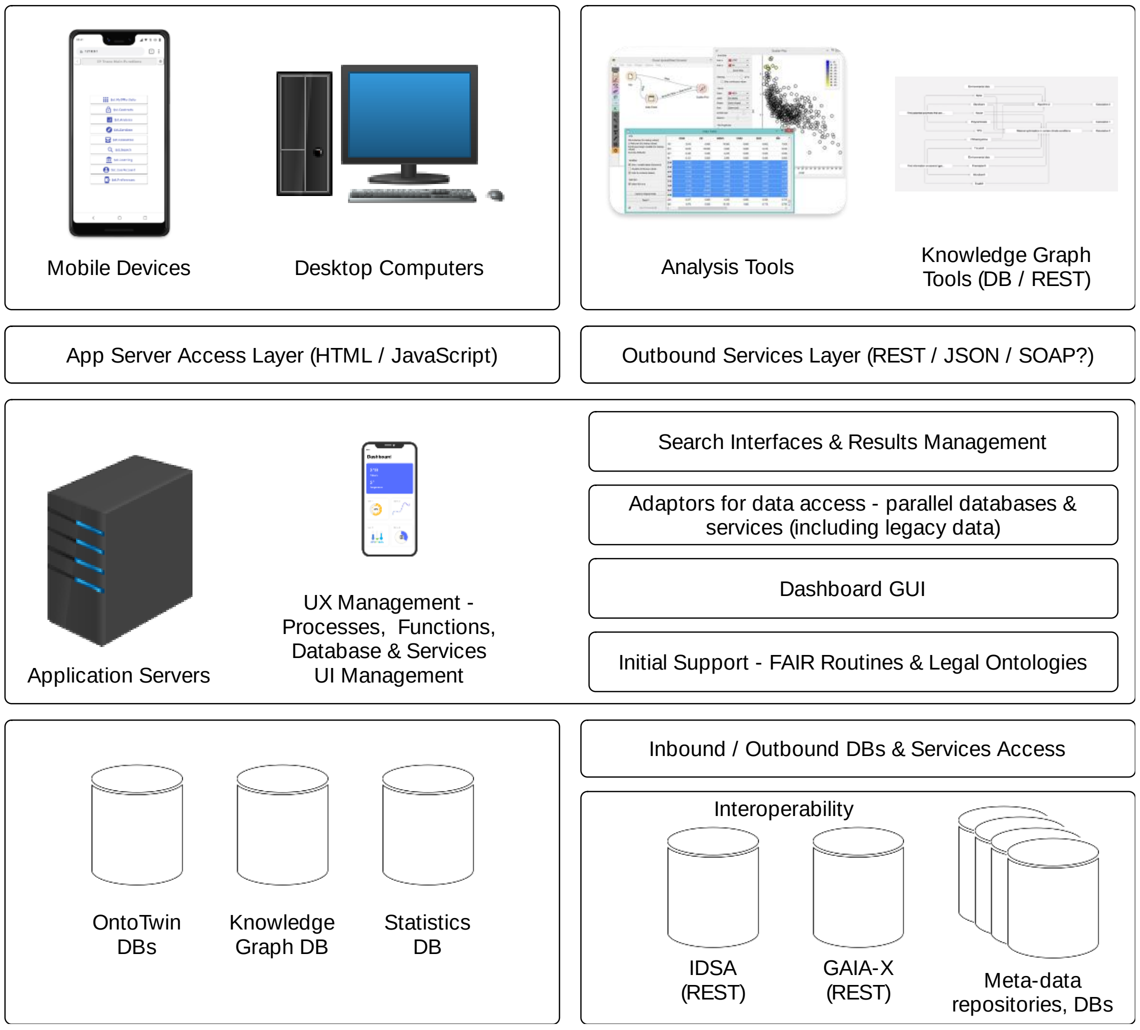

Mobile Devices
Desktop Computers
Analysis Tools
Knowledge Graph Tools (DB / REST)
App Server Access Layer (HTML / JavaScript)
Outbound Services Layer (REST / JSON / SOAP?)
Search Interfaces & Results Management
Adaptors for data access - parallel databases & services (including legacy data)
Dashboard GUI
Initial Support - FAIR Routines & Legal Ontologies
Application Servers
UX Management - Processes, Functions, Database & Services UI Management
Inbound / Outbound DBs & Services Access
Interoperability
OntoTwin DBs
Knowledge Graph DB
Statistics DB
IDSA (REST)
GAIA-X (REST)
Meta-data repositories, DBs

Figure 2: GUI of Meta Repository Demonstrator

2.2.4 Information Flow

MRD can be regarded as a resource hub architecture designed to effectively support large-scale P2P implementation, storing, and managing a multitude of data and information that can be linked freely between themselves and with data stored in external (also legacy) data.

2.2.5 Internal Architecture

The system is separated into three main functionalities: database manager, application server and interconnected end-user devices of different types.

2.2.6 API

Beyond the end-user GUI devices of different types, the main APIs for access are REST and, in relevant cases, direct-controlled DB access or SOAP/XML.

2.2.7 Implementation Technology

End-user systems: webpage-based GUI for desktop and mobile devices, native applications for Android and iOS (mobile phones and tablets).

REST access (JSON)

Application server: Omnis Studio application server (omnis.net) for managing end-user devices and internal/external database and services access.

Database server: PostgreSQL database server. Used for both core database systems and for supplementary/supporting databases such as Knowledge Graph (pgGraph), Vector (LLM) database (pgVector) and statistical support/time series database systems.

2.2.8 Comments

Omnis Studio was chosen as a development tool for application server environments and end-user systems due to its effective pilot development processes. The application server layer can later be developed in other development environments using the now-developed system as a specification example.

2.3 Component Name: OCES - Ontology Commons EcoSystem

2.3.1 Description

An ecosystem of ontologies for cross-domain interoperability and knowledge management supporting more effective and trustworthy data and information exchange and interoperability in industrial settings.

2.3.2 Input

Structured and validated data to be tagged and registered into ontologies to prepare the data for improved interoperability and quality control.

2.3.3 Output

Data and information tagged and structured for improved interoperability, quality and usability in industrial processes.

2.3.4 Information Flow

Tools, methodologies, and approaches used in the OCES environment involving multiple tools and development processes are presented below.

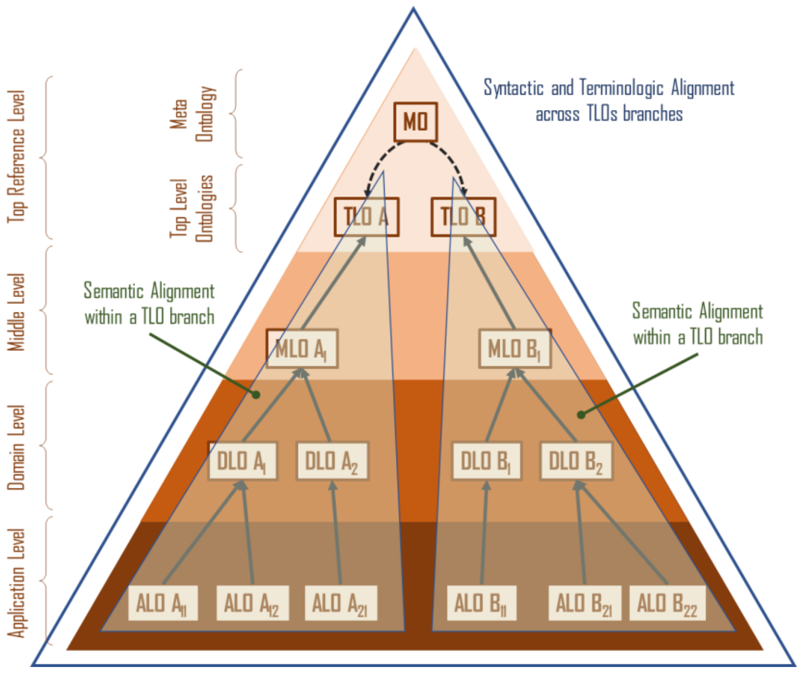


Figure 3: Tools, methodologies, and approaches of the OCES environment

2.3.5 Internal Architecture

The OCES environment consists of several software systems, collaborative environments and standards, most of them Open Source, that are used to manage ontologies.

2.3.6 API

Access to relevant ontologies and ontology development efforts can be through the OntoCommons ontology registry infrastructure in final development.

2.3.7 Implementation Technology

The OCES is implemented through standard ontology development tools.

2.3.8 Comments

Currently, some components and tools in OCES are in final development, while others are standard tools on the market for ontologies development.

2.4 Component Name: Federated Predictive Maintenance (FPdM)

2.4.1 Description

The FPdM component is expected to be a multidisciplinary software solution equipped with predictive functionalities, which are enabled by a set of microservices or sub-components. The range of predictive and monitoring capabilities that will be exploited within the life of the project will respond to the pilots' business cases, and will include amongst others, fault/defect identification and remaining useful life estimation of tools. Apart from the whole FPdM component, the aforementioned services will also be offered individually to facilitate collaboration with different Federated Frameworks (such as Alida).

The FPdM as a whole component is set up as a complete platform, integrated into the RE4DY architecture, collaborating with other platforms/infrastructures and enhanced with Federated Learning capabilities. The microservices on the platform can function as separate tools in a toolbox, ready to be activated when a specific need arises. The microservices will use analytic models and advanced Machine Learning algorithms to provide the following functionalities: 1. The identification of product or equipment defects (e.g. material surface defects) based on specific and predefined failure patterns 2. The remaining useful life of tools calculation based on the level of their wear and quality condition.

2.4.2 Input

FPdM based on the current architecture and information flow, is expected to receive pilot metadata through specialized connectors (such as True Connector), or potentially experimental data from the TEFs. Input data will be pre-processed with specially designed algorithms in ways that serve the data analysis mechanisms.

2.4.3 Output

As an output, the FPdM solution will be in the position to successfully identify known errors and defects, detect potential equipment failures/anomalies and/or monitor the remaining life of specific and preselected tools. The output of the FPdM solution will become more specific and will depend on how the Pilots' requirements will evolve within the project, quantity and quality of the available data and analysis outcome.

In the cases where the FPdM will be utilized as a complete solution, it can provide visualization of the analysis results with the use of Grafana dashboards, customizable panels for selected viewing of the results, smart notification systems (e.g. e-mail) and analysis publishing. On the other hand, in the cases where the ML models of the FPdM solution are used separately, as part of other Federated Learning frameworks, the results are expected to be visualized through the selected components/federated platforms.

2.4.4 Information Flow

The information flow within Atlantis' component varies, depending on the use case. For the use case(s) where the whole FPdM structure will be utilized, the data flow can be found in the upper part of Figure 4, referring to Use Case 1.

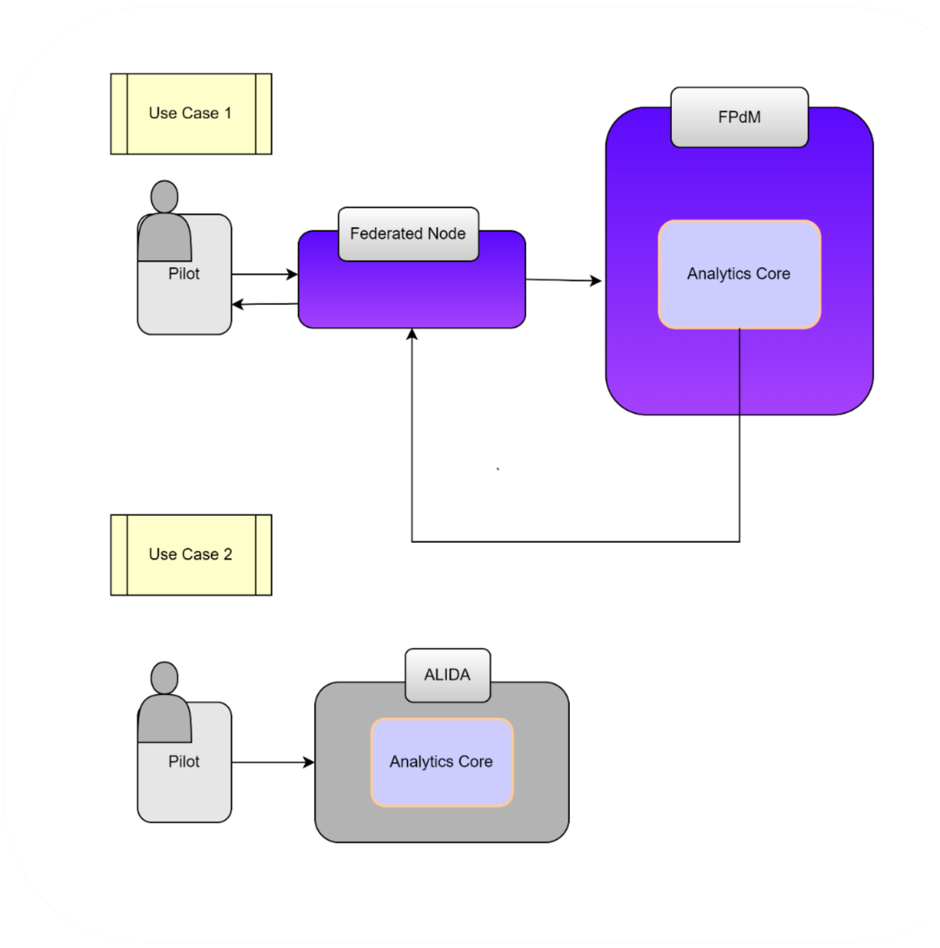


Figure 4: Information flow of the FPdM component based on the different use cases (Use case 1 & Use case 2)

On the other hand, in case a different Federated Framework is used for federation purposes, the Atlantis contribution will consist of an Analytics Core. The term "Analytics Core" refers to a trained model, integrated in different frameworks. This second case is depicted in the lower part of Figure 4: Information flow of the FPdM component based on the different use cases (Use case 1 & Use case 2). In Use Case 2, ALIDA will be used for the external Federated Framework.

2.4.5 Internal Architecture

Figure 5 provides an overview of the internal architecture of the custom Federated Learning Framework and the respective platform solution (FPdM). This architecture is expected to be updated during the life of the project.

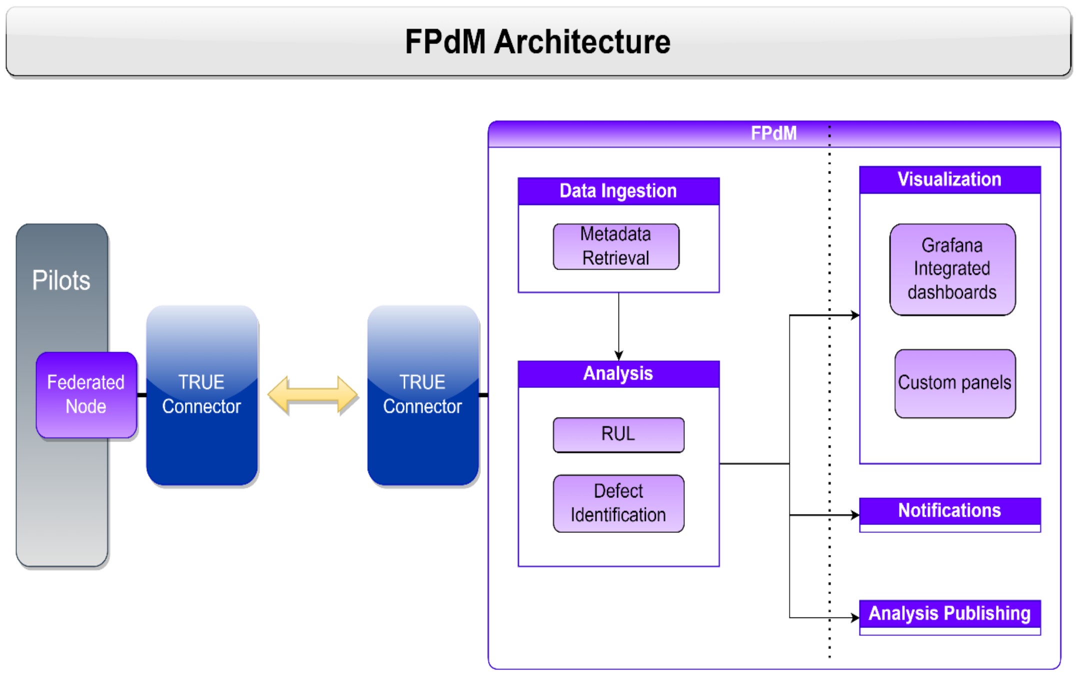
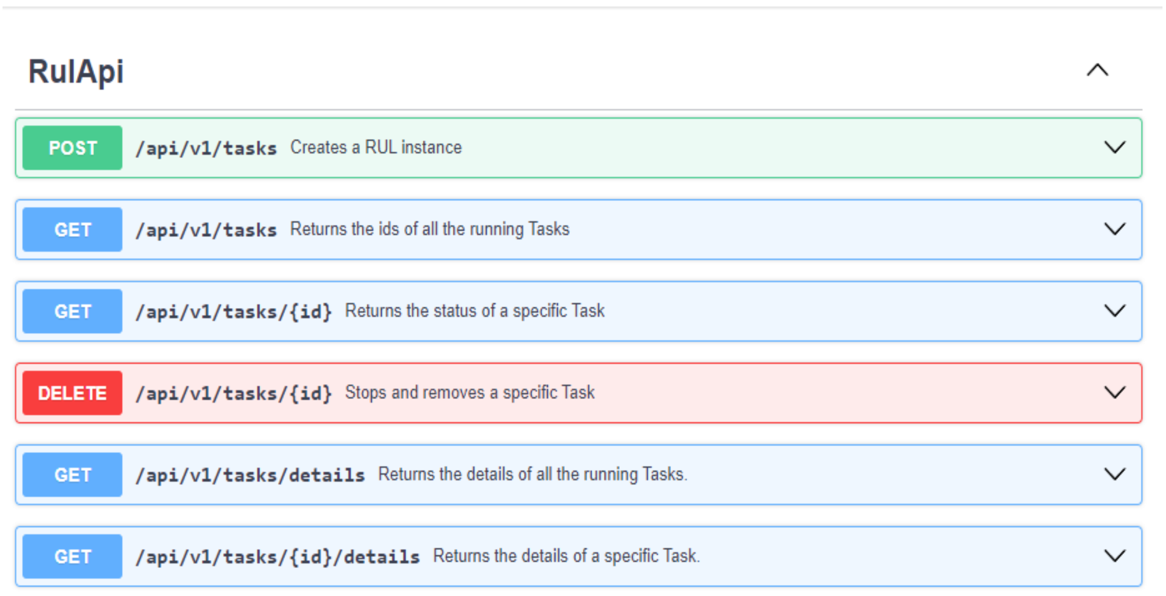


Figure 5: Internal Architecture of the complete FPdM solution

2.4.6 API

Interfacing with the platform will be achieved by using APIs. The APIs which will be made available are currently in the development and finalization phase. Thus, additional information and an in-depth description will be provided on the upcoming revisions and updates.

The basic endpoints available within an Analytics Core service are presented in Figure 6:


RulApi

POST /api/v1/tasks Creates a RUL instance

GET /api/v1/tasks Returns the ids of all the running Tasks

GET /api/v1/tasks/{id} Returns the status of a specific Task

DELETE /api/v1/tasks/{id} Stops and removes a specific Task

GET /api/v1/tasks/details Returns the details of all the running Tasks.

GET /api/v1/tasks/{id}/details Returns the details of a specific Task.

Figure 6: Available endpoints for the FPdM solution

2.4.7 Implementation Technology

Within the FPdM, a diverse array of sub-components encompassing communication and visualization services, machine learning models, and user interfaces can be found. Achieving this diversity required leveraging different programming languages and platforms. Specifically, Python was employed for developing machine learning models, harnessing various libraries like TensorFlow. The core framework was crafted using C#, and for the frontend, the Angular framework was used.

2.4.8 Comments

None

2.5 Component Name: CERTH Sovereign Data Transformation Service

2.5.1 Description

In the scope of the RE4DY project, the paramount importance of robust ETL (Extract, Transform, Load) services has necessitated the development of specialized Data Transformation Services. Central to these services is the use of Apache NiFi[1], a recognized

[1] https://nifi.apache.org/

leader in the realm of data flow technologies, which will underpin the foundational architecture of our ETL processes.

Apache NiFi's versatile capabilities allow our Data Transformation Services to accommodate a broad spectrum of file formats, including but not limited to JSON, CSV, and plain text. This adaptability ensures that our services can seamlessly integrate into diverse data ecosystems, especially in scenarios where file format restrictions are minimal or non-existent. Moreover, in instances where the incoming file format might not align with preferred standards, our services are still equipped to carry out the necessary transformations efficiently.

Leveraging the power of processors within Apache NiFi, we aim to holistically address the three core pillars of ETL – extracting data from its source, transforming it into the desired format, and loading it to the target system or database. This approach ensures that our Data Transformation Services are not only robust and reliable but also highly adaptable to the unique requirements presented by different use-case providers in the RE4DY project.

To further ensure sovereign and secure data communication the RE4DY data transformation services across the digital fabric are coupled with data space connectors. The used connectors are based on IDSA RAM [2] and provide all the necessary capabilities for setting rules regarding data access and sharing.

2.5.2 Input

Main Inputs for the Data Transformation Component are:

- Source Data: This encompasses a variety of file formats that the Data Transformation Component is designed to handle, including but not limited to JSON, CSV, plain text, etc.
- Configuration Parameters: These might include settings related to data extraction, transformation logic, or loading processes. They could be specific parameters for Apache NiFi processors, settings for JOLT[3] transformations, or custom parameters for other tools and processes within the component.
- Transformation Rules: These define how the source data should be transformed. Given that JOLT technology is being utilized, this could refer to specific JOLT transformation specifications.

[2] https://internationaldataspaces.org/publications/ids-ram/
[3] https://github.com/bazaarvoice/jolt

- Ontological Mappings: If the Data Transformation Component interfaces with the Ontology Repository for data transformation, then mappings or references to standardized data models and ontologies are essential inputs.
- Target System or Database Specifications: Information on where the transformed data needs to be loaded, whether it is a type of database, a cloud storage location, or another system altogether.
- Provenance Data: Historical data detailing the lifecycle of each FlowFile, which can be used for data lineage, auditing, or debugging purposes. This data can come from the Provenance Repository.
- Custom Processor Settings: If there are any customized Apache NiFi processors developed specifically for the RE4DY project, their configurations, scripts, or any other related settings would be vital inputs.

2.5.3 Output

Main Outputs for the Data Transformation Component are:

- Transformed Data: Post-processed data ready for ingestion into target systems. This data would be in the desired format (JSON, CSV, plain text, etc.) as required by downstream applications or storage solutions.
- Data Load Logs: Reports or logs that provide a summary of the data load processes, detailing successes, failures, or any discrepancies encountered during the data transformation.
- Provenance Records: Enhanced records that indicate how each Flow File was processed, transformed, and loaded. These records, stored in the Provenance Repository, help trace the journey of each piece of data through the transformation process.
- Transformation Audit Trails: Detailed logs that keep track of all transformation rules applied, any data mapping done using ontologies, and any exceptions or errors encountered.
- Error Reports: In cases where data fails to transform or load correctly, error reports detailing the reason for failure, the data source, and any other relevant metadata.

- Ontological Data Mapping Reports: If data is mapped using ontologies from the Ontology Repository, a report detailing these mappings, the ontologies used, and the resultant structure of the transformed data.
- Performance Metrics: Metrics detailing the performance of the Data Transformation Component, including processing times, throughput, efficiency, and other relevant KPIs.
- Data Quality Reports: Post-transformation, these reports can provide insights into the quality of the transformed data, detailing any inconsistencies, missing values, or other potential issues.

2.5.4 Information Flow

Data Mapping Using Ontology

The use case "Data Mapping Using Ontology" describes the process where a User interacts with the components of the "Data Transformation Services" to map raw data to a standardized ontology model. This process includes the provision of input data, selection of the desired ontology model from the Ontology Repository, data mapping based on the selected ontology, and retrieval of mapped data by the User. The outcome is structured data in alignment with the chosen ontology, ensuring semantic coherence and compatibility across systems.

Primary Actor: User (Data Engineer, Data Scientist)

Secondary Actors: Ontology Repository, Data Transformation Subcomponent

Preconditions:

- The Data Transformation Services are properly configured and operational.
- The desired ontology model is available in the Ontology Repository.
- Raw data, along with any necessary parameters and configurations, are available.

Main flow:

- Input Data Provision - The User provides raw data and specifies the desired ontology model for mapping.
- Ontology Fetching - The Ontology Repository retrieves the specified ontology model.

- Data Mapping - The Data Transformation Subcomponent maps the raw data based on the retrieved ontology.
- Results Retrieval - The User accesses the mapped data and any associated metadata or visualizations.

Optional steps:

- Data Transformation Feedback - Users can provide feedback on the mapping results, suggesting improvements or corrections, which can be utilized for refining mapping rules or ontology models.

Exception paths:

- If a chosen ontology model is not available, the system returns an error notification to the User.
- If mapping errors occur, the system logs details and notifies the User.

Post-conditions:

- Mapped data, conforming to the chosen ontology, is generated and available to the User.
- Feedback from users might initiate refinements in mapping rules or ontology models.

Trigger:

The User initiates the data mapping process by providing raw data and specifying the desired ontology model.

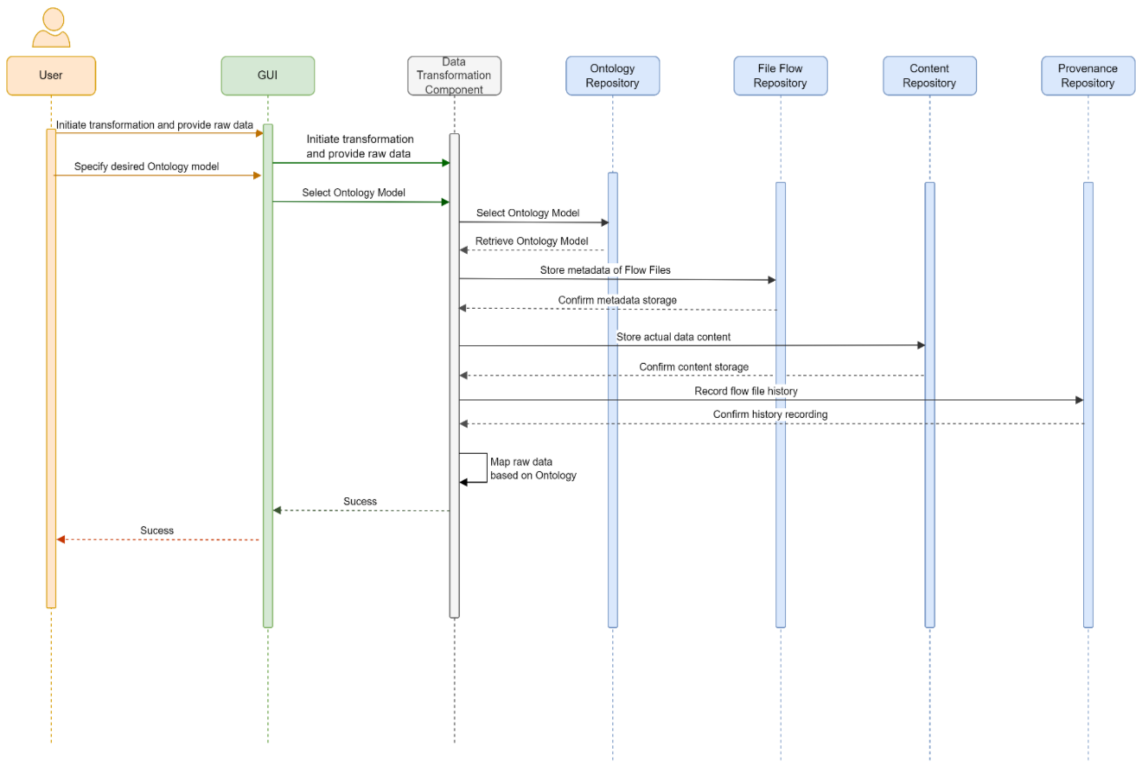

User
GUI
Data Transformation Component
Ontology Repository
File Flow Repository
Content Repository
Provenance Repository
Initiate transformation and provide raw data
Specify desired Ontology model
Initiate transformation and provide raw data
Select Ontology Model
Select Ontology Model
Retrieve Ontology Model
Store metadata of Flow Files
Confirm metadata storage
Store actual data content
Confirm content storage
Record flow file history
Confirm history recording
Map raw data based on Ontology
Sucess
Sucess

Figure 7: Data flow regarding ontological mapping of raw data

- Transforming Varied File Formats

The use case "Transforming Varied File Formats" delves into the process by which a User interacts with the components of the "Data Transformation Services" to transform diverse input file formats like JSON, CSV, and text into a desired standardized format. This transformation ensures data consistency and prepares data for further downstream processing or integration with different systems.

Primary Actor: User (Data Integration Specialist, Data Scientist)

Secondary Actors: Data Extraction Subcomponent, Data Transformation Subcomponent

Preconditions:

- The Data Transformation Services are properly configured and operational.
- Necessary configurations or transformation rules for the intended format conversion are defined and available.
- Raw data in its initial format is available for processing.

Main flow:

- File Submission - The User submits the data file in its initial format (e.g., JSON, CSV, text) and specifies the desired output format.
- Data Extraction - The Data Extraction Subcomponent extracts content from the provided file, preparing it for transformation.
- Data Transformation - The Data Transformation Subcomponent processes the extracted content, converting it into the specified format.
- Results Retrieval - The User accesses the transformed data file, which is now in the desired standardized format.

Optional steps:

- Transformation Feedback - Post transformation, Users can provide feedback on the results, potentially suggesting improvements, refinements, or corrections which can then be used to refine future transformations.

Exception paths:

- If the provided file format is unsupported or corrupt, the system returns an error notification to the User.
- If transformation errors occur due to incorrect configurations or unexpected content, the system logs details and notifies the User.

Post-conditions:

- Data is transformed into the desired format and is ready for further processing, storage, or integration.
- Feedback mechanisms may instigate refinements in transformation rules or methods.

Trigger:

The User initiates the data transformation process by submitting a data file and specifying the desired output format.

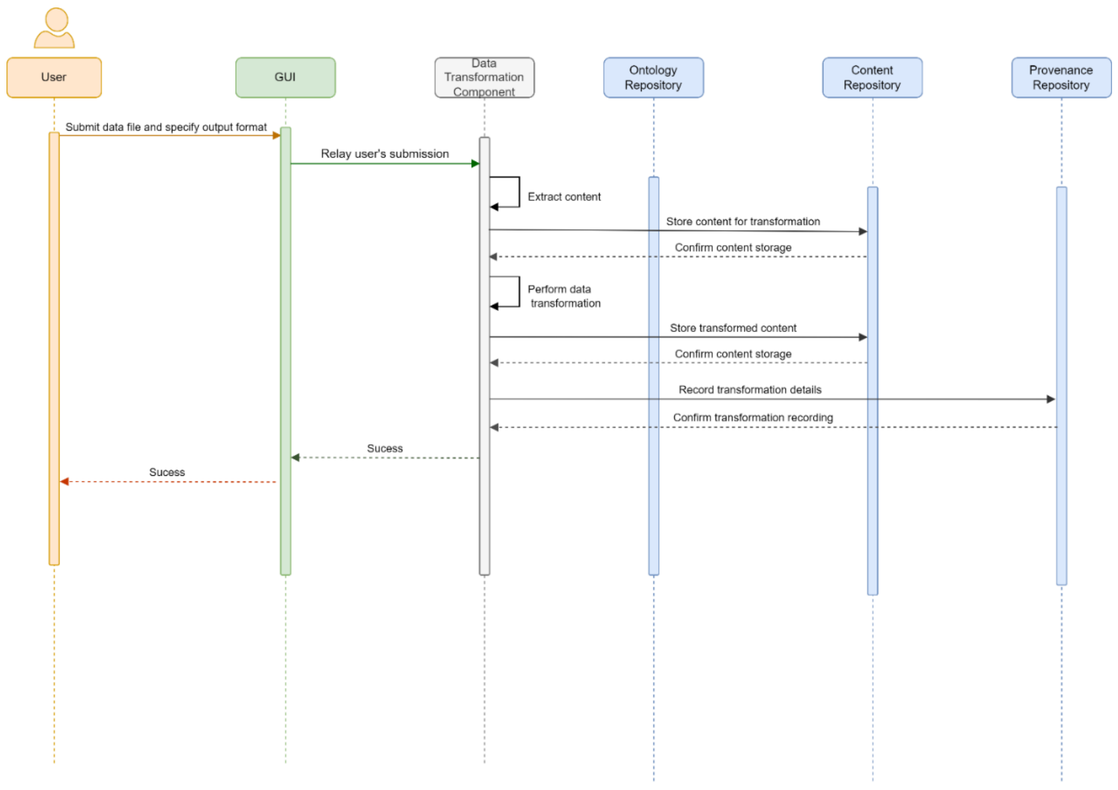

User
GUI
Data Transformation Component
Ontology Repository
Content Repository
Provenance Repository
Submit data file and specify output format
Relay user's submission
Extract content
Store content for transformation
Confirm content storage
Perform data transformation
Store transformed content
Confirm content storage
Record transformation details
Confirm transformation recording
Sucess
Sucess

Figure 8: Data flow regarding data transformation example

2.5.5 Internal Architecture

The architecture of the platform is structured into three distinct layers: the Presentation Layer, the Service Layer, and the Persistence Layer. The delineation into these three layers embodies a standard architectural design aimed at logically organizing the system's components. This segregation promotes a clear separation of concerns, paving the way for easier maintenance and scalability of the platform.

- The Presentation Layer is engineered for user interaction and information presentation.
- The Service Layer encapsulates the core business logic, acting as a conduit between the presentation and persistence layers, mediating their interactions.
- The Persistence Layer is devoted to data storage and retrieval, providing a robust foundation for the platform's data-centric operations.

Each layer, with its set of dedicated functionalities, interacts cohesively to deliver the comprehensive capabilities of the CERTH Data Transformation Component. This layered design also augments the platform's adaptability, ensuring that modifications or extensions in one layer have minimal ripple effects on the others.

Presentation Layer:

- For RE4DY end-user or other components that will consume data transformation services there are no UIs available. All the calls will be done through Data Space Connectors between data providers and data consumers.
- In general User Interface in Apache NiFi serves as the primary interaction point for users, enabling them to design, monitor, and manage their data transformations but this will be used during the development phase only. This UI is flexible, and efficient for diverse data integration needs and is going to substantially support the delivery of data transformation services.

Service Layer:

- Flow Controller: Is a core component of architecture. Essentially, it is the "brain" behind Data Transformation component operations, orchestrating the processing of data and ensuring that all NiFi and RE4DY custom processors work in harmony.
- Data Loading sub-component: Is a custom designed component for data loading, tailored to address unique RE4DY requirements. The component supports functions such as source integration, flexible data ingestion, and error handling.
- Data Extraction sub-component: Is a custom designed component for data cleaning and filtering (if needed).
- Data Transformation sub-component: Is a custom designed RE4DY data transformation processor, built on top of the Apache NiFi data transformation processors. Some of the envisioned functionalities so far are:
    - Data Conversion: Transform data from one format or structure to another. JOLT processors are used as well at this part.
    - Data Enrichment: Augmenting data with additional information.
    - Data Aggregation: Summarizing or grouping data.
    - Business Logic Application: Applying specific logic or rules.
    - Data Mapping: Mapping the data to the standardized data models and ontologies selected to be used by the RE4DY use case providers.
- Data Space Connectors[4]: They enable trusted and sovereign communication between two parties that are involved in a data transformation service. A data

[4] https://international-data-spaces-association.github.io/DataspaceConnector/

consumer that needs a data source with a specific data format will setup a data space connector. This connector will communicate with the relevant data space connector of the data provider. The data from provider data sources will be transformed through the Data Transformation Processors and the result will be transmitted to the consumer.

Persistence Layer:

- Flow File Repository: It is responsible for storing the metadata of the Flow Files that are currently being processed by the system. In essence, the Flow File Repository is crucial for ensuring data integrity, system recoverability, and providing a snapshot view of the current state of data.
- Content Repository: Is a core component of Apache NiFi that manages the actual content or data associated with the FlowFiles being processed in the system. Unlike the FlowFile Repository, which handles metadata, the Content Repository deals with the data payload.
- Provenance Repository: The Provenance Repository is responsible for recording and preserving a comprehensive history of each Flow File that flows through the system. This detailed log enables users to trace the lineage and lifecycle of the data as it is processed.
- Ontology Repository: Ontology Repository in the context of data transformation provides a structured, semantic framework that ensures that data is not just transformed in structure but also in meaning. By mapping data to standardized ontologies, it provides a way to ensure data consistency, integration, and meaningful representation across diverse RE4DY use cases.
- External Data Sources: The actual data sources from RE4DY partners that is going to be transformed.

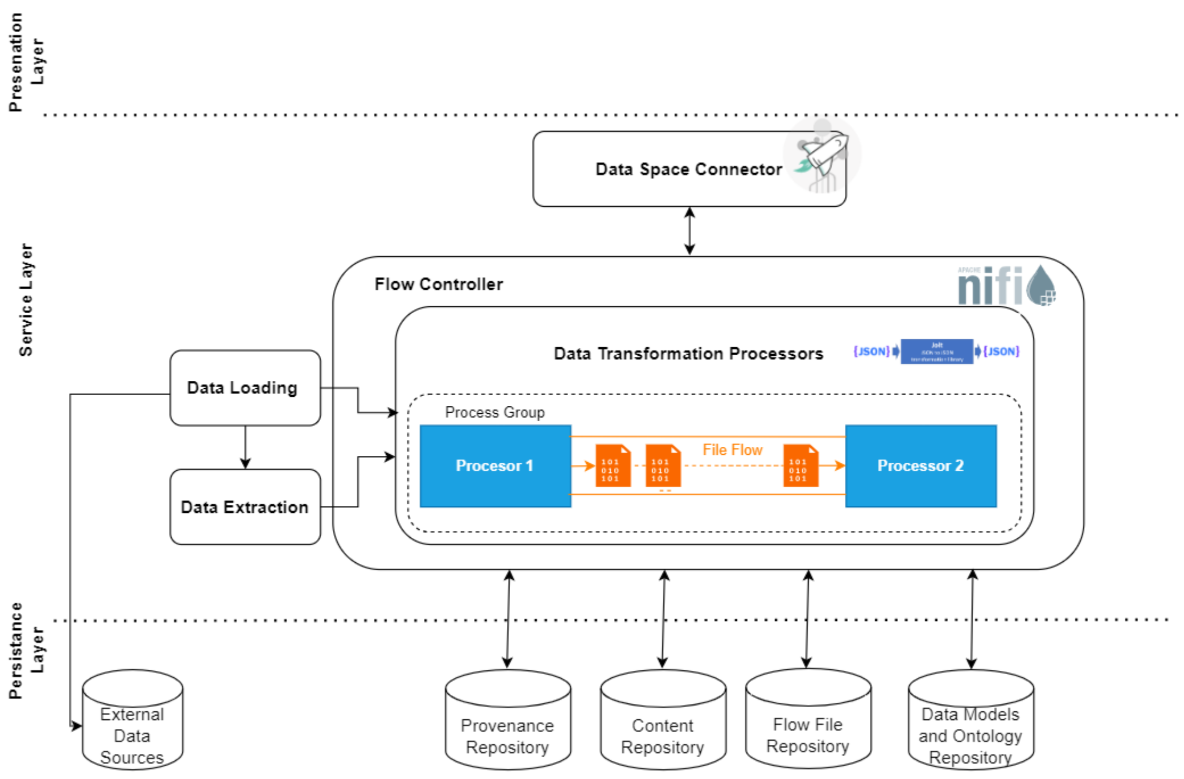


Figure 9: Architecture of RE4DY Data Transformation Services

2.5.6 API

To be defined in the updated version of the deliverable, namely D3.3

2.5.7 Implementation Technology

The combination of IDSA Connectors, Apache NiFi and JOLT for our custom data extraction component ensures that we have both a robust and scalable data extraction framework, as well as specialized capabilities for JSON data transformations. Apache NiFi offers us the infrastructure and environment to manage large-scale data flows, while JOLT provides the specificity and flexibility needed for JSON data manipulations. Together, they form a potent tech stack for our custom data extraction needs.

2.5.8 Comments

None

2.6 Component Name: CERTH XAI and Active Learning Platform for Defect Detection

2.6.1 Description

The platform offers AI-driven defect detection by analyzing images of manufacturing assets, specifically tailored for the hard metal industry. The defect detection and localization platform is enhanced with AI explainability and human-AI collaborative features. Designed using a micro-service architecture, the platform is adaptable and extensible, catering to a diverse set of users. This includes data scientists who develop AI models and maintainers who monitor conditions with these pre-trained models. At its heart, the platform employs advanced Machine Learning and Deep Learning techniques, ensuring high precision in both defect recognition and localization.

The XAI and Active Learning Core Engine is a central component of the XAI and Active Learning platform, ensuring that key functionalities related to reasoning are seamlessly integrated and executed. It consists of three components briefly described below:

1. Reasoning/Recommender Module:
    - Purpose: This module stands as the primary component responsible for offering the inference and reasoning services of the platform.
    - Technical Approach: To cater to a diverse set of use-cases and ensure robustness in its inference capabilities, this module incorporates a multitude of techniques. Notably, it employs rule-based and graph-based reasoning methodologies. Additionally, fuzzy logic is integrated, enhancing the flexibility and adaptability of the reasoning processes. This ensures that the module can cater to both well-defined and ambiguous data scenarios, thereby broadening its applicability.
2. Data Management Module:
    - Purpose: At its core, this module's responsibility is to orchestrate the collection and subsequent management of data.
    - Data Sources: Data ingested by this module can stem from a variety of origins. One significant source is the simulated data generated by the Digital Twin. This serves to provide a virtual representation of the data scenarios. Additionally, this module integrates historical data and real-time data garnered from other WP3 components, ensuring a holistic data perspective.

3. Model Management Module:

- Purpose: This module is dedicated to the meticulous management of AI models. Given the dynamic nature of artificial intelligence and the continuous evolution of models, a dedicated module for this purpose ensures that the most optimized and updated models are always in operation.

- Functionality: Beyond mere storage, this module facilitates version control, monitoring, and updates for AI models, ensuring that they remain relevant and accurate over time.

4. XAI Module:

- Purpose: This module is dedicated to interpreting AI results and decisions.
- Functionality: Visual representation (graphs etc.) regarding XAI is provided. For example, graphs explaining the importance of various features during training phase.

2.6.2 Input

Main inputs for the XAI and Active Learning Core Engine are:

- Manufacturing Asset Images that the system analyzes. Given the specificity for the hard metal industry, these images might reveal intricate details, patterns, and potential anomalies or defects.

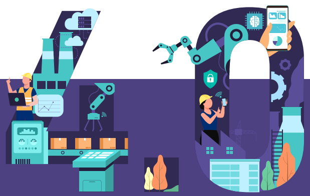
- Model Parameters and Configurations that guide how the image analysis and defect detection models should operate. For example, training configurations or specific thresholds for defect identification.

- Feedback Data for Active Learning collected during defects identification and validation (i.e., either confirmed or corrected defects). This feedback can be looped back into the system to refine and improve the AI models continually.

- Operational and Historical Data: Information about the manufacturing processes, historical defect rates, types of defects commonly encountered, and other related data can be valuable inputs. This data can provide context, aiding in the interpretation of results and in making informed recommendations.

2.6.3 Output

Main outputs of the XAI and Active Learning Core Engine are:



- Defect Detection Reports/Graphs about the type, size, location, and severity of each defect identified in the analyzed images.
- Explanations and Justification: For each defect detected, the system offers a human-readable explanation using Explainable AI (XAI) techniques, supplemented by visual aids where appropriate.
- Recommendations: Based on the identified defects, the engine generates actionable insights or recommendations, suggesting maintenance, repair, or additional inspections, and may provide predictive insights about potential future issues.
- Active Learning Feedback Loops: Leveraging active learning, the engine flags areas of uncertainty in its analyses and seeks human verification, while also reflecting improvements made from previous feedback.
- Data Summaries: Users are presented with summaries that provide overviews of the collected data, displaying trends in defect types and frequencies, as well as other relevant data usage statistics.
- Model Management Summaries: An overview of all AI models deployed, detailing their versions, update histories, and training data.
- Alerts and Notifications: Users are kept informed through real-time alerts and notifications about critical defects, anomalies, system health, and data collection status.

2.6.4 Information Flow

AI-Driven Defect Detection

The use case "AI-Driven Defect Detection" outlines the process through which a User (Data Scientist, Maintainer) interacts with the components of the “XAI and Active Learning Platform” (AI System) to analyze manufacturing asset images for potential defects. The procedure encompasses the provision of necessary input data (External Sources), processing of images using pre-trained AI models, generation of detailed defect detection reports, and the retrieval of these results by the User. Optionally, the User can contribute feedback for active learning to continually refine the AI models, enhancing the accuracy of defect detection over time. Through this structured interaction, the User can initiate a defect detection process, review the findings, and optionally participate in a feedback loop to improve future defect detection accuracy, thereby leveraging AI capabilities to maintain high standards of quality control in manufacturing assets.

Primary Actor: User (Data Scientist, Maintainer)

Secondary Actors: External Data Sources, AI System

Preconditions:

- The XAI and Active Learning Platform (AI System) is properly configured and operational.
- Relevant AI models for defect detection are available and properly trained.
- Manufacturing asset images along with any necessary model parameters and configurations are accessible.

Main flow:

- Input Data Provision - The User or External Data Sources provide manufacturing asset images, model parameters, and configurations to the AI System.
- Image Processing - The AI System processes the provided images using pre-trained machine learning and deep learning models to identify potential defects.
- Defect Detection Report Generation - Upon analyzing the images, the AI System generates defect detection reports. These reports detail the type, size, location, and severity of each detected defect.
- Results Retrieval - The User accesses the defect detection results, reports, and any associated data visualizations through the Graphical User Interface (GUI) or the REST API.

Optional steps:

- Active Learning Feedback - The User can provide feedback data for active learning, which is utilized to refine and improve the AI models continually, enhancing the accuracy of defect detection over time.

Exception paths:

- The User accesses the defect detection results, reports, and any associated data visualizations through the GUI (or a REST API).

Post-conditions:

- Defect detection reports are generated and accessible to the User.

- Active learning feedback loop may be initiated to enhance future defect detection accuracy.

Trigger:

The User initiates the defect detection process by providing the necessary input data or by instructing the system to utilize the data from External Data Sources.

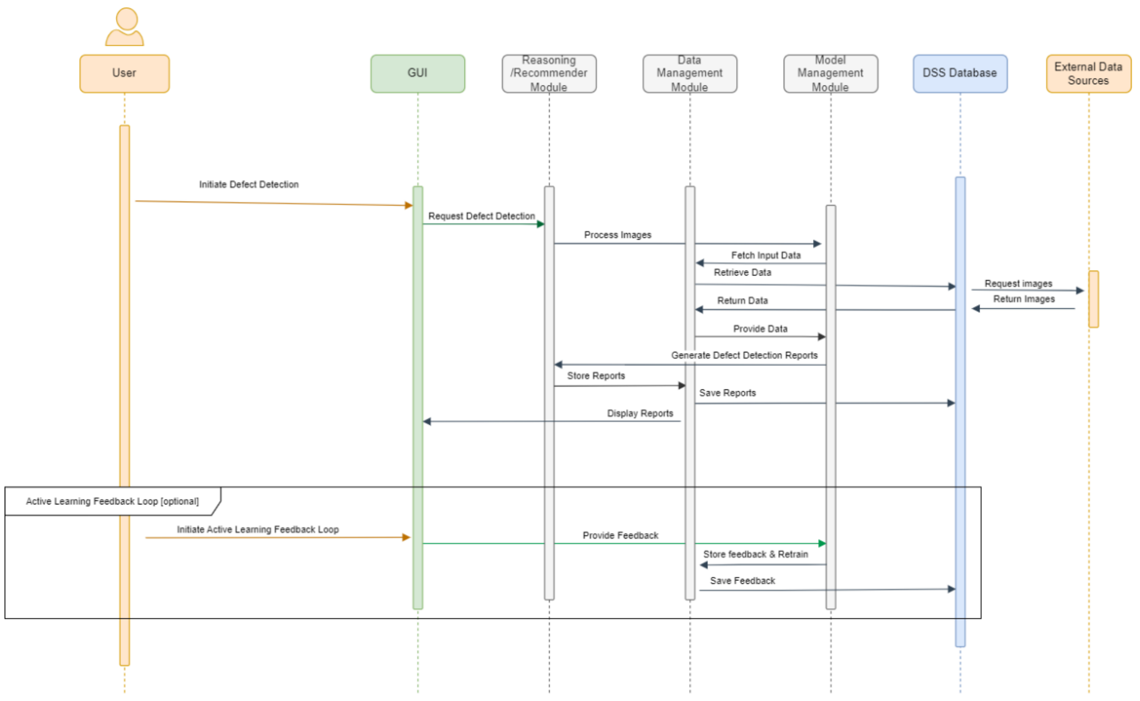


Figure 10: Data Flow of AI Driven Defect Detection

AI Explanation Retrieval

The "AI Explanation Request" use case facilitates a deeper understanding for the User (Data Scientist, Maintainer) by providing insights into the AI decisions and results, particularly pertaining to the identified defects in analyzed images, through the XAI Module of the system.

Primary Actor: User (Data Scientist, Maintainer)

Secondary Actors: AI System

Preconditions:

- AI System is properly configured and operational.
- Relevant AI models for defect explanation are available and properly trained.
- Images have been analyzed, and defects have already been identified by the AI System.

- The User has access to the system through the GUI or the REST API.

Main Flow:

- The User initiates a request for an explanation of the AI decisions and results through the GUI (or a REST API).
- The AI System processes the request through the XAI Module to generate explanations and visual representations (e.g., feature importance graphs) regarding the AI decisions.
- The explanations and visual representations are made accessible to the User through the GUI (or a REST API).

Post-conditions:

The User gains insights into the AI decisions through the explanations and visual representations provided by the XAI Module.

Trigger:

The User initiates the process by requesting an AI explanation.

Exception Paths:

In the event of any anomalies or issues during the AI explanation generation, an error notification is generated by the AI System and relayed to the User or System Administrator for resolution.

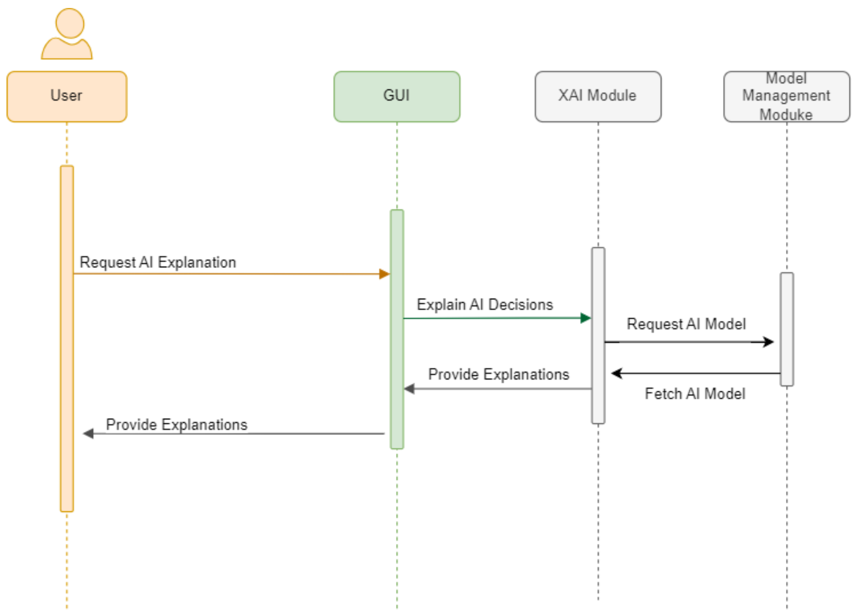


Figure 11: Data Flow of XAI

AI Model Management

The use case "Model Management" delineates the interaction between a User (Data Scientist, Maintainer) and the AI System for effectively managing the available AI models dedicated to defect detection. Through the Model Management Module, accessed via the Graphical User Interface the User engages in various model management tasks such as version control, monitoring, and updating AI models based on evolving requirements, feedback, or new data. These actions are facilitated by the AI System to ensure proper storage, tracking, and updating of the models as per the User's actions, enabling a structured approach to managing the AI models to ensure their optimum performance and relevancy over time. Through this defined flow, any encountered anomalies or issues trigger error notifications, ensuring the User or System Administrator is informed for prompt resolution, contributing to the effective management and utilization of AI models for defect detection.

Primary Actor: User (Data Scientist, Maintainer, System Administrator)

Secondary Actor: AI System

Preconditions:

- The AI System is properly configured and operational.
- Relevant AI models for defect detection are available and properly trained.
- The User has access to the system through the Graphical User Interface (GUI).

Main Flow:

- The User accesses the Model Management Module through the GUI to view and manage the available AI models.
- The User can perform various model management tasks including:
  - Version control to track and manage different versions of AI models.
  - Monitoring to oversee the performance and usage of AI models.
  - Updating AI models based on evolving requirements, feedback, or new data.
- The AI System facilitates these model management tasks, ensuring the models are properly stored, tracked, and updated as per the User's actions.

Post-conditions:

AI models are managed effectively through the Model Management Module, with their versions controlled, performance monitored, and updates properly executed.

Trigger:

The User initiates the process by accessing the Model Management Module to manage the AI models.

Exception Paths:

In the event of any anomalies or issues during the model management phases, an error notification is generated by the AI System and relayed to the User or System Administrator for resolution.

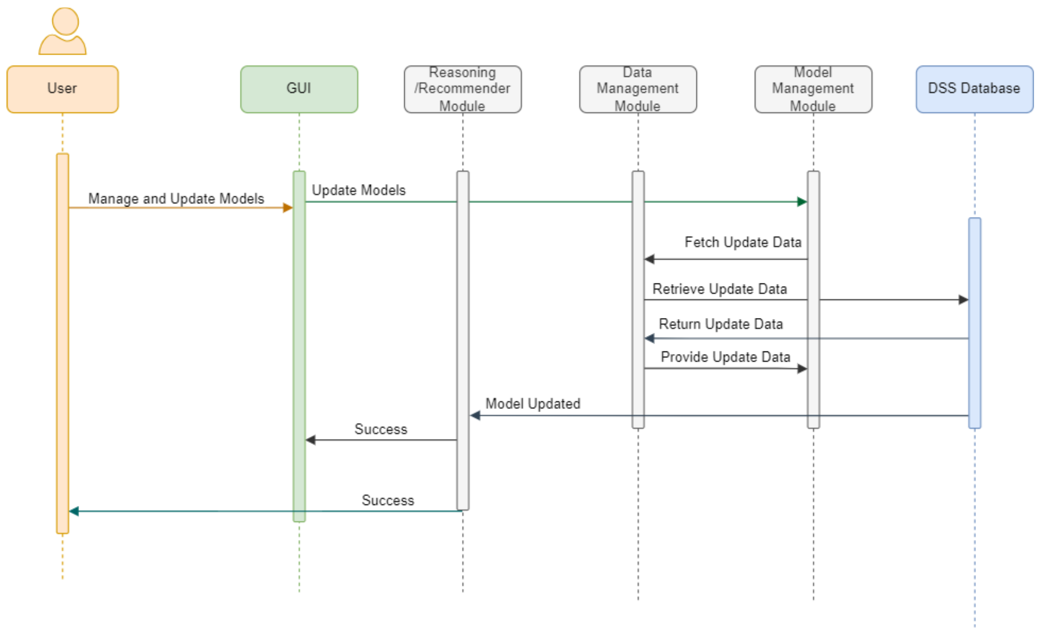


Figure 12: Data Flow for AI Models Management

2.6.5 Internal Architecture

The architecture of the platform is structured into three distinct layers: the Presentation Layer, the Service Layer, and the Persistence Layer. The delineation into these three layers embodies a standard architectural design aimed at logically organizing the system's components. This segregation promotes a clear separation of concerns, paving the way for easier maintenance and scalability of the platform.

- The Presentation Layer is engineered for user interaction and information presentation.

- The Service Layer encapsulates the core business logic, acting as a conduit between the presentation and persistence layers, mediating their interactions.
- The Persistence Layer is devoted to data storage and retrieval, providing a robust foundation for the platform's data-centric operations.

Each layer, with its set of dedicated functionalities, interacts cohesively to deliver the comprehensive capabilities of the CERTH XAI and Active Learning Platform for Defect Detection. This layered design also augments the platform's adaptability, ensuring that modifications or extensions in one layer have minimal ripple effects on the others. This is particularly advantageous in the microservices architecture employed by the platform, facilitating modular development and continuous enhancements.

Presentation Layer:

- REST API: Facilitates structured communication and data exchange between the client and server-side, generally employing JSON for data formatting.
- GUI: Provides a user-friendly visual interface, enabling users to interact with the system and navigate through its functionalities effortlessly.

Service Layer:

- Reasoning/Recommender Module: Administers the inference, recommendation, and reasoning functionalities of the platform.
- Data Management Module: Orchestrates the collection, management, and integration of data from various sources, ensuring a well-rounded data perspective for the platform's analysis and decision-making processes.
- Model Management Module: Dedicates itself to the meticulous management of AI models, ensuring they are always optimized, updated, and operationally relevant.
- XAI Module: Concentrates on interpreting the AI's decisions and results, delivering visual representations and explanations that aid users in comprehending the AI's operations and outcomes.

Persistence Layer:

- DSS Database: Serves as the data storage hub where all pertinent data for the platform is securely stored and managed, ensuring data persistence and consistent availability over time.

- Historical Database: Houses the external data sources, preserving historical data, which can be leveraged for trend analysis, predictive maintenance, and other data-driven insights.

- Digital Twin Database (optional): Accommodates Simulation Data, providing a virtual representation of the manufacturing assets and processes. This database is crucial for testing, simulation, and analysis, enabling a better understanding and troubleshooting of real-world manufacturing scenarios.

The external data housed in the Historical Database and Digital Twin Database are crucial for the holistic data analysis and decision-making processes carried out by the platform. By segregating the Simulation Data and Historical Data, the architecture ensures organized, efficient, and accurate data management and retrieval, which in turn, supports the platform's AI-driven defect detection and reasoning capabilities. This addition emphasizes the storage of external data in dedicated databases within the Persistence Layer, explaining the significance of such segregation for the platform's operations.

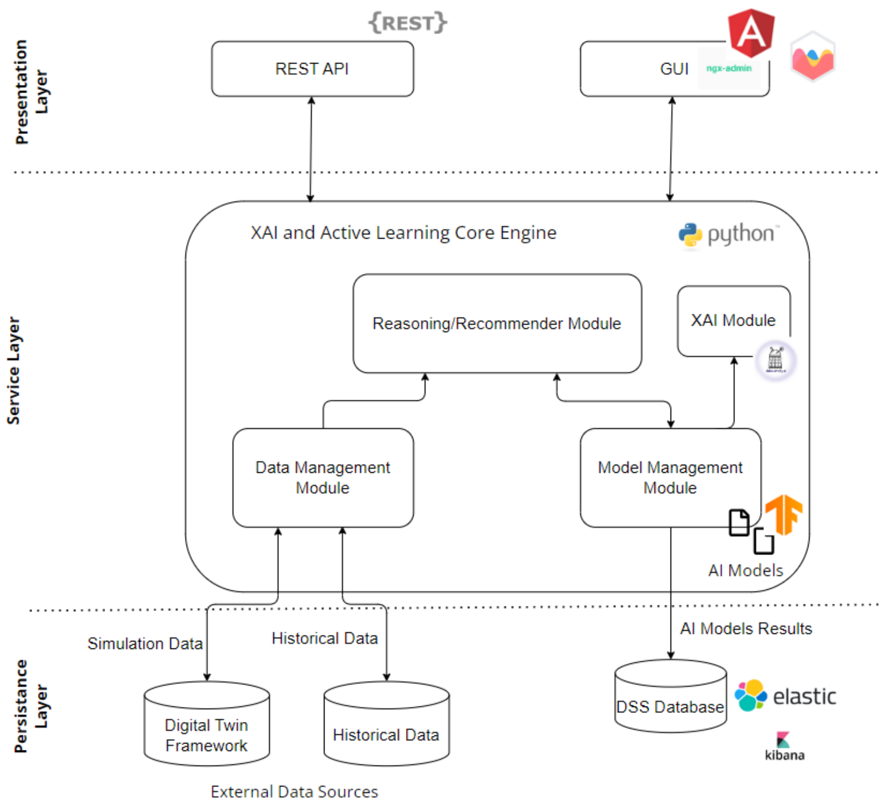


Figure 13: Architecture of RE4DY XAI and Active Learning Platform for Defect Detection

2.6.6 API

To be added in the last version of this deliverable, namely D3.3.

2.6.7 Implementation Technology

The component was developed using Python. TensorFlow and Keras libraries were used alongside architectures like ResNet, Deep CNN, XG-Boost, Random Forest, Yolov3, Yolov5 regarding Machine Learning. It supports HTTP communication and JSON format for input/output. Regarding XAI, DALEX package is used. The front-end is developed on Angular TypeScript and ngx-admin framework. Libraries such as Chart.js were used.

Some of the implementation decisions:

- Deployment: The platform will be offered as a Docker Container. The only prerequisite for deployment is having Docker installed, facilitating a smooth experience to run the containerized image. The platform for inference capabilities can be deployed even in an edge device. The full platform with training/re-training needs a machine with at least 6GB GPU, 32GB RAM.
- Data Storage and Management: To address the storage needs, an Elastic Search Instance will be set up. It will be deployed locally within a Docker container. The choice of Elastic Search will be driven by its capabilities to efficiently store, search, and analyze vast data volumes in near real-time. In tandem, a Kibana Instance will be employed to visualize the content of Elastic Search indexes, offering a graphical perspective.

2.6.8 Comments

None

2.7 Component Name: Decentralized data management & analytics

2.7.1 Description

The component will provide a set of completely decentralized ML algorithms supporting extraction from local data, targeting specifically unsupervised ML cases. It innovates with respect to conventional Federated Machine Learning as nodes will be fully responsible to self-organise their federation(s) in order to exchange partial models and train them in a collaborative fashion.

This component will allow optimized management of local data, which will not be shared among locations, still extracting knowledge out of them through the collaboration between the involved nodes.

Specifically, the component will be specialized at least for the task of unsupervised clustering in order to identify classes of quality products.

2.7.2 Input

The component requires a series of datasets describing certain physical phenomena, where each data point must be described according to a feature vector defined by the domain owner. Ideally, classification of data points according to feature similarities should allow the data owner to clearly identify classes relevant to the required task (e.g., presence of classes of low quality parts in the case of production environments).

2.7.3 Output

The component provides a set of classes identified, on a node-by-node basis.

2.7.4 Information Flow

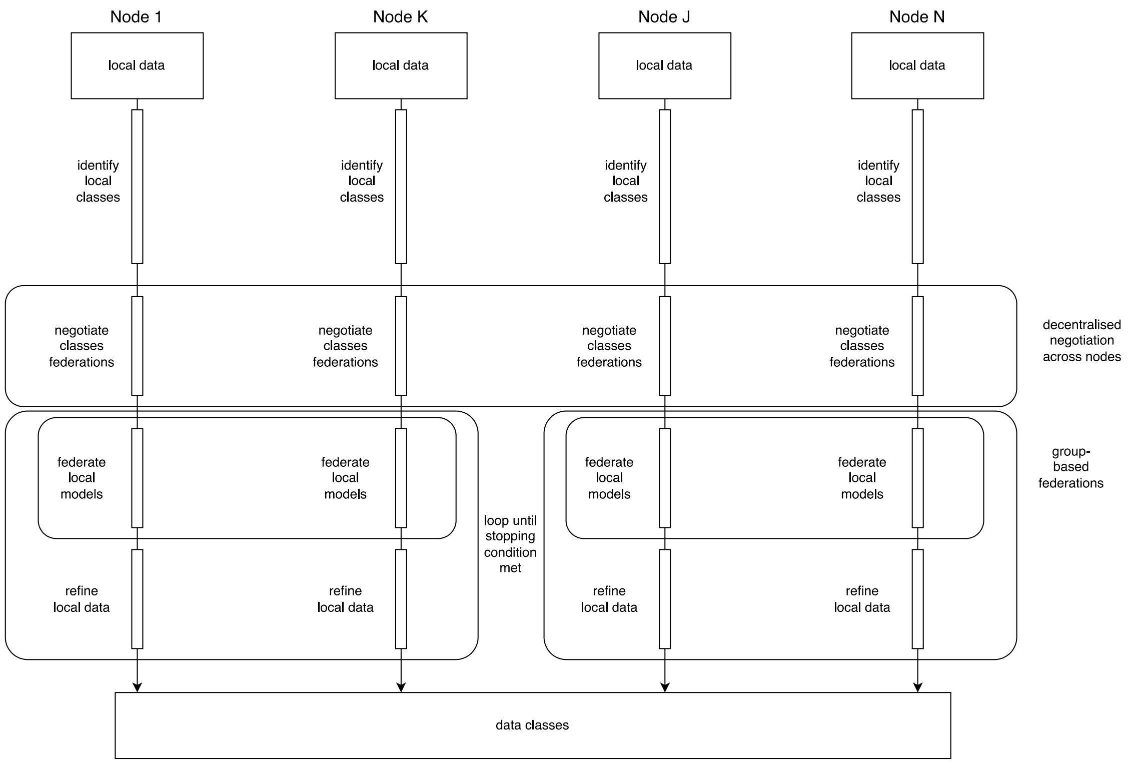


Figure 14: Information Flow

2.7.5 Internal Architecture

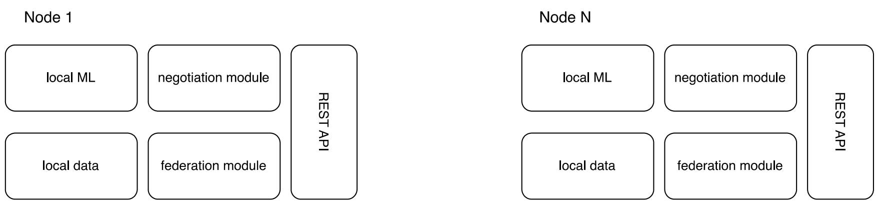


Figure 15: Node-by-Node Internal Architecture

2.7.6 API

The component is basically a set of algorithms packed in a library. The specific APIs will be specified based on the requirements of the pilot where the algorithms will be used.

In general, similar to other complementary components developed in the project (such as the “vanilla” Federated Learning libraries), the algorithms exploit the Flower open-source toolkit for the internals of decentralized learning.

2.7.7 Implementation Technology

Python using conventional ML libraries, depending on the specific task the component is applied to.

2.7.8 Comments

None

2.8 Component Name: FEDMA - Federated Maintenance for Milling Machines

2.8.1 Description

This component utilizes a federated learning approach, which eliminates the need to centralize and process raw data from individual machines on an extremal server for training an ML model. Instead, multiple edge clients are equipped with isolated copies of the same ML model and perform dynamic retrains based on a pre-specified number of rounds at the factory site level. Clients distribute only model weights, and an aggregation event takes place at the server side during each training round, achieving increased security, abstraction, and a collaborative learning scheme.

CORE's methodology focuses on processing and analyzing data from multiple milling machines, leveraging federated learning to execute predictive maintenance tasks (e.g. estimating the remaining useful life, anomaly detection). This approach allows the final model to learn from data contributed by multiple machines, potentially enhancing the overall predictive capabilities and enabling more accurate maintenance strategies for minimizing downtime and optimizing milling operations.

2.8.2 Input

During each round of the model training stage, batches of operational data (provided by GF & FRAISA) from machines (e.g., vibration, torque, temperature, etc.) that execute their own milling process, will be given as input to the local models which have been deployed on the edge. Then, after local model training has been completed for each round, only the ML model parameters will be transmitted to the server's side for the aggregation step. This process will be repeated for a prespecified number of rounds every time the training phase has been scheduled to take place. After the aggregation step takes place on the server side the updated global model weights return to each edge device as an input for the new training session.

2.8.3 Output

After each local training cycle, the output comprises the essential Machine Learning model weights, which are transmitted to the server for the aggregation phase. Those weights, along with model performance metrics will be saved in a database located on the server and will be analyzed to identify patterns, trends, or anomalies in the training process. This analysis can help in understanding how the model is learning from different edge machines and if there are any specific insights to be gained regarding the federated learning process. Visualization tools can help make these patterns more understandable to stakeholders. After training and evaluation, the aggregated model will be deployed and made available for use. This model, which has learned from the data of multiple edge machines, can be used to facilitate informed decision-making by stakeholders in predictive maintenance tasks for each edge machine.

2.8.4 Information Flow

1. Edge node collects operational data from the milling machine.
2. The data is pre-processed and then used as input for training the FL model.
3. Each node is trained locally on its collected data using its current model weights (which were sent originally after the global model was initialized on the central server).

4. All edge nodes send their respective model weight updates to the central server.

5. The server receives model updates from the selected client nodes and then aggregates these updates. This aggregation typically involves calculating the weighted average of the updates.

6. Once the global update has been computed the central server sends the updated global model weights back to each node.

7. Each edge node updates its local model with the new global weights received from the server.

8. The process above is repeated for a specified number of rounds.

9. After the training stage has been completed, model inference takes place until a trigger event initiates again the federated training process described above (e.g., data drift, new data available, etc.)

10. Model inference results are stored in a database and visualized.

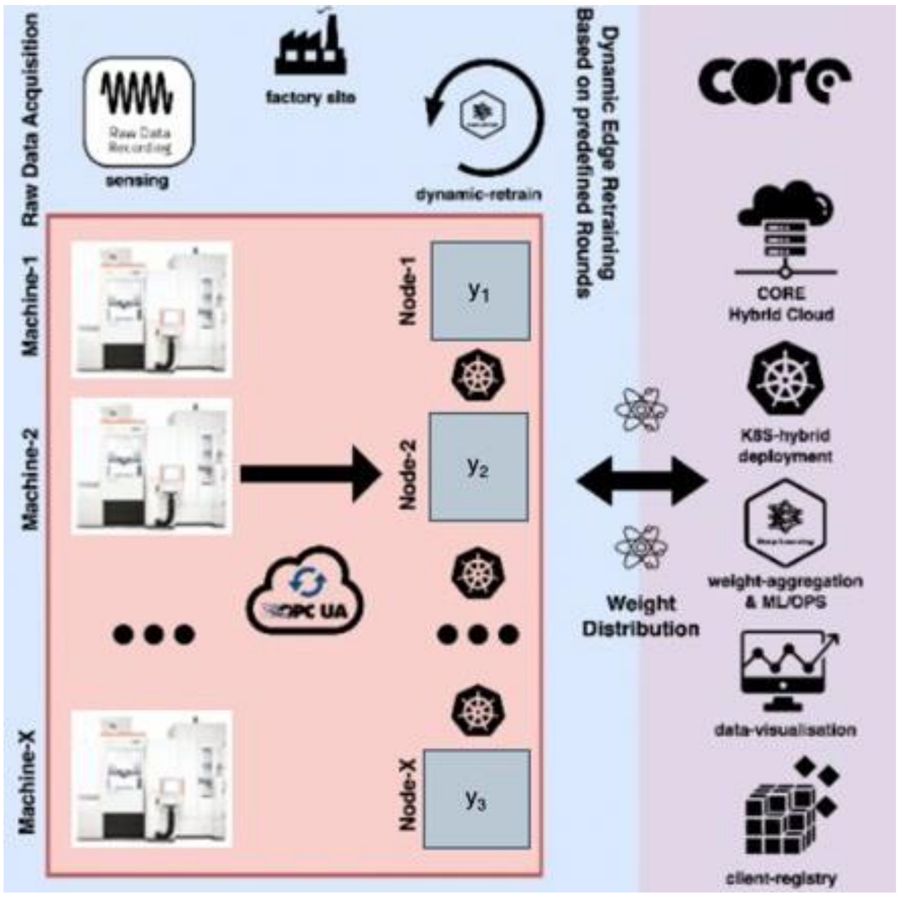


Figure 16: Federated Learning System Architecture

2.8.5 Internal Architecture

- Preprocessing Module: Cleans and structures data for input.
- Local Node Module: Processes data and computes local model updates.
- Server Module: Applies the aggregation step when receiving updates on local model parameters. This module is also responsible for sending the incoming inference result to the Storage Module.
- Communication Module: Sends model parameters to/from the central server from/to the edge nodes. It also sends the inference to the central server.
- Inference Module: Infers trained model using incoming data.
- Storage Module: Stores model inference output to a database.

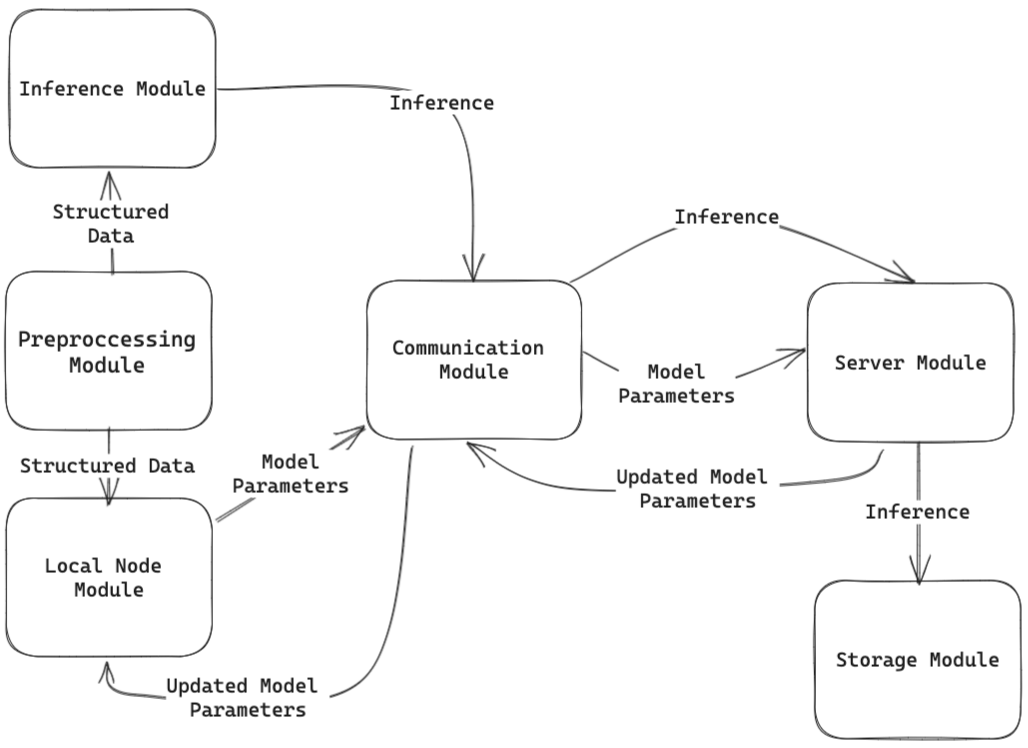


Figure 17: Interlinking of Internal Components: Communication Blueprint

2.8.6 API

- `preprocessData(rawData)`: Returns pre-processed data.

- `trainModel(data)`: Computes local model updates.
- `sendUpdates()`: Sends model updates to the central server.
- `receiveUpdates()`: Retrieves updated model weights from the server.
- `sendInference()`: Send output model inference.
- `saveInference()`: Store inference in database.

A more comprehensive API definition will be provided in the next stages of this component's development.

2.8.7 Implementation Technology

The selected programming language is Python, known for its readability and widespread use in data science and machine learning. The project also employs popular open-source libraries. TensorFlow or PyTorch are both powerful platforms for developing sophisticated Machine Learning models. The Flower framework plays a critical role in orchestrating Federated Learning processes, enabling the decentralized training of models across numerous devices, thereby enhancing data privacy, and optimizing communication costs.

A pivotal component in the setup is the message broker service, which could be either MQTT or Kafka, facilitating seamless and real-time transmission of model inference results to the central server. Triton Inference Server is an open-source inference serving software that streamlines and optimizes AI inferencing while managing requests in real-time. All logging details and model inference results are carefully stored and managed using InfluxDB, a high-performance time-series database.

2.8.8 Comments

The Flower framework selection speeds up the implementation of the Federated Learning architecture. Collaboration with GF and FRAISA is essential to ensure data quality and model effectiveness.

2.9 Component Name: ALIDA

2.9.1 Description

ALIDA provides a micro-service-oriented platform for the deployment and execution of Big Data Analytics (BDA) application compositions in several domains and scenarios. Within RE4DY project this platform will be enriched with Federating Learning capabilities in order to assure a privacy-preserving federated learning environment with which to build data-driven AI models to be assessed by RE4DY trials.

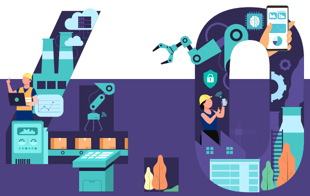
Therefore, this component enables users to create AI/ML pipelines by dragging and dropping reusable blocks from its own catalogue through a graphical visual designer. The pipelines can either run on the ALIDA cloud or be exported and executed locally. If the needed blocks are not available in the catalogue, the user can code them starting from the provided templates.

2.9.2 Input

- Graphical specification of pipelines: that users enter through the graphical designer integrated in the ALIDA platform.
- Datasets*:* that users can upload and use as starting blocks in their pipelines. After the upload, they become ("canvas draggable") ALIDA entities.
- Generic Local datasets/data sources (e.g., local filesystem folder or similar): rather than being ALIDA entities, these are completely external and independent data sources which can be accessed from the exported pipelines.
- Custom BDA Services: that users can develop by following the provided templates, upload and use whenever needed in order to assemble pipelines.
- External Streaming Data Sources: that can be accessed through the ALIDA Streaming Data Ingestion Services - a type of BDA Service - already available in the catalogue or that can be developed by the user.

2.9.3 Output

- Trained ML models: resulting from the interaction between pipelines of Participants (that train the model locally) and Aggregators (that aggregate the partial models returned by the Participants).
- Participant pipelines: that can be exported from ALIDA and executed externally on both edge and cloud sides.



2.9.4 Information Flow

Note: the following diagrams are only examples.

Aggregator and Participants Pipelines Preparation

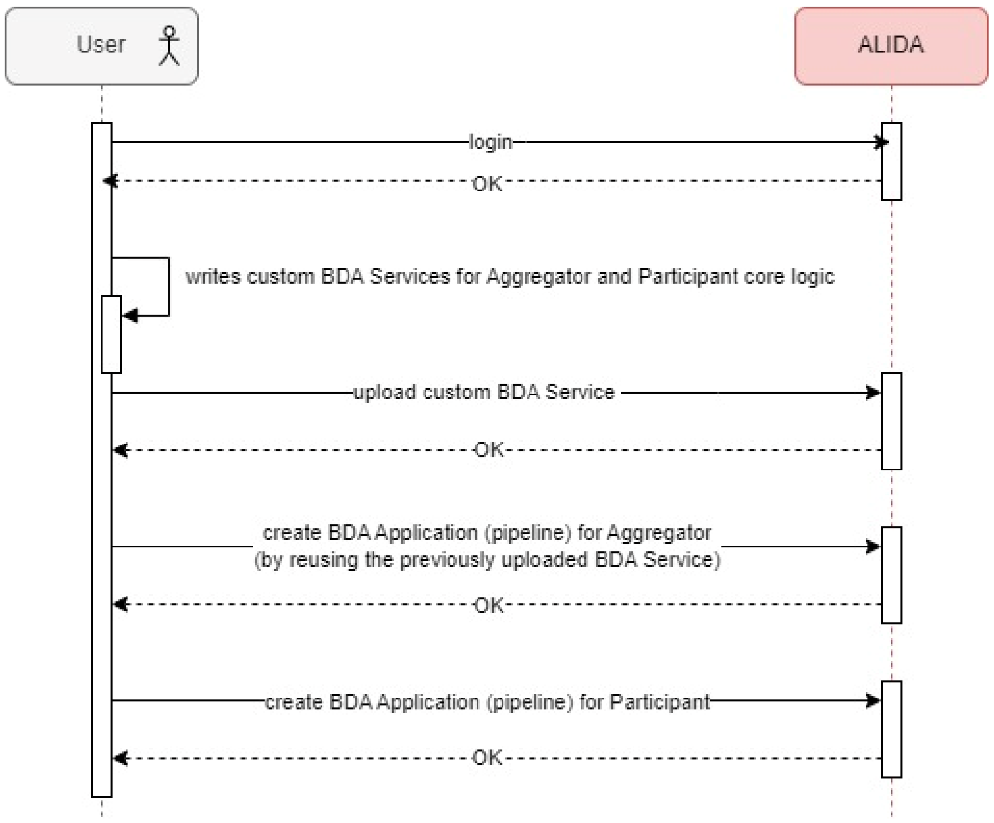

User
ALIDA
login
OK
writes custom BDA Services for Aggregator and Participant core logic
upload custom BDA Service
OK
create BDA Application (pipeline) for Aggregator
(by reusing the previously uploaded BDA Service)
OK
create BDA Application (pipeline) for Participant
OK

Figure 18: ALIDA FML application design steps example

As shown by the sequence diagram above, the design of an FML application with ALIDA platform is summarized through the following main steps.

1. The user logs in to ALIDA platform.
2. If not already available from catalogue, the user defines and develops the BDA services accordingly to his/her needs for the Aggregator and Participants roles.
3. The user uploads custom BDA services to ALIDA.
4. At this point the user can design the pipeline for the Aggregator BDA application by simply dragging & dropping the BDA services needed, including those previously developed and uploaded.
5. Likewise, the user can design the pipeline for the Participant BDA application.

Federated Training

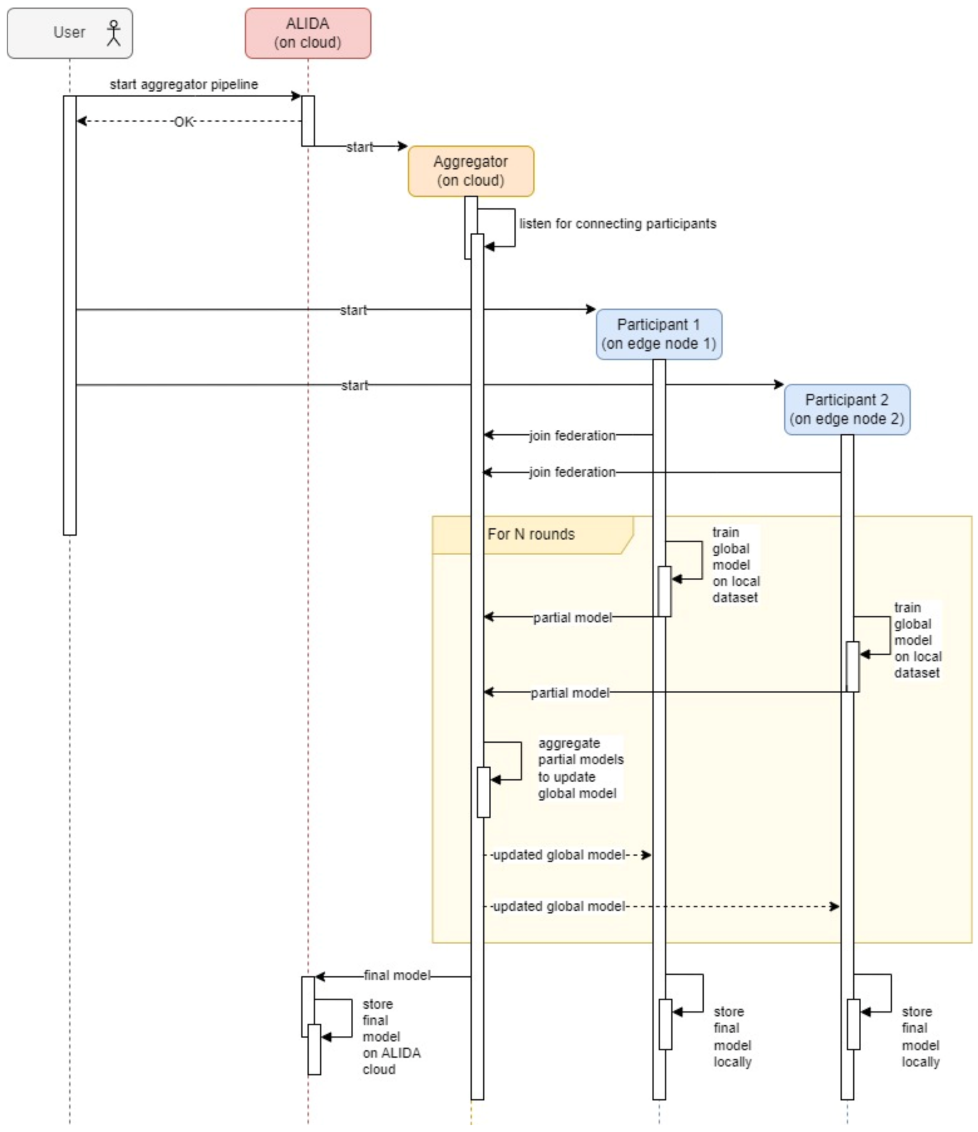

User
ALIDA (on cloud)
start aggregator pipeline
OK
start
Aggregator (on cloud)
listen for connecting participants
start
Participant 1 (on edge node 1)
start
Participant 2 (on edge node 2)
join federation
join federation
For N rounds
train global model on local dataset
partial model
train global model on local dataset
partial model
aggregate partial models to update global model
updated global model
updated global model
final model
store final model on ALIDA cloud
store final model locally
store final model locally

Figure 19: FML training example with ALIDA platform

According to the sequence diagram above, the typical process of Federating Training of models with ALIDA involves the following steps.

1. The User starts the Aggregator pipeline on ALIDA platform.
2. ALIDA starts the execution of the Aggregator pipeline.
3. The Aggregator starts to listen for connecting pipeline Participants.
4. The User starts - on two different edge nodes - Participant 1 and 2 pipelines.
5. The Participants contact the Aggregator to join the training federation.
6. Participants and Aggregator interact[5] with each other for N rounds to carry out the training. The number of rounds can be configured by the developer.

[5] *The interaction between Aggregator and Participants during the training is entrusted to the Flower library mechanisms and it is transparent to the end users.*

7. At the end of the training, a copy of the final model is stored by Participants locally and by the Aggregator on the ALIDA cloud.

2.9.5 Internal Architecture

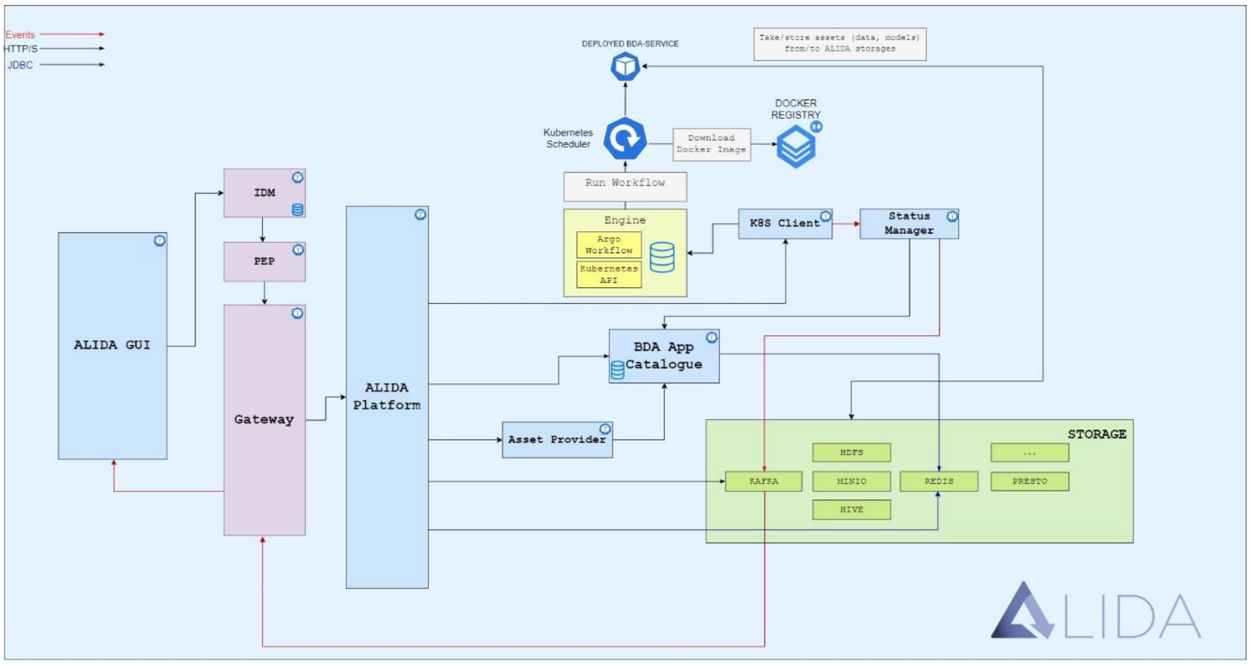


Figure 20: ALIDA internal architecture

- *ALIDA GUI*: the graphical user interface of ALIDA, based on React JavaScript UI library.
- *ALIDA Platform*: the main ALIDA back-end service, responsible for managing all requests coming from the GUI and other components.
- *BDA Application Catalogue*: contains all operations (creation, reading, updating, deletion) related to the objects managed in ALIDA: datasets and models metadata, BDA services, BDA applications, and more.
- *Asset Provider*: component of ALIDA responsible for parsing the BDA application pipeline generated through the ALIDA graphical designer and providing assets such as exporting the BDA application and the generated model.
- *K8S Client*: component for executing/stopping BDA applications workflows and sending BDA application events that are executed in ALIDA, interacting with K8S and Argo Workflow APIs.
- *Status Manager*: based on incoming status events, manages, and updates the states of the BDA applications by interfacing with the *BDA Catalogue*.

2.9.6 API

Notes:

- The Flower open-source framework (https://flower.dev/) has been used to enable Federated Machine Learning (FML) applications. The framework allows to focus on the coding of the training part placed at each participant and how the generated model weights can be subsequently aggregated by the aggregator server, including defining multiple strategies. Data exchange between a participant and an aggregator server is done - *transparently to the user by the Flower internal mechanisms* - using the gRPC (Remote Procedure Call) protocol, a protocol originally optimized by Google more than RPC, which uses protocol buffers and HTTP 2 for data transfer. Because it is a machine-to-machine transfer, gRPC is faster than REST call invocations for both writing and reading. In addition, gRPC supports the use of authentication mechanisms via SSL/TLS to authenticate the server and encrypt all data exchanged between clients and the (model aggregator) server.

- ALIDA REST API description is not publicly available.

REST API

BDA Application Controller

Endpoint	Summary
POST /api/v2/bundle/bdas	Register a new BDA application
GET /api/v2/bundle/bdas/**	Get all the registered BDA applications the logged user can get
DELETE /api/v2/bundle/bdas/{id}	Delete a BDA application by ID
PUT /api/v2/bundle/bdas/{id}	Updates an existing BDA application
POST /api/v2/bundle/start/bdas/{bdaId}/workflows/{workflowId}	Start a BDA application by ID

POST /api/v2/bundle/stop/bdas/{bdaId}/workflows/{workflowId} }	Stop a BDA application by ID

Models Controller

Endpoint	Summary
GET /api/object/models	Get all the machine learning models metadata
POST /gateway/platform-alida/api/proxy/ap/export/bdas/{modelId}	Download a machine learning / deep learning model produced by a specific BDA application by the "modelId"

Registration Controller

Endpoint	Summary
GET /api/platform/services	Get metadata about all the registered BDA-services in the ALIDA catalogue
POST /api/platform/services	Register a new BDA-service
DELETE /api/platform/services/{id}	Unregister a BDA- service
GET /api/platform/services/{id}	Get metadata about a specific BDA-service

2.9.7 Implementation Technology

Programming Language:

- Python: micro-services for the workflow parser, and for communicating with the Kubernetes and Argo Workflows APIs, were built in Python programming language to execute workflows designed through the ALIDA GUI (graphical user interface).
- Java: most of the micro-services of the ALIDA platform were developed in Java, especially the management of the ALIDA catalogue, and the workflow status manager, was developed using Java libraries.
- JavaScript: was used for the realization of the graphical user interface of ALIDA, along with the use of JavaScript libraries such as React.

Platform:

- Kubernetes: the micro-service based ALIDA platform is run and managed within a Kubernetes cluster (K8s). This enables resource scaling and support of other native Kubernetes frameworks for orchestration and execution of micro-services workflows.
- Docker: the designed workflows (or pipelines) can be exported as Docker compose files referencing Docker images published on a Docker registry. Therefore, they can be run on any node supporting Docker and able to access the registry to download the Docker images.

Libraries/Frameworks

- Flower - used as a framework to enable Federated Machine Learning in BDA services.
- Argo Workflows - used for the creation and execution of workflows of BDA services within a K8s cluster.
- Apache Spark - big data analytics framework used in BDA services for processing and training models on large amounts of data.
- Apache Kafka - message bus framework used as a means of communication between streaming BDAs services.
- MinIO - object storage used for historicizing and storing data and trained ML/DL models.
- Redis - an in-memory data structure store used to store metadata, useful for the ALIDA platform.
- HDFS - distributed file system for storing and historicizing data and trained ML/DL models.
- Hive - database to store tabular datasets.
- FIWARE KeyRock - used by ALIDA as identity management.
- Spring Boot - framework used to develop the main components of the ALIDA backend.
- React - library used to develop the front-end component of the ALIDA platform.
- React Flow - library used for the development of the workflows builder.
- ALIDA allows users to write custom pipeline blocks, BDA (Big Data Analytics) services by using Python starting from the provided templates (https://gitlab.alidalab.it/external/templates-bda-services). The user can include the Python the own or imported/installed machine learning / deep learning / other libraries as:
    - TensorFlow
    - Keras
    - Scikit-learn
    - Pandas

2.9.8 Comments

None.

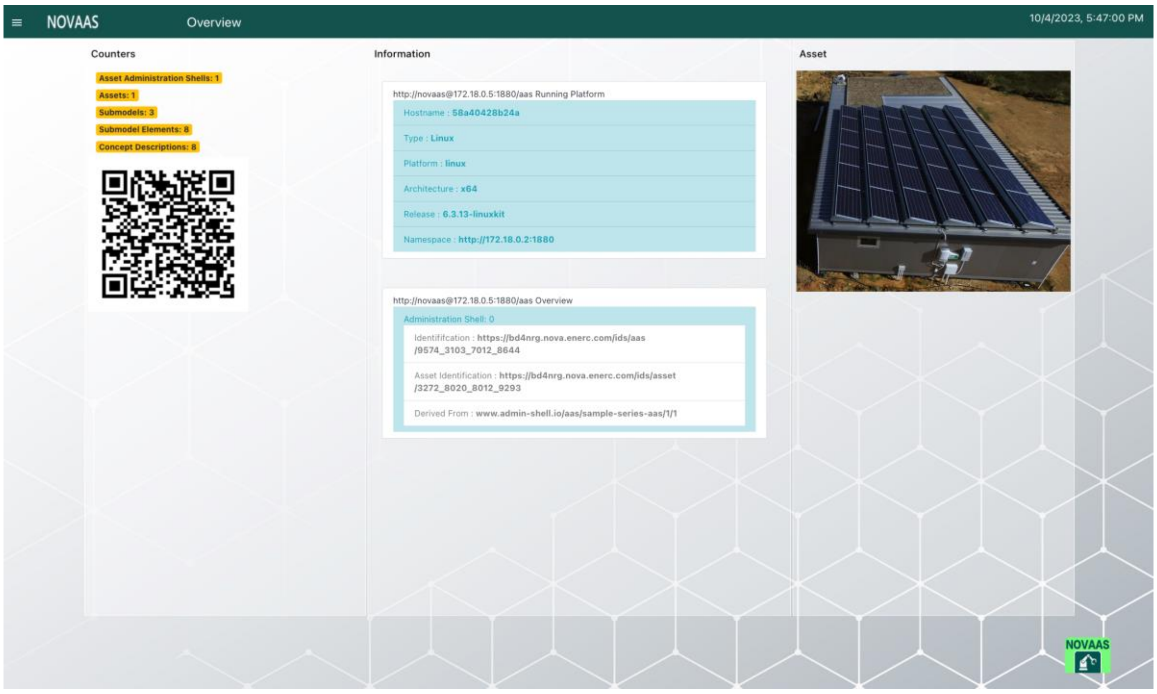
2.10 Component Name: NOVA Asset Administration Shell (NOVAAS)

2.10.1 Description

NOVAAS is an open-source implementation of the RAMI4.0 Asset Administration Shell concept. The Asset Administration Shell (AAS) is a key concept within the context of the Industrial Internet of Things (IIoT) and Industry 4.0. It is a standardized framework for describing and managing industrial assets and their digital representations ("data image" of the asset) in a way that enables seamless interoperability between various components and systems in industrial environments. The AAS is used to digitalize any physical asset in order to be integrated seamlessly into an I4.0 compliant system. Briefly, the AAS implements the Digital Twin concept for Industry 4.0. NOVAAS is currently compliant with the V2 specification of the AAS. We are now working on giving support to the newer V3 specification of the AAS. The tool is implemented using next-generation software development principles i.e. using low/no code platforms, microservices and APIs and containers. In the context of RE4DY the NOVAAS will be used for designing and developing I4.0 compatible Digital Twins. It will be used to provide and harmonized and standardized connection to the physical asset within a manufacturing production system. Moreover, NOVAAS can be used to run AI algorithms at the edge for several applications such as quality inspection, predictive maintenance, dashboarding etc.. Finally, the NOVAAS (as an implementation of the AAS concept) will enable the creation of data pipelines for seamless data flows across the production lifecycle (Digital Thread).

2.10.2 Input

- The tool has a user interface available on ip:port/ui that shows the description of the physical asset in terms of submodels and related properties. The submodels and properties can be used to model both static and dynamic information. Each property can be semantically described using the concept description class (this is also shown within the UI). The UI also provides plotting capabilities. It allows users to plot the real time data extracted from the physical asset.

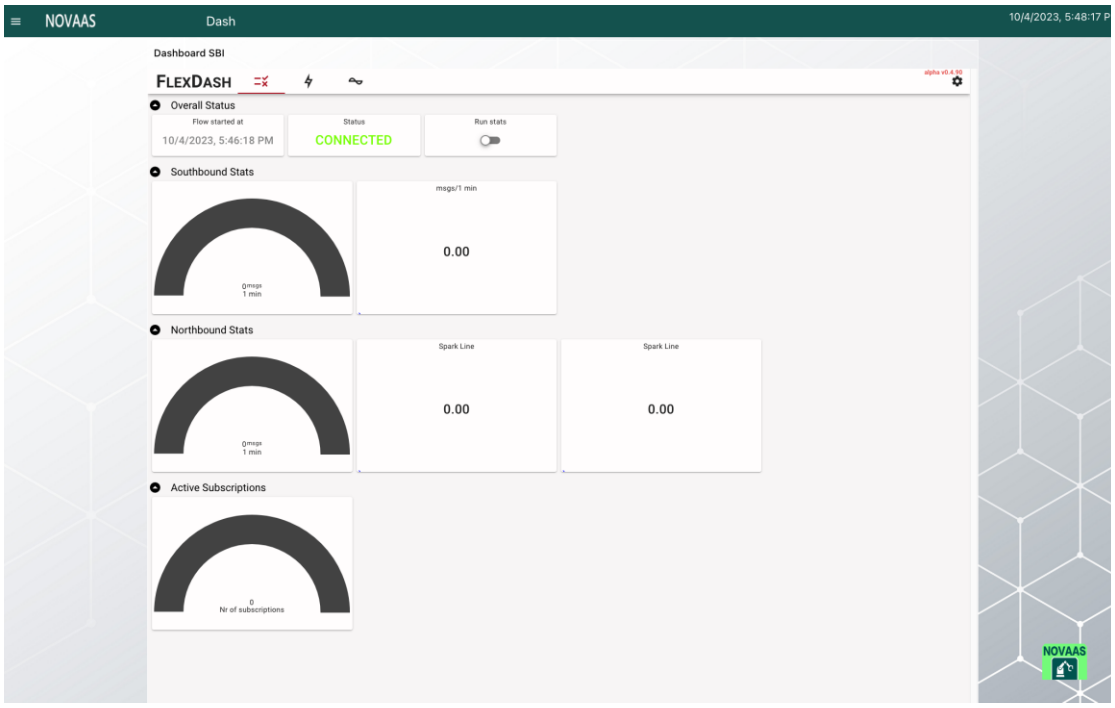


Asset Administration Shells: 1
Assets: 1
Submodels: 3
Submodel Elements: 8
Concept Descriptions: 8

http://novaas@172.18.0.5:1880/aas Running Platform
Hostname : 58a40428b24a
Type : Linux
Platform : linux
Architecture : x64
Release : 6.3.13-linuxkit
Namespace : http://172.18.0.2:1880

http://novaas@172.18.0.5:1880/aas Overview
Administration Shell: 0
Identififcation : https://bd4nrg.nova.enerc.com/ids/aas/9574_3103_7012_8644
Asset Identification : https://bd4nrg.nova.enerc.com/ids/asset/3272_8020_8012_9293
Derived From : www.admin-shell.io/aas/sample-series-aas/1/1

Figure 21: NOVAAS - Overview



Flow started at: 10/4/2023, 5:46:18 PM
Status: CONNECTED
Run stats

0 msgs 1 min
msgs/1 min: 0.00

0 msgs 1 min
Spark Line: 0.00
Spark Line: 0.00

0 Nr of subscriptions

Figure 22: NOVAAS - Dashboard

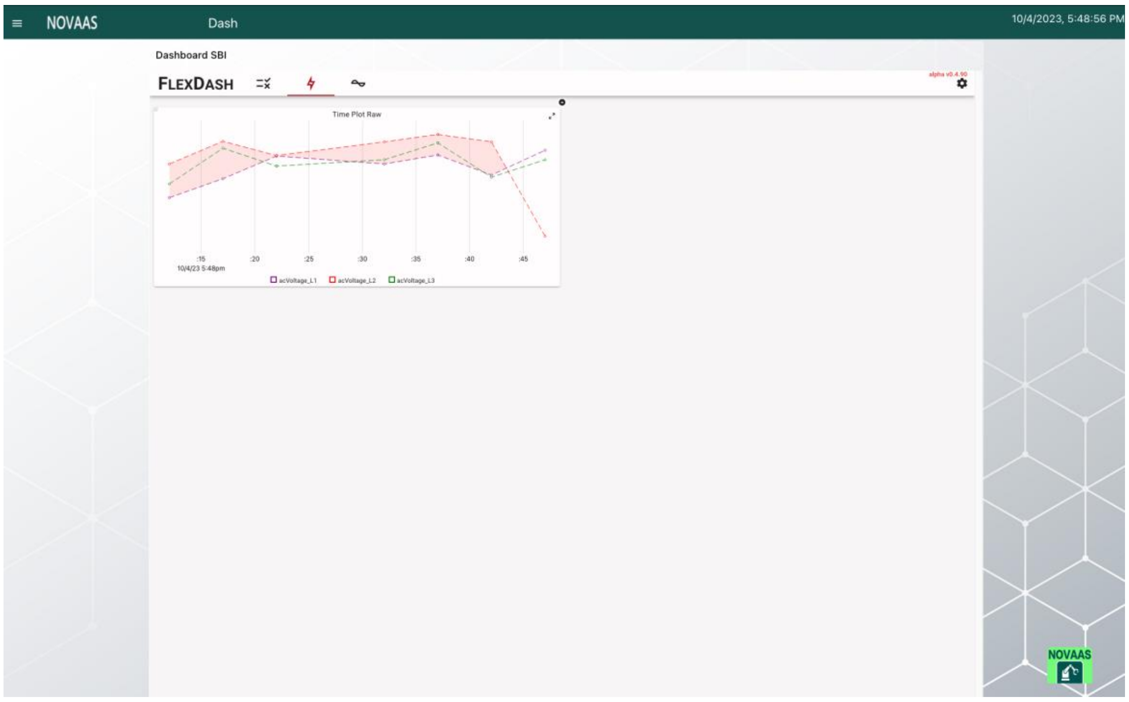
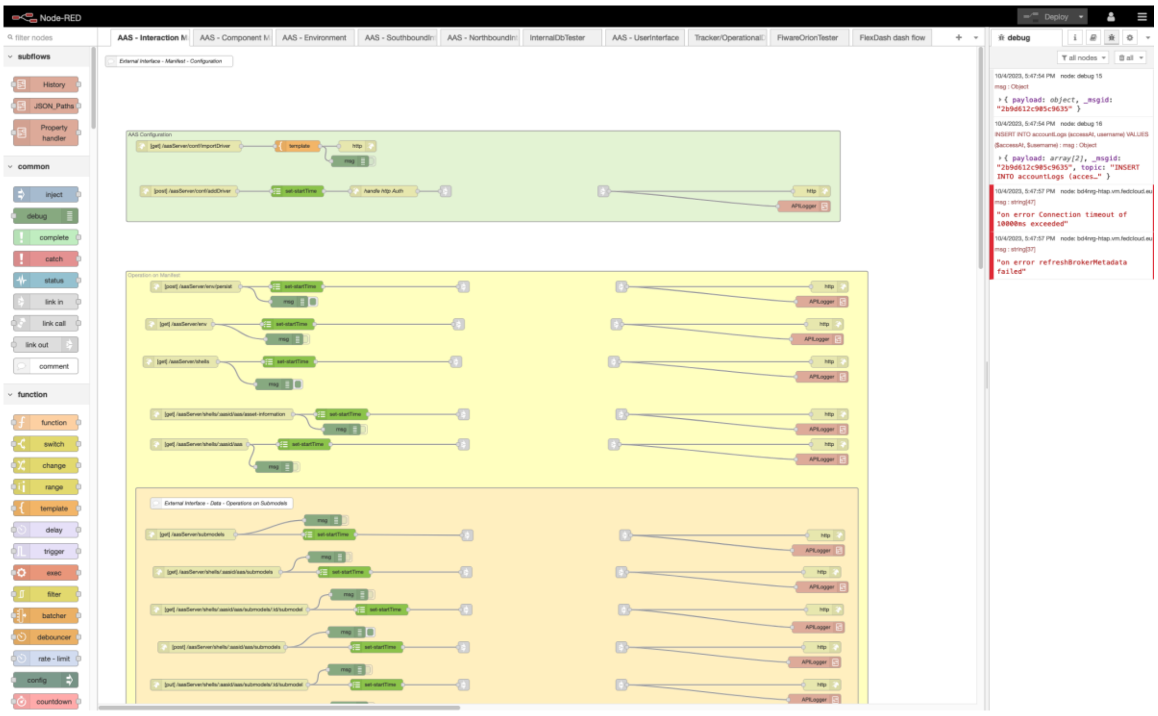


Figure 23: NOVAAS – Dashboard plot

- The backend of the tool is accessible by using the ip:port link (see the image below)



```
10/4/2023, 5:47:54 PM   node: debug 15
msg : Object
 ▸{ payload: object, _msgid:
 "2b9d612c985c9635" }
10/4/2023, 5:47:54 PM   node: debug 16
INSERT INTO accountLogs (accessAt, username) VALUES
($accessAt, $username) : msg : Object
 ▸{ payload: array[2], _msgid:
 "2b9d612c985c9635", topic: "INSERT
 INTO accountLogs (acces…" }
10/4/2023, 5:47:57 PM   node: bd4nrg-htap.vm.fedcloud.eu
msg : string[47]
 "on error Connection timeout of
 10000ms exceeded"
10/4/2023, 5:47:57 PM   node: bd4nrg-htap.vm.fedcloud.eu
msg : string[37]
 "on error refreshBrokerMetadata
 failed"
```

Figure 24: NOVAAS – Backend interface

- The tool is totally generic, however it requires the development of a simple connector to enable the connection with the asset. The technology used depends on the asset, however the tool supports a large number of connectors (ROS, Modbus, OPC-UA, MQTT, HTTP, etc) basically the whole ecosystem provided by Node-Red (that is the development platform).

2.10.3 Output

- All the data from the asset can be accessed by using the REST API (standardized API).
- Event-based communication is also supported. The technology adopted is MQTT and the events follow the V2 AAS specifications.

2.10.4 Internal Architecture

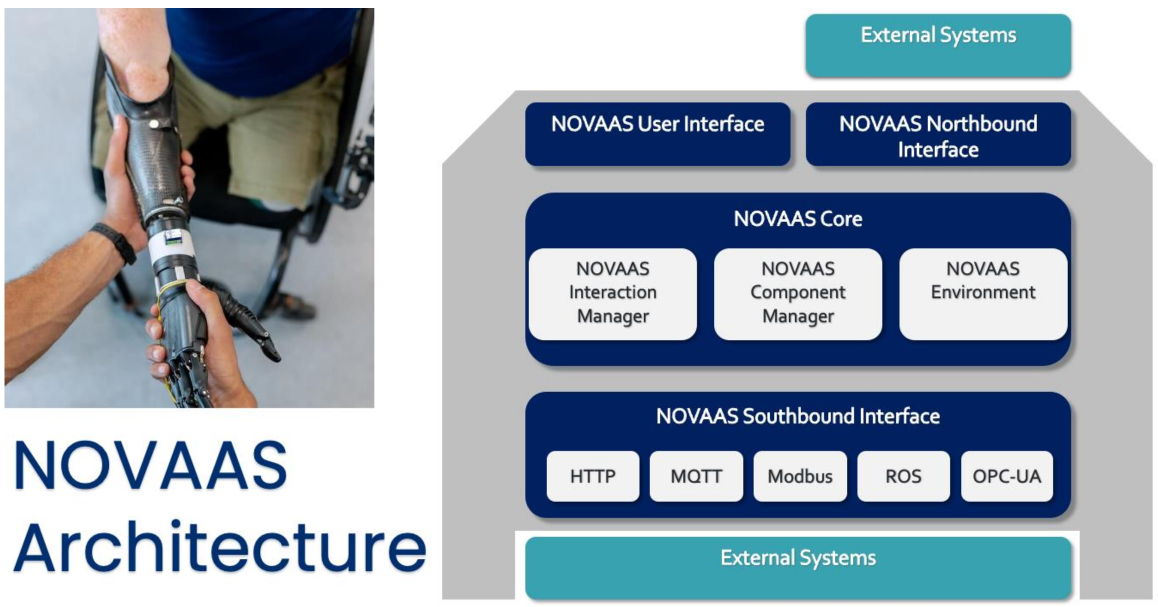
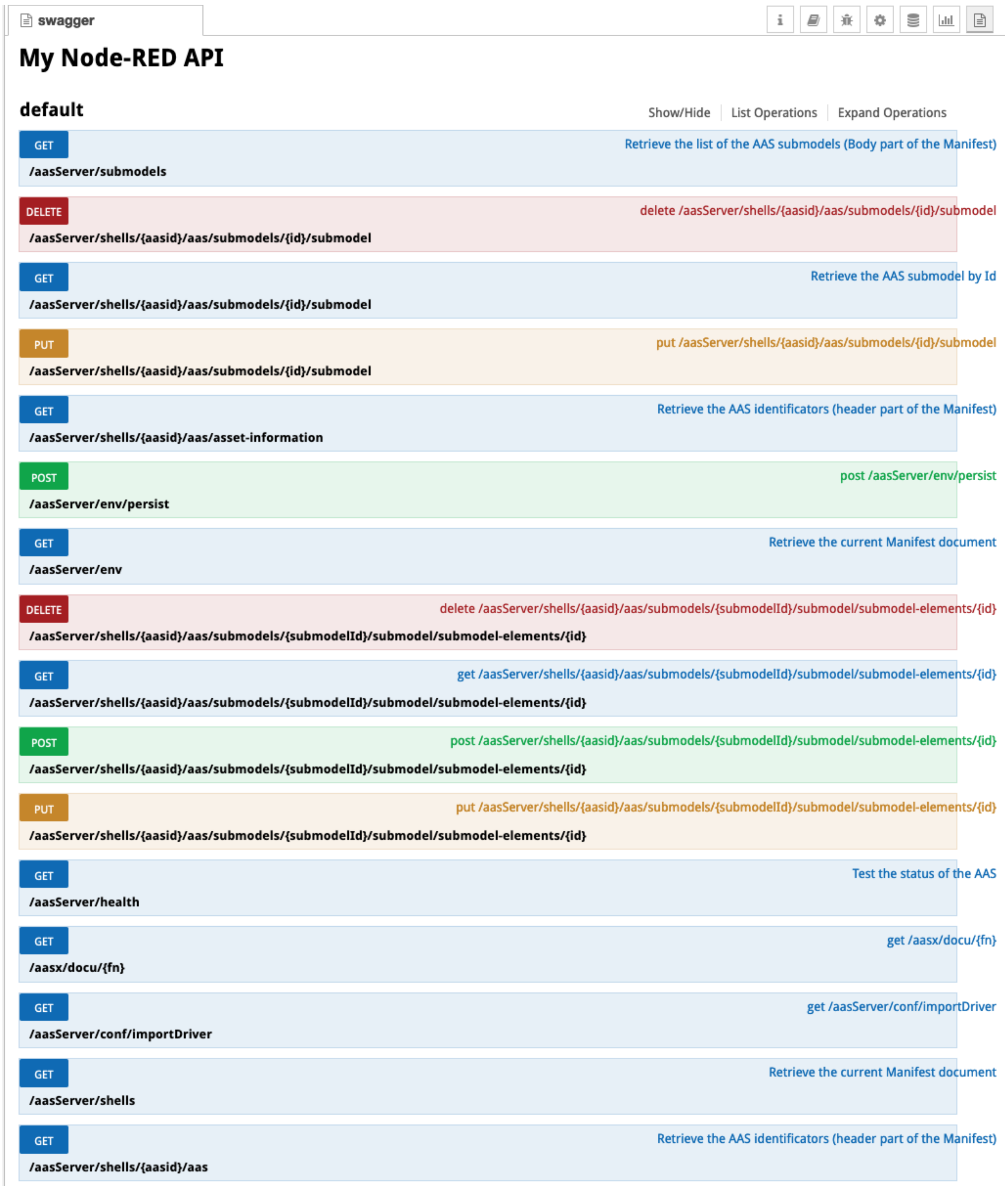


Figure 25: NOVAAS – Internal Architecture

2.10.5 API

- The API is documented in V2 AAS specifications that can be retrieved from: https://industrialdigitaltwin.org/en/
- Internally the API is also documented using a swagger plug in.



swagger

My Node-RED API

default

Show/Hide | List Operations | Expand Operations

GET /aasServer/submodels — Retrieve the list of the AAS submodels (Body part of the Manifest)

DELETE /aasServer/shells/{aasid}/aas/submodels/{id}/submodel — delete /aasServer/shells/{aasid}/aas/submodels/{id}/submodel

GET /aasServer/shells/{aasid}/aas/submodels/{id}/submodel — Retrieve the AAS submodel by Id

PUT /aasServer/shells/{aasid}/aas/submodels/{id}/submodel — put /aasServer/shells/{aasid}/aas/submodels/{id}/submodel

GET /aasServer/shells/{aasid}/aas/asset-information — Retrieve the AAS identificators (header part of the Manifest)

POST /aasServer/env/persist — post /aasServer/env/persist

GET /aasServer/env — Retrieve the current Manifest document

DELETE /aasServer/shells/{aasid}/aas/submodels/{submodelId}/submodel/submodel-elements/{id} — delete /aasServer/shells/{aasid}/aas/submodels/{submodelId}/submodel/submodel-elements/{id}

GET /aasServer/shells/{aasid}/aas/submodels/{submodelId}/submodel/submodel-elements/{id} — get /aasServer/shells/{aasid}/aas/submodels/{submodelId}/submodel/submodel-elements/{id}

POST /aasServer/shells/{aasid}/aas/submodels/{submodelId}/submodel/submodel-elements/{id} — post /aasServer/shells/{aasid}/aas/submodels/{submodelId}/submodel/submodel-elements/{id}

PUT /aasServer/shells/{aasid}/aas/submodels/{submodelId}/submodel/submodel-elements/{id} — put /aasServer/shells/{aasid}/aas/submodels/{submodelId}/submodel/submodel-elements/{id}

GET /aasServer/health — Test the status of the AAS

GET /aasx/docu/{fn} — get /aasx/docu/{fn}

GET /aasServer/conf/importDriver — get /aasServer/conf/importDriver

GET /aasServer/shells — Retrieve the current Manifest document

GET /aasServer/shells/{aasid}/aas — Retrieve the AAS identificators (header part of the Manifest)

2.10.6 Implementation Technology

- Node-RED: Node-RED is an open-source flow-based development tool and runtime environment for connecting hardware devices, APIs, online services, and various software applications. It provides a visual programming interface that allows users, often referred to as "makers" or developers, to create and deploy Internet of Things (IoT) applications and automation tasks without the need for extensive coding. Node-RED is a flow-based, visual development platform and runtime environment that simplifies the creation of IoT applications and automation workflows. It offers a web-based, drag-and-drop interface for building and connecting nodes (blocks) that represent different functions and devices. These nodes can include data sources, data processors, output actions, and more. Users can create complex workflows by connecting these nodes in a visual manner, and Node-RED handles the underlying logic and communication between components.
- JavaScript: most of the more complex logic is implemented using JavaScript.

Platform:

- NOVAAS has been built since the very beginning to be cloud-native.
- Kubernetes
- Docker

2.10.7 Comments

None

2.11 Component Name: eIDAS

2.11.1 Description

The eIDAS solution adopts an electronic identification scheme that may be used as proof of identity for individuals across Europe. It enables citizens of a European Union Member State to verify their identity when using online services in other Member States, thus facilitating electronic transactions. The eIDAS network is composed of a number of eIDAS nodes, one per Member State implemented at the national level, and each node may either request or provide cross-border authentication. In the RE4DY context, the functionality provided by eIDAS will be utilized for the identity management of the RE4DY user base. In particular, the eIDAS node will provide authentication for individuals joining the RE4DY platform during registration, allowing secure access to our services.

2.11.2 Input

Since the eIDAS node does not use a database, but plain text configuration files to record users, we have developed a Python API which is responsible for making the necessary changes in these files so as to create new users. When adding a new eIDAS user to the system, it is necessary to record a set of attributes such as first and last names, national identification number and gender. These attributes constitute the input of the endpoint implemented by our Python API. It is also possible to create new, custom attributes that may be stored along with the given user information.

Furthermore, the first phase of the user registration process requires first authenticating with eIDAS. We have developed an authentication back-end using Spring Boot which is responsible for communicating with the eIDAS node during this process. The back-end service provides an endpoint through which authentication is possible. It accepts a username and password which are then forwarded to the eIDAS node to fulfill the request.

2.11.3 Output

In the eIDAS node, upon successful user creation, the API returns the message "User inserted Successfully". If the username or national identification number of the user provided have already been recorded under a user that already exists, the process fails with "User Already Exists".

Our Spring Boot authentication back-end is responsible for interfacing with the eIDAS node to achieve authentication. Upon successful authentication with eIDAS, it returns a JSON object, eidas_response, embedded with all the related attributes stored in the eIDAS node as well as an assertion from the eIDAS node.

2.11.4 Information Flow

Adding an eIDAS user

An eIDAS administrator is able to add a new eIDAS user through the eIDAS Management Web UI. The administrator inserts user credentials and information into a form and the Management Web UI forwards them to the eIDAS API. Since the eIDAS node does not use a database but file-based storage for users, the relevant files are modified so that the new user is registered in the eIDAS node. If the user was inserted successfully, the administrator user is notified in the front-end.

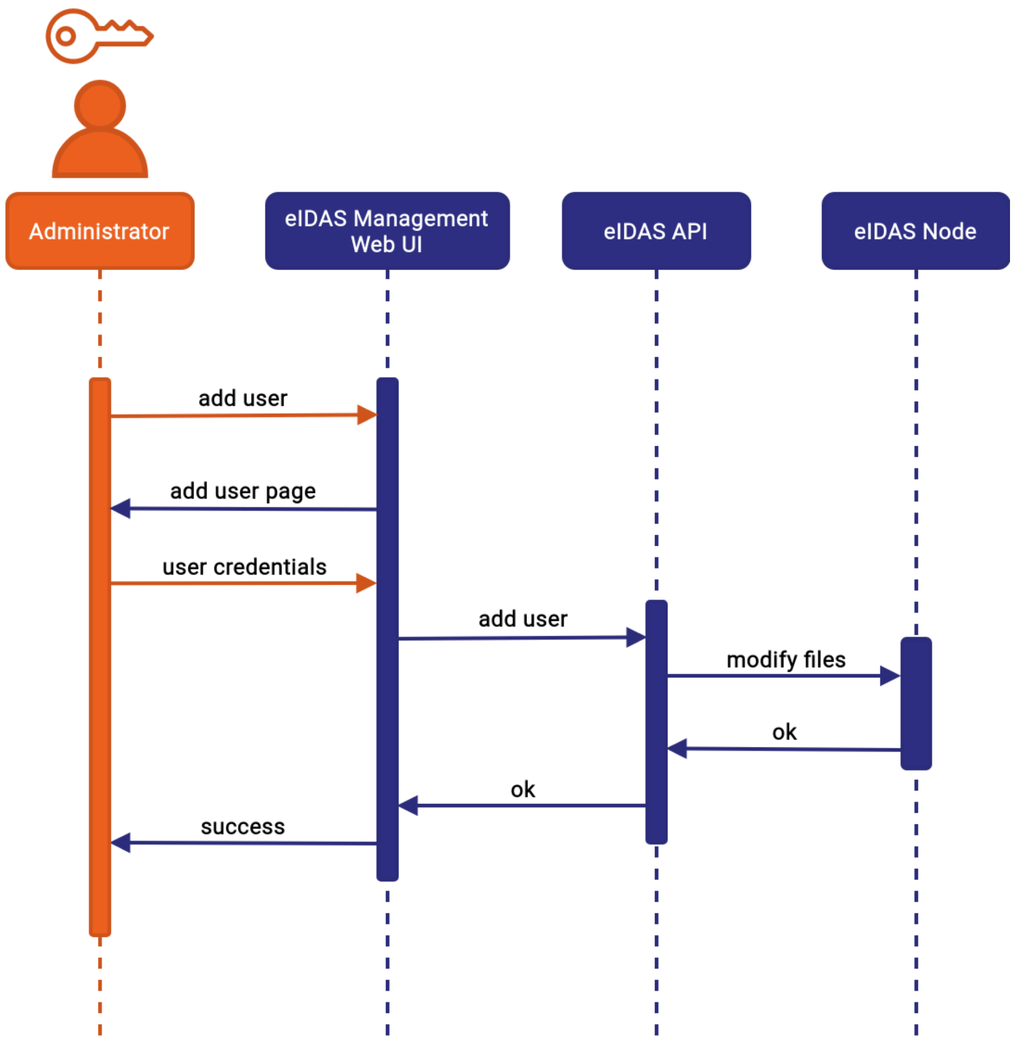


Figure 26: UML Diagram of the administrator adding a new eIDAS user to the system and the communication of the eIDAS management front-end, eIDAS API and eIDAS node.

RE4DY Registration

Registering begins through the RE4DY front-end application where the user is initially presented with an eIDAS login form to insert their eIDAS credentials. The credentials are sent from the front-end to the authentication back-end service and from there they are sent to the eIDAS node for authentication. If the credentials match with an existing eIDAS user, the eIDAS node returns an access token to the authentication back-end which is then forwarded to the front-end. This is where the involvement of the eIDAS component ends and the user is allowed to create their RE4DY account.

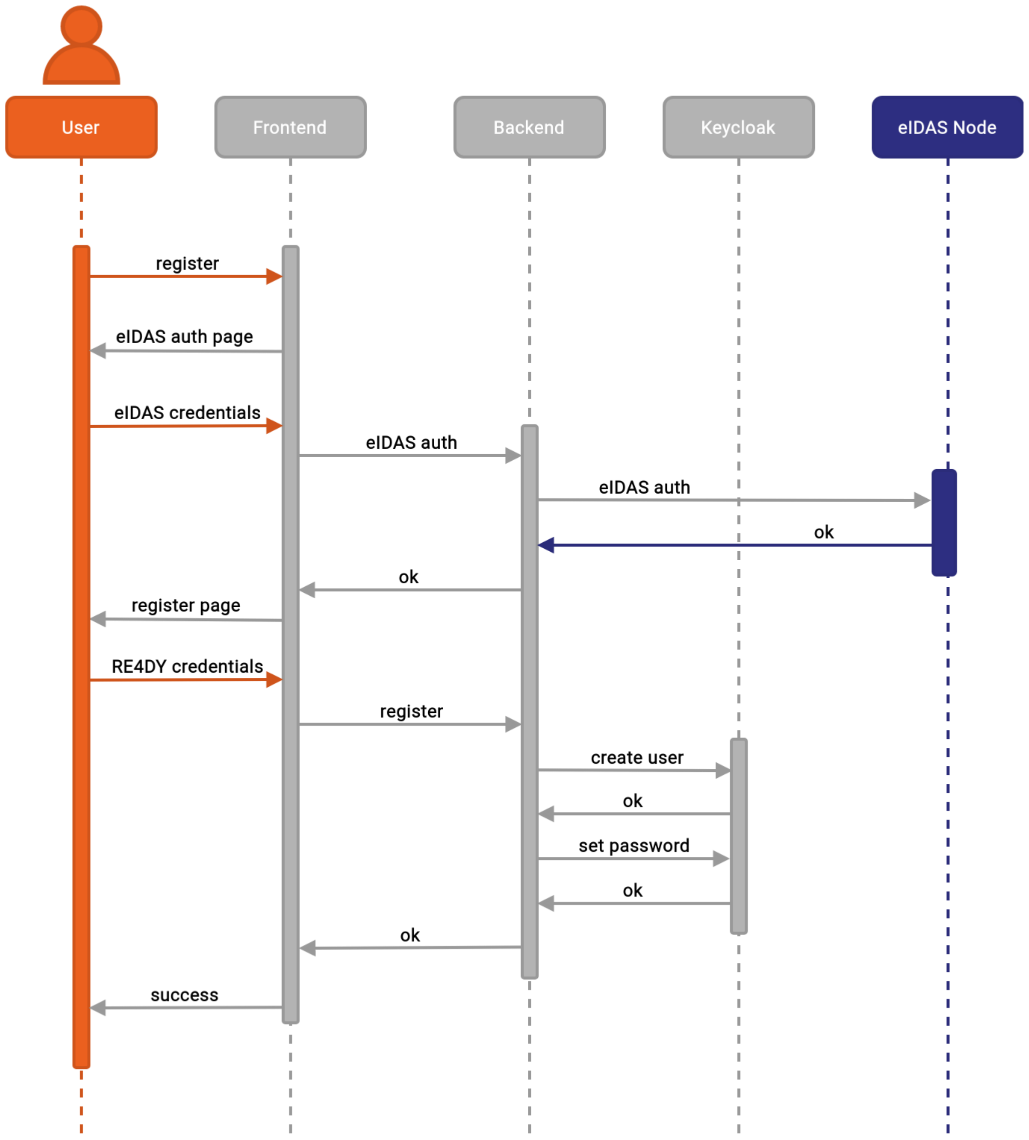


Figure 27: UML Diagram of the user registration process and the communication of the front-end, authentication back-end, Keycloak and eIDAS services.

2.11.5 Internal Architecture

For the facilitation of various operations of the eIDAS node, an eIDAS API has been implemented. This is because the eIDAS node does not use a database which means internal files need to be directly modified for its configuration as well as the registration of new eIDAS users. The eIDAS API may be used by the eIDAS Management Web UI for the creation of users or by the authentication back-end during the first stage of the RE4DY registration, which involves authenticating with eIDAS credentials.

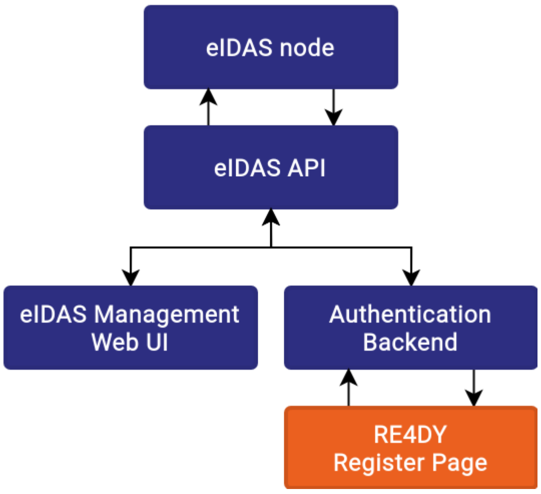


Figure 28: Internal architecture of the eIDAS component and related services

2.11.6 API

- POST {{eidas-api}}/addUser

- Input:

```
{
"username": "johndoe",
"password": "12345678",
"name": "John",
"surname": "Doe",
"identifier": "ZX1234567",
"birthday": "1986-05-06",
"placeOfBirth": "Greece,+Athens",
"extraName": "someCustomAttribute",
"extra": "someCustomAttributeValue",
"gender": "Male",
"phone": "1234567891"
  }
```

- Output (HTML):

```
<!-- on success -->
<div>User inserted Successfully</div>
<div class="message">User inserted Successfully</div>
...

<!-- on duplicate username or identifier -->
<div class="error">User Already Exists</div>
```

- Description:

  This endpoint is implemented by the API developed to facilitate the user creation process in the eIDAS node. The username and identifier fields are unique per user, meaning that if another user exists with the same username or identifier, the user creation will fail.

- POST {{auth-backend}}/auth/authenticate

- Input:

```
{
"username": "newuser",
"password": "1234",
}
```

- Output:

```
{
"eidas_response": "eyJ0e...",
}
```

- Description:

  This endpoint is implemented by our Spring Boot authentication back-end. It interfaces with the eIDAS node to determine whether the provided credentials are associated with an existing eIDAS user. On success, the returned eidas_response may be decoded to retrieve the assertion given by the eIDAS node as well as the attributes which correspond to the user.

The API documentation is also available as a Swagger editor .yaml file.

2.11.7 Implementation Technology

The eIDAS artifact is officially distributed as a zipped archive which includes both binaries and source code required to execute the node. It is written in Java and utilizes the Tomcat web server for this purpose. We have created a Docker image which installs the required dependencies and initializes the eIDAS node, the main goal being to simulate the functionality of the eIDAS Node of a European Union Member State.

Furthermore, we have developed a small Python API to facilitate the management and configuration of the node and the creation of users and roles, since these procedures require the modification and replacement of settings across several server configuration files. We have also developed a minimal web page using pure HTML, CSS and JavaScript to additionally provide a graphical interface through which these procedures may be performed.

For the authentication, authorization, and user management throughout the RE4DY project, we have implemented a Spring Boot back-end, written in Java. The back-end communicates both with the eIDAS node and with Keycloak, our Identity Manager of choice. Finally, to demonstrate the eIDAS-related functionality, a proof-of-concept user registration and login web UI has been developed using the React framework.

2.11.8 Comments

None.

2.12 Component Name: F-UJI (FAIRsFAIR Research Data Object Assessment Service)

2.12.1 Description

F-UJI is a tool for data FAIRness assessment which provides the ability to specify a data set identifier (e.g. DOI, URL) in order to generate an evaluation based on the seventeen FAIRness principles released by GO FAIR. Optionally, a metadata service (OAI-PMH, SPARQL, CSW) endpoint URI may be provided as input to identify additional information about the data set. The assessment is automated, leveraging the metadata included with or embedded in the data set and providing a score as well as pass or fail statements for each metric. In the context of RE4DY, the tool will be utilized to programmatically assess the FAIRness of data sets as a means of compliance check and assurance. The functionality offered by F-UJI will be integrated with other tools and components.

2.12.2 Input

The F-UJI tool provides its FAIRness assessment services by implementing a REST API. Regarding the evaluation, the automated F-UJI tool requires an identifier of a data set, such as a DOI or a URL as input. It also provides an endpoint that accepts a metadata service (OAI-PMH, SPARQL, CSW) endpoint URI to fetch additional metadata of the data set to be assessed. The input may be provided either by an individual or another service in the RE4DY platform. Moreover, apart from the Python implementation of the tool, F-UJI has been integrated into a web demo which provides the evaluation output in a more human-friendly format.

2.12.3 Output

Based on the metadata extracted from the dataset, the FAIRness assessment results are returned as a JSON object and include pass or fail statements and the score earned for each FAIRness metric.

2.12.4 Information Flow

To evaluate the FAIRness of a dataset, it is possible to directly call the REST API offered by F-UJI using the URL of the dataset to be evaluated, as shown in the diagram below. The procedure is identical for the rest of the API endpoints offered by F-UJI.

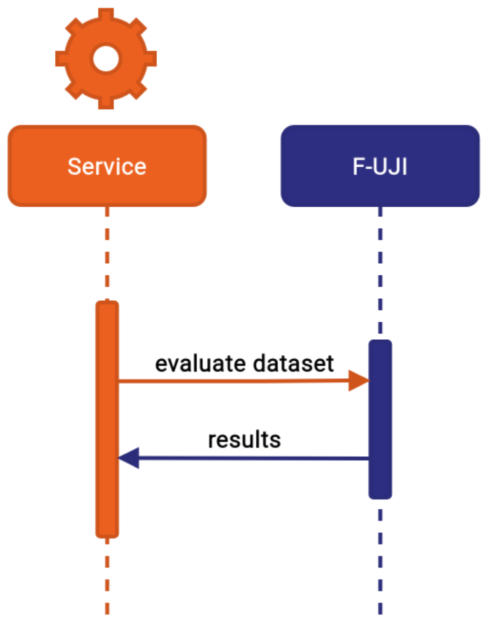


Figure 29: UML Diagram of the dataset evaluation process and the communication of a client service with the F-UJI component.

2.12.5 Internal Architecture

The F-UJI component utilizes a number of Python libraries to provide its functionality. Particularly, it uses rdflib and urllib to access network resources and extruct, tika and lxml to extract metadata from text of various file types. Finally, it uses Flask to implement and expose the API endpoints required for dataset evaluation and metadata harvesting.

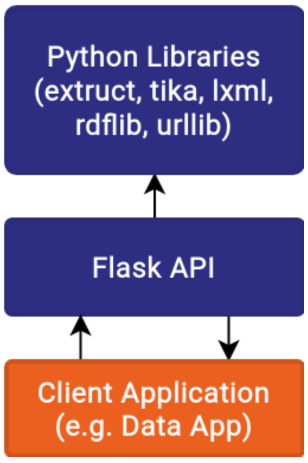

Python Libraries (extruct, tika, lxml, rdflib, urllib)
Flask API
Client Application (e.g. Data App)

Figure 30: Internal architecture of the F-UJI component and related libraries and services

2.12.6 API

```
•     POST /evaluate

–     Input
      {
    "metadata_service_endpoint":
"http://ws.pangaea.de/oai/provider",
    "metadata_service_type": "oai_pmh",
    "metric_version": "metrics_v0.5",
    "object_identifier": "https://doi.org/10.1594/PANGAEA.908011",
    "test_debug": true,
    "use_datacite": true
}

–     Output
      Some fields are truncated or omitted for brevity.

      {
    "metric_specification":
"https://doi.org/10.5281/zenodo.6461229",
    "metric_version": "metrics_v0.5",
    "resolved_url": "https://doi.pangaea.de/10.1594/PANGAEA.908011",
    "results": [
        {
            "id": 3,
            "maturity": 3,
            "metric_identifier": "FsF-F2-01M",
            "metric_name": "Metadata includes descriptive...",
            "metric_tests": {
                "FsF-F2-01M-1": {
                    "metric_test_maturity": 1,
                    "metric_test_name": "Metadata has been...",
                    "metric_test_score": 0.5,
                    "metric_test_status": "pass"
                },
                // ...
            },
            "score": {
                "earned": 2,
                "total": 2
            },
            "test_status": "pass"
        },
    ],
    "summary": {
        // ...
    },
    // ...
}
```

- Description

This endpoint evaluates the FAIRness of a data object. It outputs a JSON object which includes metadata about the dataset, the results of the assessment with scores and pass or fail statements for each metric, and finally a summary of the results. The endpoint supports basic authentication which requires a username and password to be provided. Users and their credentials are managed through a configuration file which is read by the tool upon request. It returns 400 if an invalid data object identifier is provided and 401 if the authentication information passed with the request headers is missing or invalid.

• POST /harvest

- Input

```
      {
    "object_identifier": "https://doi.org/10.1594/PANGAEA.908011"
}
```

- Output

Some fields are truncated or omitted for brevity.

```
      {
    "metadata": [
        {
            "format": "application/ld+json",
            "metadata": {
                "access_free": true,
                "access_level": "unrestricted",
                "creator": "Robert Huber",
                "creator_first": "Robert",
                "creator_last": "Huber",
                "language": "en",
                "license": [
                    "https://creativecommons.org/licenses/by/4.0/"
                ],
                // ...
                "summary": "This data set contains...",
                "title": "Maximum diameter of..."
            },
            "method": "SCHEMAORG_EMBEDDED",
            "namespaces": [
                "http://schema.org/"
            ],
            "schema": "http://schema.org",
            "url": "https://doi.pangaea.de/10.1594/PANGAEA.908011"
        },
        // ...
        ,
    ],
    "target_uri": "https://doi.org/10.1594/PANGAEA.908011"
```

```
}

-     Description
This endpoint may be used to optionally harvest metadata given the
DOI of a data set.
```

```
•     GET /metrics/{version}

-     Output
Some fields are truncated or omitted for brevity.
      {
    "metrics": [
        {
            "created_by": "FAIRsFAIR",
            "date_created": "2020-07-08",
            "date_updated": "2020-11-25",
            "description": "A data object may be assigned with...",
            "evaluation_mechanism": "Identifier is considered...",
            "fair_principle": "F1",
            "metric_identifier": "FsF-F1-01D",
            "metric_name": "Data is assigned a globally unique...",
            "metric_number": 1,
            "metric_short_name": "Unique Identifier",
            "metric_tests": [
            ],
            "target": "Data",
            "test_scoring_mechanism": "alternative",
            "total_score": 1,
            "version": 0.5
        },
        // ...
    ],
    "total": 17
}

-     Description
When given a version of the FAIRness metrics (e.g. 0.5), it
retrieves and returns the specified version of the list of metrics
used to perform a data set's FAIRness assessment, their
descriptions, scores and additional metadata. This information is
loaded directly from the .yaml files available in the tool's source
code.
```

```
•     GET /metrics/{version}/{metric}

-     Output
      {
    "created_by": "FAIRsFAIR",
    "date_created": "2020-07-08",
    "date_updated": "2020-11-25",
    "description": "A data object may be assigned with a globally
unique identifier such that ...",
    "evaluation_mechanism": "Identifier is considered unique if...",
    "fair_principle": "F1",
    "metric_identifier": "FsF-F1-01D",
    "metric_name": "Data is assigned a globally unique identifier.",
    "metric_number": 1,
    "metric_short_name": "Unique Identifier",
     "metric_tests": [
     ],
     "target": "Data",
     "test_scoring_mechanism": "alternative",
     "total_score": 1,
     "version": 0.5
},

-      Description
       When given a version of the FAIRness metrics (e.g. 0.5) and a
metric identifier (e.g. FsF-F1-01D), it retrieves and returns a
single metric and its definition from the list of metrics used to
perform a data set's FAIRness assessment.
```

The API documentation is also available in the Annex as a Swagger editor .yaml file.

2.12.7 Implementation Technology

F-UJI is written in the Python programming language and internally utilizes Flask to provide the necessary API endpoints that implement the data FAIRness evaluation. The documentation presents two methods of deploying the tool, either by installing the dependencies and directly executing the F-UJI server or by using the provided Docker image. Since we have adopted a container-based deployment architecture based on Docker, we chose to deploy F-UJI in our premises using the Docker image.

2.12.8 Comments

This component has been set up, tested and deployed. We have configured the service to accept RE4DY-specific credentials for the API authentication process and we have added a new service which starts the F-UJI container in our deployment scripts. It is currently operational and ready to use in a standalone manner and be integrated with the rest of the RE4DY components.

2.13 Component Name: Data Provenance and Traceability application

2.13.1 Description

Hyperledger Fabric is an open source blockchain developed under the Hyperledger project of the Linux Foundation. It is mainly aimed at organizations wishing to deploy permissioned blockchain networks. Its modular architecture permits the customization of the blockchain network to meet specific needs. An example of this is the support for pluggable consensus algorithms. Moreover, it offers communication privacy through channels, a membership mechanism to restrict channel access and access control lists

(ACLs) to handle access to resources using various policies. It supports the deployment of smart contracts (chaincode) which allow the secure execution of custom processes and algorithms that modify the state of the network.

Within the context of the RE4DY project, Hyperledger Fabric will be used to deploy a permissioned blockchain and leverage its immutable and append-only transaction history for the implementation of our data provenance and traceability application. In particular, we will use the functionality offered by Hyperledger Fabric smart contracts to create a dataset lifecycle tracking system.

Communication with the Hyperledger Fabric network will be executed through a separate application responsible for communicating with the Fabric Gateway, an API running within network peers with the purpose of facilitating transaction submission.

Furthermore, Hyperledger Fabric is expected to have a minor role in the RE4DY user authentication process. Through a process initiated by the RE4DY authentication backend implemented in Spring Boot, RE4DY accounts will acquire certain attributes and will be associated with Hyperledger public addresses. Using Attribute-Based Access Control (ABAC) these addresses will be either allowed to or restricted from invoking chaincode methods that insert and update dataset information on-chain.

2.13.2 Input

As the chaincode will serve as a foundation for data provenance and traceability, the majority of the smart contract methods manage data entry on-chain. Depending on the contract method to be called, sending a transaction involves procuring various information regarding a dataset or its traceability events as transaction input. For instance, when recording the creation of a new dataset, it is necessary to provide a dataset ID, a number of dataset characteristics and a list of related datasets. Regarding retrieving on-chain data, Hyperledger Fabric supports JSON queries and key-based queries in its world state, a database that holds the current values of all objects.

2.13.3 Output

In Hyperledger Fabric, for each transaction invocation, the calling application is notified whether the transaction was valid, meaning it has been included in the chain, or invalid. For example, upon successful smart contract method invocation, the Fabric Gateway returns "Chaincode invoke successful" and includes the status code of the request, using the HTTP response status codes defined by RFC 9110. Finally, in cases where a value is returned with the function call, it is returned in JSON format.

2.13.4 Information Flow

Creating a dataset

A user may create a dataset by providing a dataset ID, a list of related dataset IDs and a number of dataset characteristics. Sending a transaction to the blockchain is required for such an operation. The tracking and management of the transaction lifecycle is the responsibility of the Fabric Gateway, which decouples the submission logic from the application where the transaction originates from. Initially, the client application sends a transaction proposal to the Fabric Gateway. The transaction proposal includes the name of the contract method to be invoked and optionally a number of arguments to the method. The Fabric Gateway forwards the transaction proposal to the Fabric Network. Peers in the Fabric Network execute the transaction locally to determine its validity according to the rules of the smart contract. If the transaction is valid, the peers endorse the transaction proposal. Depending on the policies agreed to by the members of the channel when the chaincode definition was approved, a certain number of endorsements are required. After the specified number of endorsements is collected, the Gateway submits the transaction and its endorsements and waits for commit status events. During this time, the new block which includes the transaction is propagated in the network and the peers validate all transactions included in it. When this happens, the transactions are committed to the ledger and the chain state is updated to reflect the changes made by these transactions. Finally, the Gateway offers the capability to access a minimal API for any client application to receive and handle chain code events.

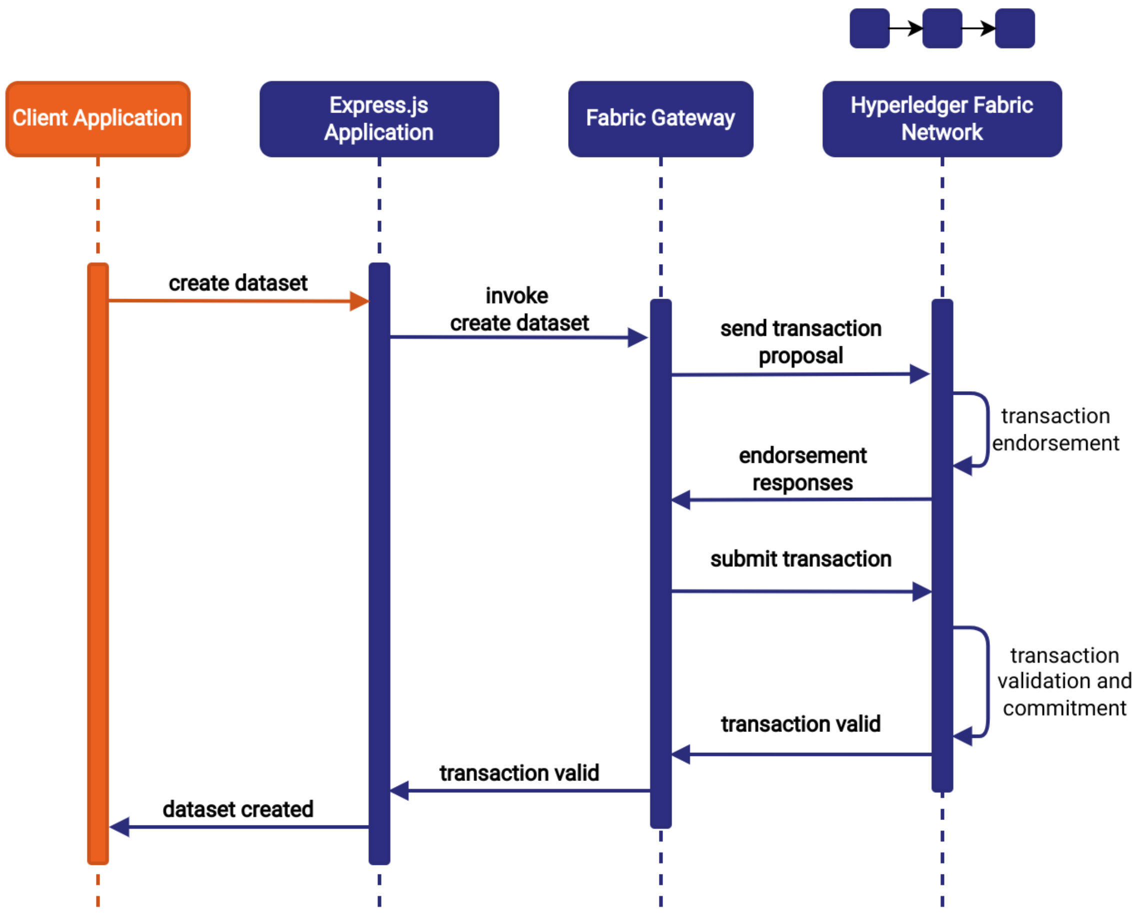


Figure 31: UML Diagram of the product creation process starting from the client application to the low-level transaction validation procedures taking place in the Fabric Network

Reading a dataset

A user provides a dataset ID in order to retrieve information for the related dataset. The Gateway will invoke the smart contract method which queries the state of the chain. The method will be executed on one Fabric Network peer and the function result will be returned to the Gateway and then the client application. It is noted that the Gateway will prefer a peer which is part of the same organization as the Gateway peer and select the peer with the highest block height, meaning the peer with the most up to date state of the chain. Also, the Gateway will ultimately select a peer from a different organization if no other peer is available in its own.

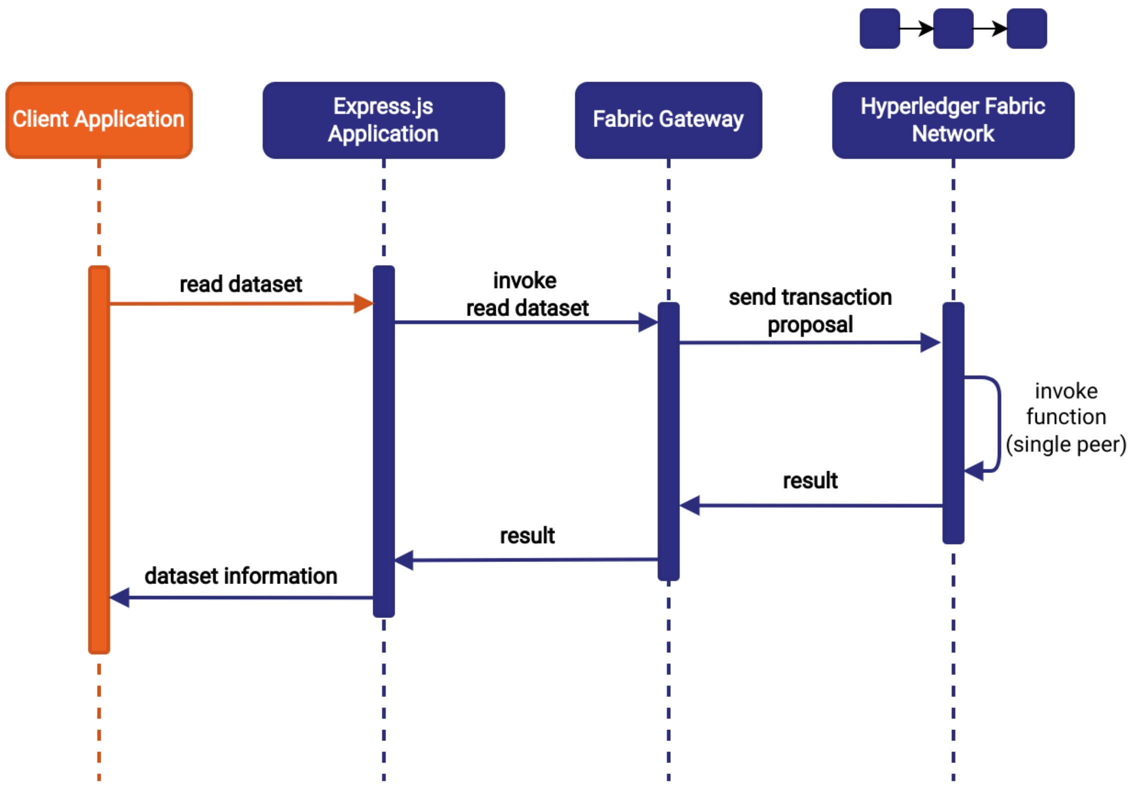


Figure 32: UML Diagram of the invocation of the product reading method of the RE4DY chaincode from the client application to the Fabric Network and back.

2.13.5 Internal Architecture

The Fabric Network is composed of a number of peers running peer software to participate in it.

The Fabric Gateway is a service running in every peer which offers the capability to use a minimal API for transaction submission. This facilitates the interaction with the Fabric Network as clients are not required to perform actions such as collecting transaction endorsements from other peers. Our Express.js application transacts with the Fabric Gateway and offers a single-entry point to the Fabric Network. Any client application as well as the authentication back-end may then interact with the chain using the Express.js application.

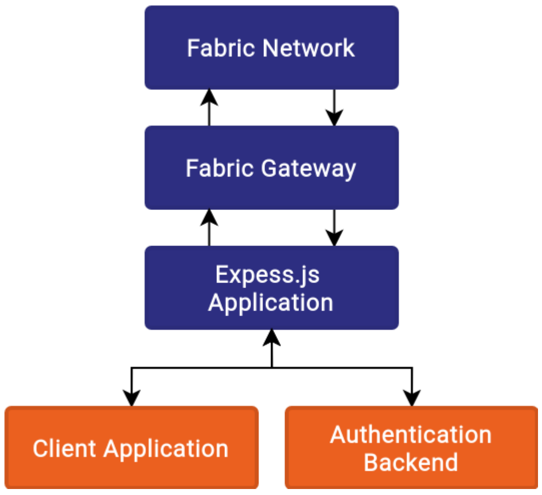


Figure 33: Internal architecture of the Hyperledger Fabric component and related services

2.13.6 API

The documentation of the Hyperledger Fabric component API is included in the Annex as a Swagger editor .yaml file.

2.13.7 Implementation Technology

In Hyperledger Fabric, chain code may be written either in the Go programming language, in JavaScript, using the node.js runtime environment and in Java. We have opted for JavaScript due to its ease of use and its large popularity in the Hyperledger Fabric community. Furthermore, there are application Software Development Kits (SDKs) available in JavaScript, Java, Python and Go. We will develop an application using Express.js which will implement an API that will provide an entry point to our Hyperledger Fabric blockchain network by interacting with it through the Fabric Gateway. In particular it will be used to invoke contract methods which encompass Create, Read, Update, Delete (CRUD) operations for datasets and related data structures.

Hyperledger Fabric peer software is offered as a collection of Docker images and helper scripts which may be used to set up and connect to a private Fabric network, deploy smart contracts, invoke chain code methods, or query the chain state. Peer software is required for each participating node. Moreover, we have created new auxiliary scripts to facilitate development. In particular, we developed a Bash script for redeploying a certain smart contract without being forced to destroy and recreate the chain and a script for sending transactions as certain peers to simulate transactions from different users or users of different organizations.

2.13.8 Comments

The design and implementation of this component is currently in progress and its integration with the rest of the RE4DY tools is currently ongoing.

2.14 Component Name: Keycloak

2.14.1 Description

Keycloak is an open-source identity and access management tool which plays a vital role in the authentication and authorization processes of the RE4DY project. It performs the storage and management of the RE4DY user base and provides authentication and authorization functionality in the form of libraries, APIs and user forms that enable a seamless integration with other tools. It provides Single-Sign On functionality which unifies the authentication and authorization procedures across the entire set of applications comprising RE4DY, allowing users to log in and log out once rather than performing this process for each RE4DY application individually. Moreover, the administrator may employ the Admin Console web UI to manage users and realms, as well as configure security and cryptography related settings. Finally, the Account Management Console grants users the capability to manage their account information and security.

2.14.2 Input

The Keycloak identity manager communicates with the back-end to provide the necessary authentication and authorization functionality. Therefore, it only receives input from the back-end during the registration and login phases.

Registration

During the user registration phase, the back-end is responsible for propagating attributes retrieved by the eIDAS node to Keycloak so that new users may be created. In order to create a new user, the user details that are sent to Keycloak as input are enumerated below:

- Username: Username given by the user during the registration process, after having authenticated with their eIDAS credentials. It is always the same as the user's eIDAS username.
- First Name: The user's first name, retrieved from the access token returned by the eIDAS node.

- Last Name: The user’s last name, retrieved from the access token returned by the eIDAS node.

In addition, in the current implementation, four custom attributes are sent to Keycloak:

- assertion: The assertion returned by the eIDAS node. It is stored as it is received, in an encoded format.
- birthdate: The user’s date of birth, retrieved from the access token returned by the eIDAS node.
- gender: The user’s gender, retrieved from the access token returned by the eIDAS node.
- person_identifier: The user’s national identification number, retrieved from the access token returned by the eIDAS node.

Furthermore, Keycloak may also accept a password to be set on a user account after it has been successfully created.

Login

Regarding the user login process, two methods that involve Keycloak may be employed. Keycloak natively offers an API endpoint through which users may be authenticated. This API may be used directly; However, the back-end also implements a relevant endpoint which communicates with Keycloak for this purpose, essentially acting as the middleman between the web UI and Keycloak. In both cases, Keycloak accepts the following fields as input:

- username: The user’s username, as given during registration.
- password: The user’s password, as given during registration.
- grant_type: "password" (constant) - It indicates that the client is requesting an access token using the user’s credentials directly.
- client_id: "re4dy-login" - It denotes the application name through which authentication takes place.

2.14.3 Output

Keycloak returns standard HTTP response codes for all requests it receives, based on the outcome or result of the request processing.

Registration

Keycloak returns the "201 Created" response code to indicate that the user creation process has successfully completed.

Login

When Keycloak has determined that the given credentials given are correct and correspond to a user stored in the related realm, it returns the following data:

- access_token: An access token corresponding to the user. It may be decoded to retrieve the information regarding the account, its Keycloak roles and access to Keycloak resources, as well as user information stored during the registration process (e.g. last name).
- expires_in: The remaining number of seconds until the access token expires.
- refresh_token: The refresh token which may be used to obtain a new access token.
- refresh_expires_in: The remaining number of seconds until the refresh token expires.
- token_type: Specification of the token type being used. Usually "Bearer".
- not-before-policy: The number of seconds before which the token is not valid.
- session_state: The session identifier or the token associated with the user's session. This identifier may be used to associate the token with the user's session in Keycloak.
- scope: The resources that the client making the request is authorized to access on behalf of the user through the access token.

2.14.4 Information Flow

Register

Registering begins through the RE4DY front-end application where the user is initially presented with an eIDAS login form to insert their eIDAS credentials. The credentials are sent from the front-end to the authentication back-end service and from there they are sent to the eIDAS node for authentication. If the credentials match with an existing eIDAS user, the eIDAS node returns an access token to the authentication back-end which is then forwarded to the front-end. Next, the user is presented with a registration form where they are able to create their RE4DY account. Their filled in RE4DY credentials are sent to the back-end along with the token returned by eIDAS. The token is decoded in order to extract information such as name, surname, gender and more. Using this information, the back-end uses the Keycloak Java API to create a new user and set the account password. Finally, upon success, the front-end and subsequently the user is informed that they may login with their RE4DY account.

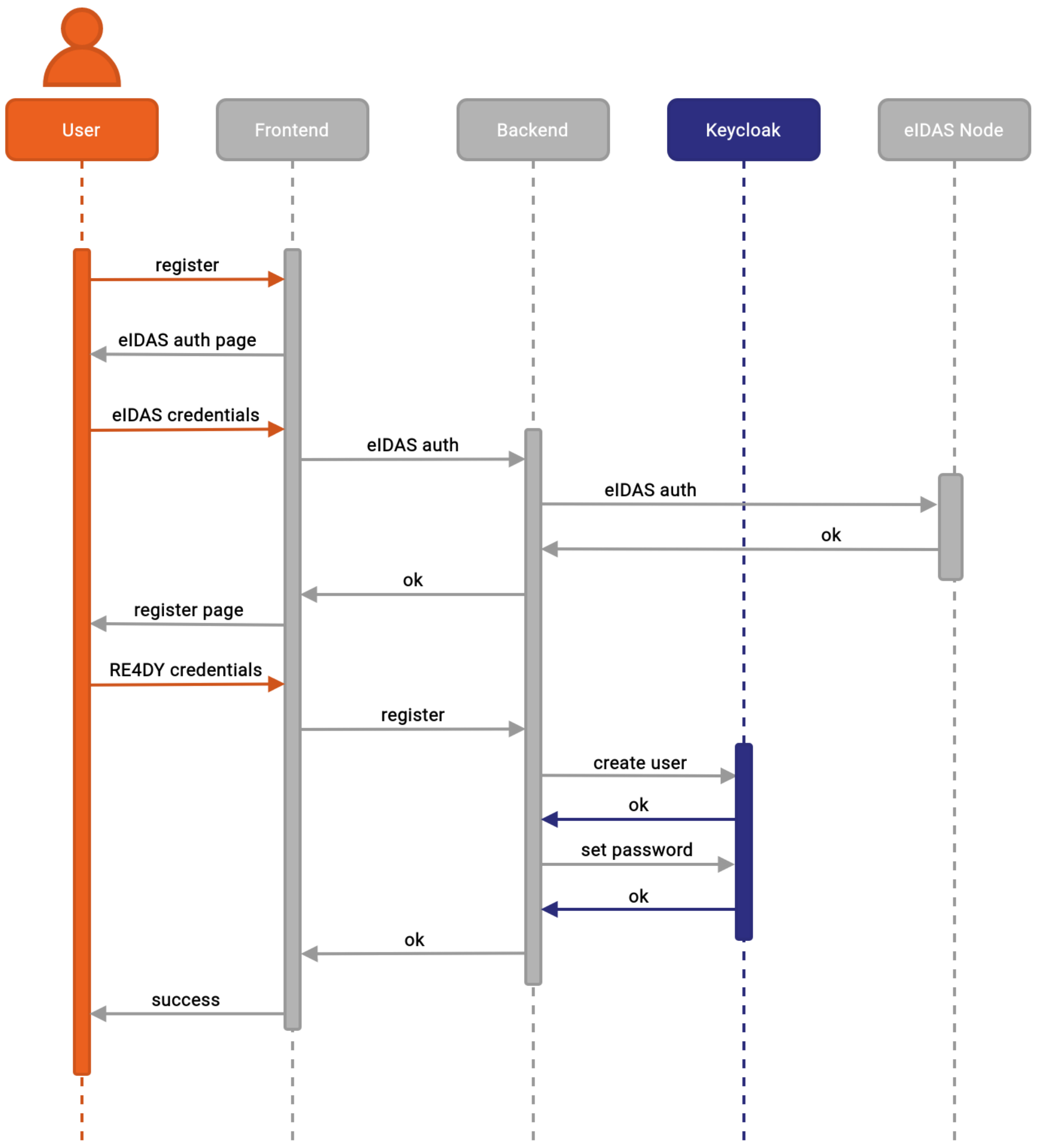


Figure 34: UML Diagram of the registration process and the communication of the front-end, authentication back-end, Keycloak and eIDAS services.

Login

The login procedure commences through the RE4DY front-end application where the user is presented with a login form where they are able to insert their username and password. Upon submitting the credentials, a request is sent to the authentication back-end, which communicates with Keycloak in order to determine whether the user exists and the credentials for the given username are valid. Upon successful authentication, an access token is returned to the back-end and propagated to the front-end, which allows the user to access the RE4DY application.

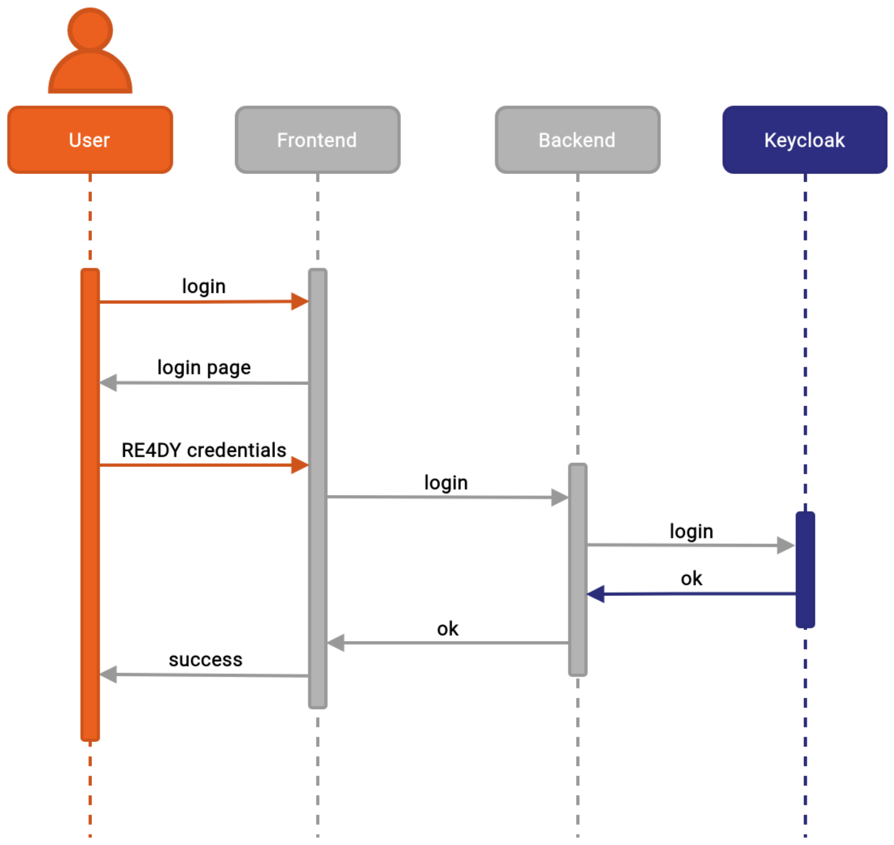


Figure 35: UML Diagram of the login process and the communication of the front-end, authentication back-end and Keycloak services.

2.14.5 Internal Architecture

Keycloak utilizes a PostgreSQL database for the storage of the RE4DY user base and its configuration settings. It offers the capability for an external application to communicate with it directly through its REST API, or to use a library, such as the Keycloak Java API. Our authentication back-end uses the Java API for the register and login operations. In turn, the RE4DY authentication page in the front-end of the platform uses the exposed API endpoints implemented by the authentication back-end.

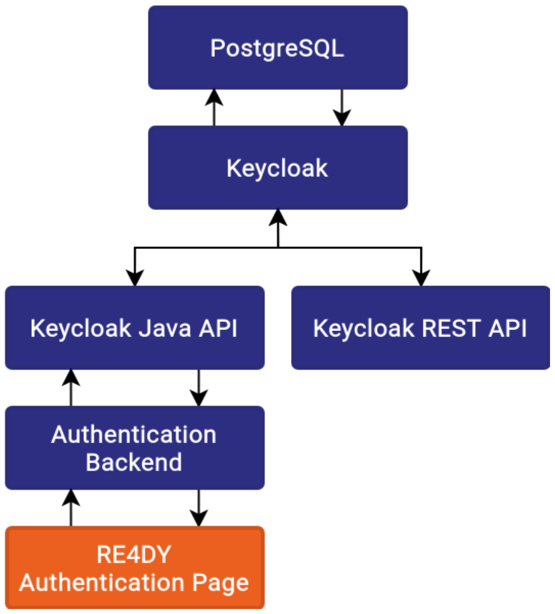


Figure 36: Internal architecture of the Keycloak component and related libraries and services

2.14.6 API

```
•      POST /auth/realms/re4dy/protocol/openid-connect/token

–      Input:
       {
    "username": "newuser",
    "password": "1234",
    "grant_type": "password",
    "client_id": "re4dy-login",
}
–      Output:
       {
    "access_token": "eyJhb...",
    "expires_in": 300,
    "refresh_expires_in": 1800,
    "refresh_token": "eyJhb...",
    "token_type": "Bearer",
    "not-before-policy": 0,
    "session_state": "6c033496-5551-4bff-898a-78156db13692",
    "scope": "profile email"
}

–      Description:
       This endpoint is provided natively by Keycloak and it may be
used for logging into any application in the re4dy realm. The
client_id field denotes the application the user is trying to
authenticate through.
```

The API documentation is also available in the Annex as a Swagger editor .yaml file.

2.14.7 Implementation Technology

Keycloak is open-source software which may be installed and executed on bare metal, on a virtual machine, as well as in a containerized manner using Docker and Kubernetes. We have opted for a container-based approach for all the services comprising the authentication back-end of the RE4DY platform, including Keycloak. Therefore, Docker is installed on a virtual machine running the CentOS 7 Linux operating system and the Compose plugin is used to orchestrate the deployment of the necessary services. We use the official Keycloak Docker image as a base for our Keycloak instance.

Keycloak supports the usage of integration APIs for a number of programming languages. In RE4DY, the integration of the other back-end services with the authentication and authorization functionality provided by Keycloak is performed using its Java API. Furthermore, a PostgreSQL database, deployed as a Docker container, is used by the Keycloak container as storage for the realm configuration and user data. Through the Admin Console, we have created the re4dy realm which encompasses the RE4DY user base and a client named re4dy-login, to be used for user registration and login through the RE4DY dashboard.

2.14.8 Comments

None.

2.15 Component Name: IPscreener

2.15.1 Description

With IPscreener, it is made possible for non-IP professionals to explore and understand the knowledge hidden in patents. The AI performs a semantic analysis from a text input and presents an instant dashboard of the related word-wide innovation landscape. The overview presents related trends, identifying similar documents revealing similar content and assisting the reading procedure by highlighting key terms and prioritising the most relevant text paragraphs to read first. The platform also enables online collaboration and reporting on results, which train the AI model to perform better on consecutive iterations and searches within a case. As a general overview, the IPscreener shows smarter ways to validate and improve ideas by avoiding re-inventing, speeding up research time, minimising dependences on competitor rights and assisting in better understanding related market and research trends on related activities.

2.15.2 API Structure and workflow

The workflow for accessing and retrieving data via the IPscreener APIs is shown below in Figure 37.

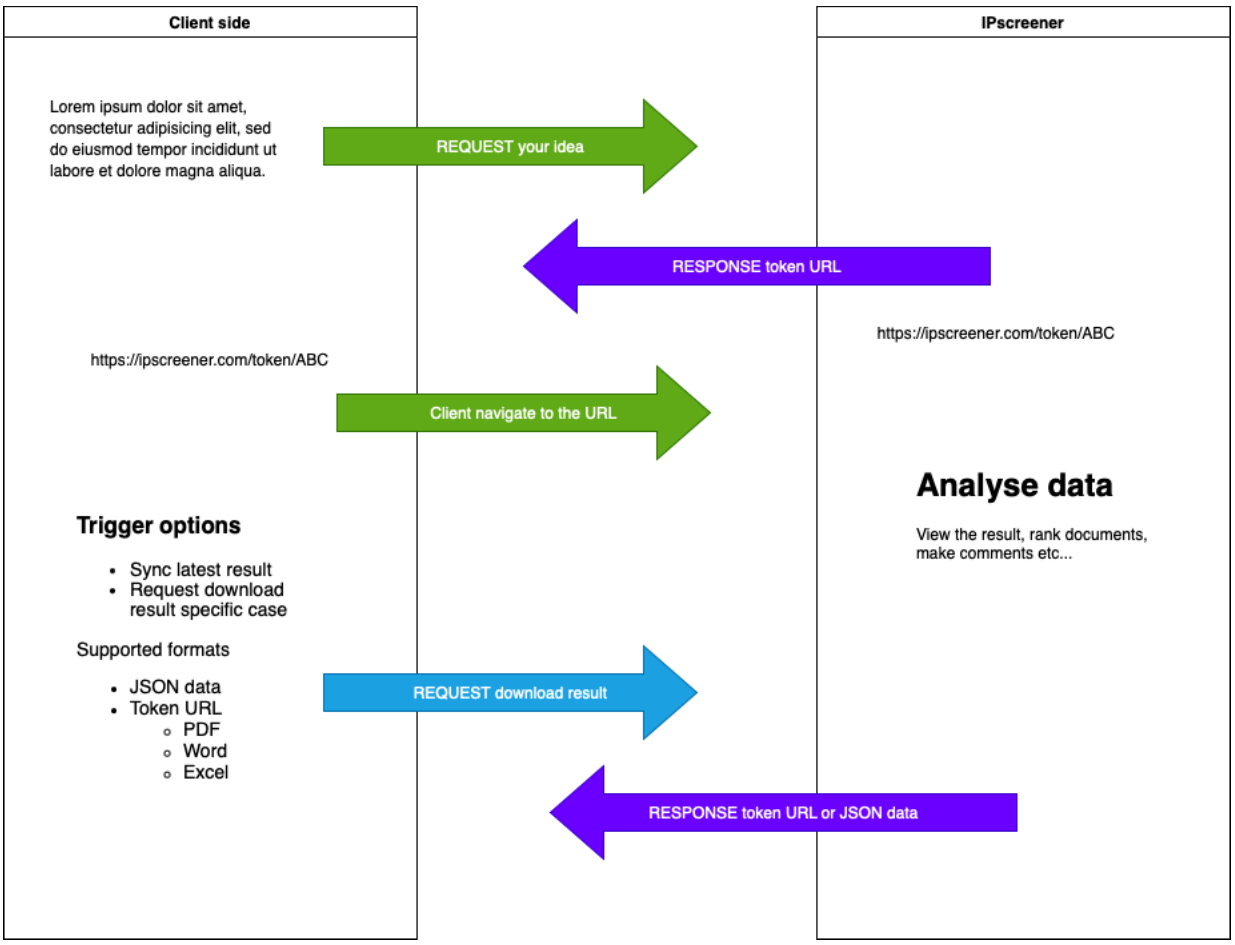

Client side
IPscreener
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
REQUEST your idea
RESPONSE token URL
https://ipscreener.com/token/ABC
https://ipscreener.com/token/ABC
Client navigate to the URL
Analyse data
View the result, rank documents, make comments etc...
Trigger options
Sync latest result
Request download result specific case
Supported formats
JSON data
Token URL
PDF
Word
Excel
REQUEST download result
RESPONSE token URL or JSON data

Figure 37: "IPscreener API workflow overview"

2.15.3 Input format

The input API supports input in text format or voice converted to text. Requests are made in JSON format.

2.15.4 Output format

The output API returns an output based on the input string in JSON format. The output is selectable via parameters to include from text data to objects including statistics, trend analysis, identified documents, comments and scores.

API documentation and access point

The latest official documentation on IPscreener APIs with access instructions, example queries and data fetching procedures is accessible via: https://docs.ipscreener.com/developer

IPscreener Implementation Technology

The IPscreener platform consists of a network of interconnected Docker containers. With the high-level system of Docker engines, the infrastructure may be hosted either in a cloud environment or in local setup on physical/virtual servers.

The backbone configuration is entirely built on 20+ API endpoints to provide real time services between internal system resources and functional modules as well as to to external service points. The storage nodes are hosting both warm and cold data instances, comprising a mix from client specific data and adaptions to the 100+ world-wide millions patent records being processed. The database use relates to the service function and processing function, and some major systems used are SQL-servers, Redis, Elasticsearch and Lucene/Solr.

The backend information retrieval system includes a multiplicity of pre-trained and on-the-fly adapted Machine Learning models, supported by Python libraries and languages frameworks such as SVM, BM25, Tensor flow, BART, BERT, GPT and other LLM models.

2.15.5 General Comments

The IPscreener platform is operating both as a WEB-service and as a REST access point. Data points as well as GUI/UX components are accessible, for end usage or for integration, via both access options.

2.16 Component Name: Incident Detection

2.16.1 Description

The Incident Detection component acquires monitoring awareness from all the events received from the monitored endpoints: authentication errors, software/malware installation, real time file monitoring (looking for unauthorized modifications), network attacks, privilege escalation, or others that can be configured. Incident detection monitoring also normalizes all the info received to prepare it for the next phase.

This component provides cybersecurity functionality to RE4DY endpoints and enables the detection of the events described above, both endpoints as component running.

2.16.2 Input

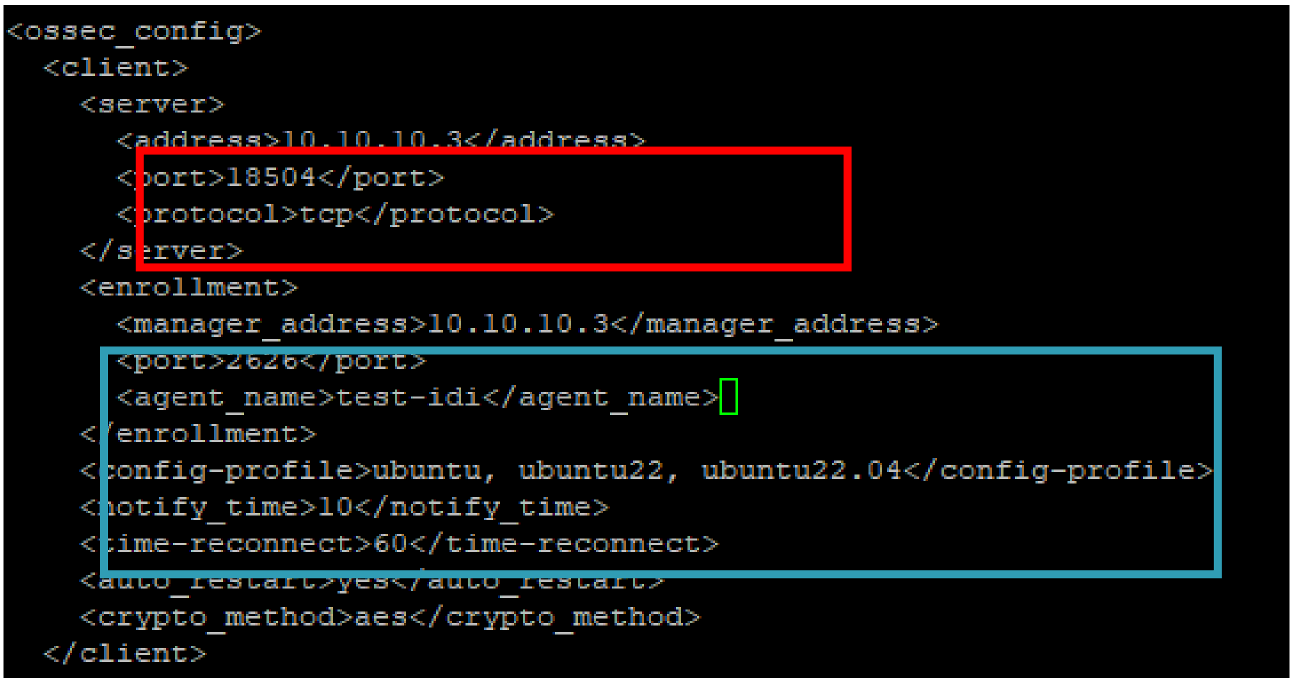
The Incident Detection communicates with the agents in the monitored endpoints to get the information about all the events registered, such as authentication errors, software/malware installation, real time file monitoring (looking for unauthorized modifications), network attacks, privilege escalation, or others that can be configured.

The incident detection component is integrated by the agent and the server:

1) Agent: Running in the endpoint, is in charge of collecting all the information from the different logs in the endpoints and other components and sends them to the server.

2) Incident detection: Collects all logs received from the agents in the network.

The agent running in each endpoint must be enrolled in the server and for that it is needed to change the configuration in the ossec.conf file, as showed in Figure 38, and restart the service.



```
<ossec_config>
  <client>
    <server>
      <address>10.10.10.3</address>
      <port>18504</port>
      <protocol>tcp</protocol>
    </server>
    <enrollment>
      <manager_address>10.10.10.3</manager_address>
      <port>2626</port>
      <agent_name>test-idi</agent_name>
    </enrollment>
    <config-profile>ubuntu, ubuntu22, ubuntu22.04</config-profile>
    <notify_time>10</notify_time>
    <time-reconnect>60</time-reconnect>
    <auto_restart>yes</auto_restart>
    <crypto_method>aes</crypto_method>
  </client>
```

Figure 38: ossec.conf file configuration example

2.16.3 Output

The incident detection component communicates with the incident response component and sends all the normalized alerts detected to be treated. For each alert or incident a JSON file will be generated as the following one:

```
{
  "_index": "wazuh-alerts-4.x-2023.02.27",
  "_type": "_doc",
  "_id": "9q8DlIYBeKdNQkVPVHDn",
  "_version": 1,
  "_score": None,
  "_source": {
    "input": {
      "type": "log"
    },
    "agent": {
      "ip": "192.168.15.112",
      "name": "RE4DY agent 1",
      "id": "005"
    },
    "manager": {
      "name": "wazuh-manager"
    },
    "data": {
      "srcip": "192.168.15.177",
      "in_iface": "enp0s3",
      "src_ip": "192.168.15.186",
      "src_port": "6443",
      "event_type": "alert",
      "alert": {
        "severity": "3",
        "signature_id": "2210046",
        "rev": "2",
        "gid": "1",
        "signature": "SURICATA STREAM SHUTDOWN RST invalid ack",
        "action": "allowed",
        "category": "Generic Protocol Command Decode"
      },
      "tls":{
          "version":"TLS 1.3",
          "ja3":{
              "hash":"89be98bbd4f065fe510fca4893cf8d9b",
              "string":"771,49199-49200-49195-49196-52392-52393-
49171-49161-49172-49162-156-157-47-53-49170-10-4865-4867-4866,5-10-
11-13-65281-18-43-51,29-23-24-25,0"
          },
          "ja3s":{
              "hash":"f4febc55ea12b31ae17cfb7e614afda8",
              "string":"771,4865,43-51"
          }

      },
      "app_proto":"tls",
      "flow_id": "1331957630929130.000000",
      "dest_ip": "192.168.15.177",
      "proto": "TCP",
      "dest_port": "48580",
      "flow": {
        "pkts_toserver": "18",
        "start": "2023-02-27T01:54:58.624874+0000",
        "bytes_toclient": "5318",
        "bytes_toserver": "2212",
        "pkts_toclient": "17"
      },
      "timestamp": "2023-02-27T01:59:53.111013+0000"
    },
    "rule": {
      "firedtimes": 1,
      "mail": True,
      "level": 12,
      "description": "DoS attack has been detected.",
      "groups": [
        "custom_active_response_rules"
      ],
      "mitre": {
        "id": [
          "T1498"
        ],
        "tactic": [
          "Impact"
        ],
        "technique": [
          "Network Denial of Service"
        ]
      },
      "id": "100200"
    },
    "location": "/var/log/suricata/eve.json",
    "decoder": {
      "name": "json"
    },
    "id": "1677463193.114738",
    "full_log": "{\"timestamp\":\"2023-02-
27T01:59:53.111013+0000\",\"flow_id\":1.33195763092913e+15,\"in_ifac
e\":\"enp0s3\",\"event_type\":\"alert\",\"src_ip\":\"192.168.15.186\
",\"src_port\":6443,\"dest_ip\":\"192.168.15.177\",\"dest_port\":485
80,\"proto\":\"TCP\",\"alert\":{\"action\":\"allowed\",\"gid\":1,\"s
ignature_id\":2210046,\"rev\":2,\"signature\":\"SURICATA STREAM
SHUTDOWN RST invalid ack\",\"category\":\"Generic Protocol Command
Decode\",\"severity\":3},\"tls\":{\"version\":\"TLS
1.3\",\"ja3\":{\"hash\":\"89be98bbd4f065fe510fca4893cf8d9b\",\"strin
g\":\"771,49199-49200-49195-49196-52392-52393-49171-49161-49172-
49162-156-157-47-53-49170-10-4865-4867-4866,5-10-11-13-65281-18-43-
51,29-23-24-
25,0\"},\"ja3s\":{\"hash\":\"f4febc55ea12b31ae17cfb7e614afda8\",\"st
ring\":\"771,4865,43-51\"}},\"app_proto\":\"tls\",
\"flow\":{\"pkts_toserver\":18,\"pkts_toclient\":17,\"bytes_toserver
\":2212,\"bytes_toclient\":5318,\"start\":\"2023-02-
27T01:54:58.624874+0000\"}}",
    "timestamp": "2023-02-27T01:59:53.809+0000"
  },
  "fields": {
    "data.timestamp": [
      "2023-02-27T17:54:49.468Z"
    ],
    "timestamp": [
      "2023-02-27T17:54:49.909Z"
    ]
  },
    "highlight": {
      "rule.id": [
        "@kibana-highlighted-field@100201@/kibana-highlighted-field@"
      ]
    },
    "sort": [
      1677520489908
    ]
  }
```

2.16.4 Information Flow

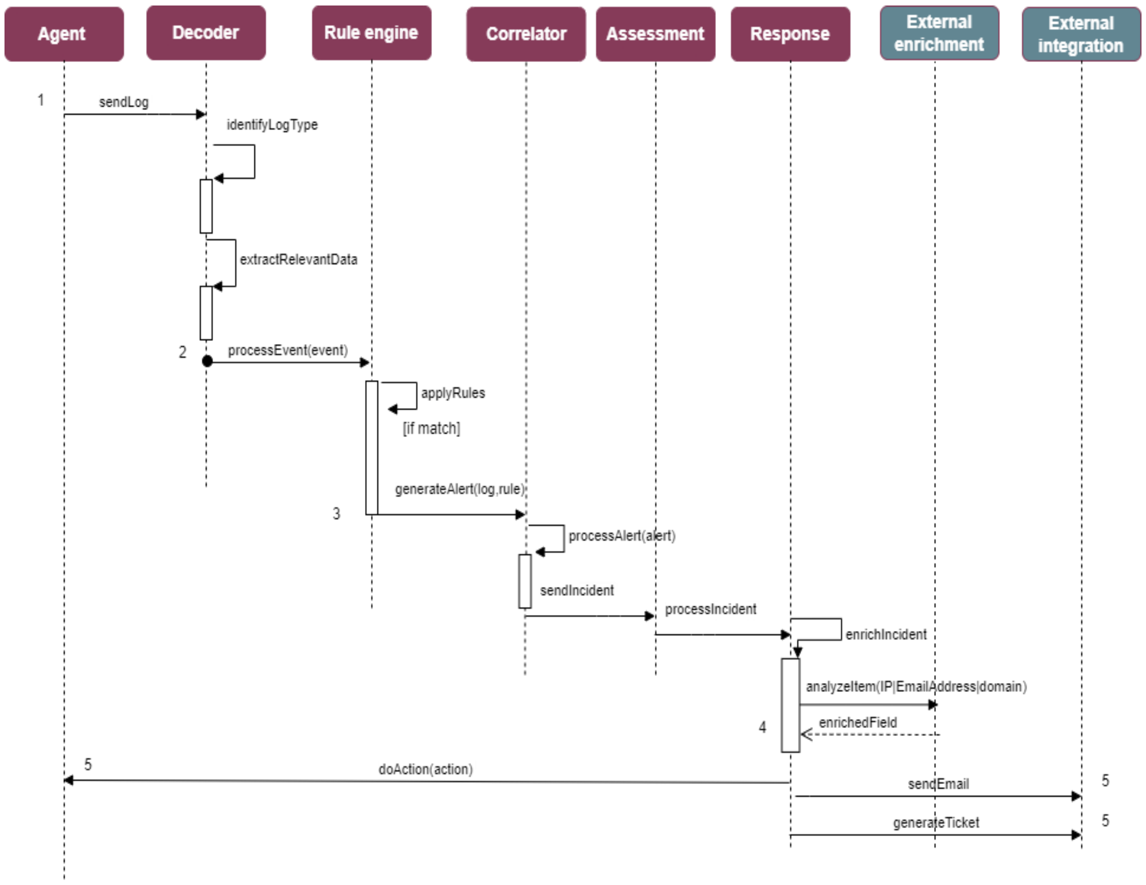


Figure 39: Incident detection and response information flow diagram

- Incident Detection will receive logs and information from the agents deployed.
- Incident Detection will decode the log, identify the type of log and extract some useful fields.
- Incident Detection will have a ruleset to be applied to the received logs.
- Incident Detection will apply the active rules to the received log, and if there is a match, it will generate an alert.

- Incident Response will normalize the alert event and correlate until determine if it is only a simple alert or a real incident.
- Incident Response will enrich the incident with useful information, to facilitate the assignment of the risk level of the incident and the response actions to be done.
- Incident Response can do predefined actions for incident mitigation depending on the incident, such as communicate with the agent so that it performs an action, send an email or send the incident to a ticketing system.

Incident Detection will update information on a GUI so that the admin user can see the status of the agents and the alert/incident information.

2.16.5 Internal Architecture

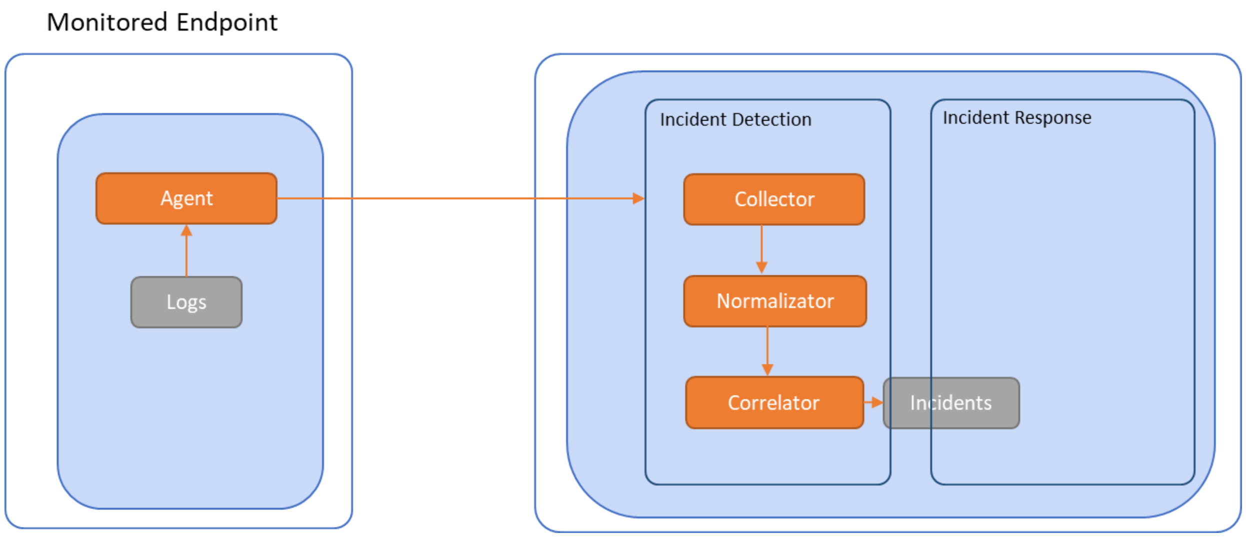


Figure 40: Incident detection component architecture

Figure 40 shows how the different blocks are connected in the component. The left side shows the monitored endpoint with an agent running, that collects all the information from the different logs.

The right side shows the server side, where the incident detection and incident response are running. The Agent sends the info to the incident detection component and after passing through all the steps send the incidents that must have a response to the incident response component.

2.16.6 API

METHOD	DESCRIPTION	ENDPOINT
PUT	Run an Active Response command on all agents or a list of them	{SIEM}/active-response
PUT	Restart all agents or a list of them	{SIEM}/agents/restart
PUT	Restart the specified agent	{SIEM}/agents/{agent_id}/restart
POST	Add an agent specifying its name, ID and IP. If an agent with the same ID already exists, replace it using 'force' parameter	{SIEM}/agents/insert
POST	Add a new agent with basic info	{SIEM}/agents
DELETE	Delete all agents or a list of them based on optional criteria	{SIEM}/agents
GET	Obtain a list with information of the available agents	{SIEM}/agents
PUT	Restart the manager	{SIEM}/manager/restart
GET	Return statistical information for the current or specified date	{SIEM}/manager/stats
PUT	Replace configuration with the data contained in the API request	{SIEM}/manager/configuration
GET	Return enabler configuration used	{SIEM}/manager/configuration

GET	Basic information such as version, compilation date, installation path	{SIEM}/manager/info
GET	Return the status of the monitoring server	{SIEM}/manager/status

2.16.7 Implementation Technology

Wazuh is open-source software which can be installed and executed on bare metal, on a virtual machine, as well as in containerized manner using Docker or Kubernetes. We have chosen the container-based approach for all services related to incident detection in the RE4DY platform. Therefore, Docker is installed on a virtual machine running Ubuntu 22.04 Linux OS and the compose plugin is used to orchestrate the deployment of the necessary services. We use the official Wazuh Docker image as base for our Incident Detection instance.

2.16.8 Comments

The Incident Detection component will be integrated with the Incident Response component.

2.17 Component Name: Incident Response

2.17.1 Description

The Incident Response component receives all the alerts from the Incident Detection component and creates a security case to perform a treatment and mitigation process. Information received can be enriched from other sources related to the original alert. Once the information is ready, with the incident response tool it is possible to orchestrate different kinds of actions, such as automated responses like introducing a condition such as a firewall drop rule in the monitored endpoint. Other orchestrated actions are possible such us notifications to third party applications like ticketing systems, email or instant messaging applications.

2.17.2 Input

The incident response communicates with incident detection and can communicate with other information sources to enrich the cases and offer a better understanding of the response given to the system. The alert or incident received from the Incident Detection component is in JSON format:

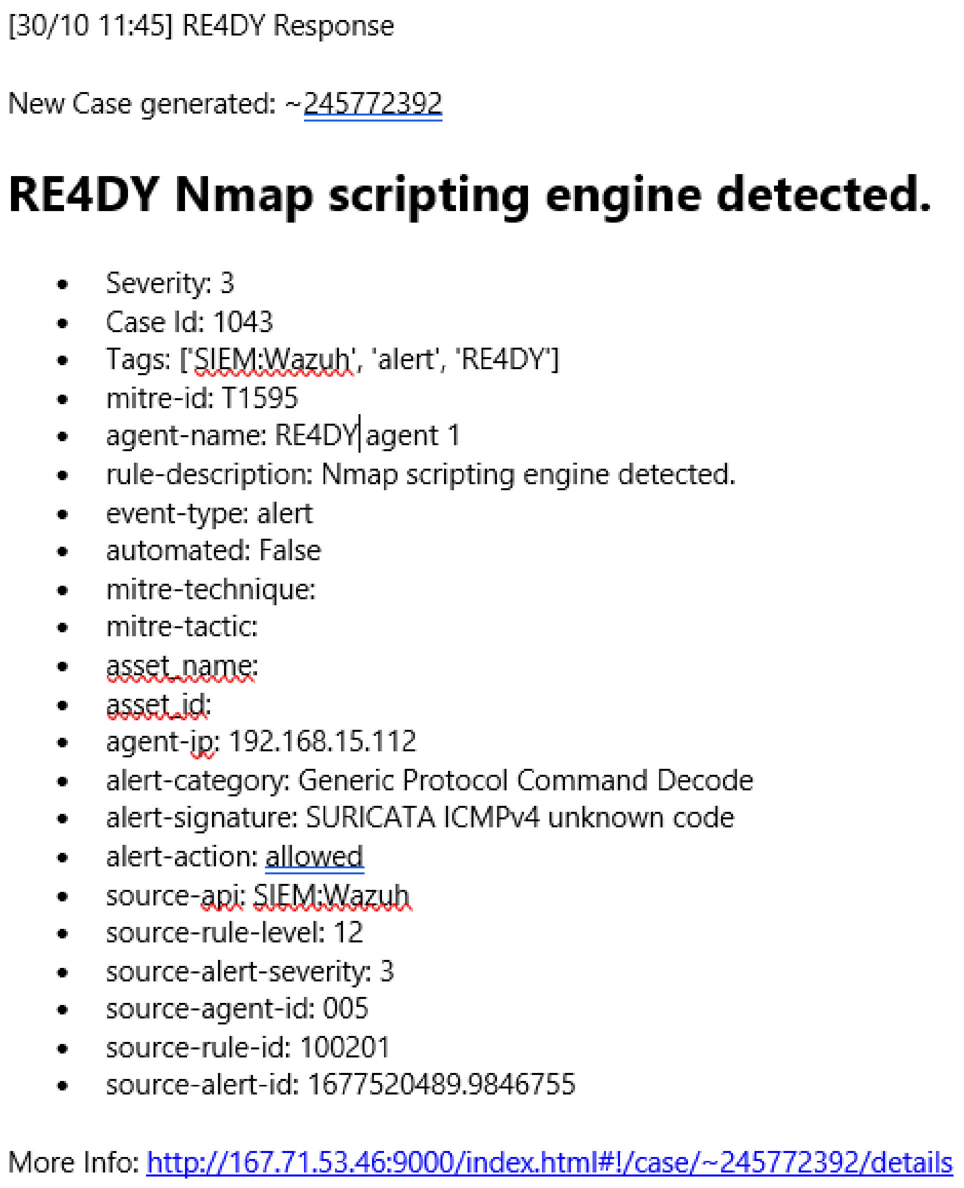
```
{
  "_index": "wazuh-alerts-4.x-2023.02.27",
  "_type": "_doc",
  "_id": "9q8DlIYBeKdNQkVPVHDn",
  "_version": 1,
  "_score": None,
  "_source": {
    "input": {
      "type": "log"
    },
    "agent": {
      "ip": "192.168.15.112",
      "name": "RE4DY agent 1",
      "id": "005"
    },
    "manager": {
      "name": "wazuh-manager"
    },
    "data": {
      "srcip": "192.168.15.177",
      "in_iface": "enp0s3",
      "src_ip": "192.168.15.186",
      "src_port": "6443",
      "event_type": "alert",
      "alert": {
        "severity": "3",
        "signature_id": "2210046",
        "rev": "2",
        "gid": "1",
        "signature": "SURICATA STREAM SHUTDOWN RST invalid ack",
        "action": "allowed",
        "category": "Generic Protocol Command Decode"
      },
      "tls":{
          "version":"TLS 1.3",
          "ja3":{
              "hash":"89be98bbd4f065fe510fca4893cf8d9b",
              "string":"771,49199-49200-49195-49196-52392-52393-
49171-49161-49172-49162-156-157-47-53-49170-10-4865-4867-4866,5-10-
11-13-65281-18-43-51,29-23-24-25,0"
          },
          "ja3s":{
              "hash":"f4febc55ea12b31ae17cfb7e614afda8",
              "string":"771,4865,43-51"
          }

      },
      "app_proto":"tls",
      "flow_id": "1331957630929130.000000",
      "dest_ip": "192.168.15.177",
      "proto": "TCP",
      "dest_port": "48580",
      "flow": {
        "pkts_toserver": "18",
        "start": "2023-02-27T01:54:58.624874+0000",
        "bytes_toclient": "5318",
        "bytes_toserver": "2212",
          "pkts_toclient": "17"
        },
        "timestamp": "2023-02-27T01:59:53.111013+0000"
      },
      "rule": {
        "firedtimes": 1,
        "mail": True,
        "level": 12,
        "description": "DoS attack has been detected.",
        "groups": [
          "custom_active_response_rules"
        ],
        "mitre": {
          "id": [
            "T1498"
          ],
          "tactic": [
            "Impact"
          ],
          "technique": [
            "Network Denial of Service"
          ]
        },
        "id": "100200"
      },
      "location": "/var/log/suricata/eve.json",
      "decoder": {
        "name": "json"
      },
      "id": "1677463193.114738",
      "full_log": "{\"timestamp\":\"2023-02-
27T01:59:53.111013+0000\",\"flow_id\":1.33195763092913e+15,\"in_ifac
e\":\"enp0s3\",\"event_type\":\"alert\",\"src_ip\":\"192.168.15.186\
",\"src_port\":6443,\"dest_ip\":\"192.168.15.177\",\"dest_port\":485
80,\"proto\":\"TCP\",\"alert\":{\"action\":\"allowed\",\"gid\":1,\"s
ignature_id\":2210046,\"rev\":2,\"signature\":\"SURICATA STREAM
SHUTDOWN RST invalid ack\",\"category\":\"Generic Protocol Command
Decode\",\"severity\":3},\"tls\":{\"version\":\"TLS
1.3\",\"ja3\":{\"hash\":\"89be98bbd4f065fe510fca4893cf8d9b\",\"strin
g\":\"771,49199-49200-49195-49196-52392-52393-49171-49161-49172-
49162-156-157-47-53-49170-10-4865-4867-4866,5-10-11-13-65281-18-43-
51,29-23-24-
25,0\"},\"ja3s\":{\"hash\":\"f4febc55ea12b31ae17cfb7e614afda8\",\"st
ring\":\"771,4865,43-51\"}},\"app_proto\":\"tls\",
\"flow\":{\"pkts_toserver\":18,\"pkts_toclient\":17,\"bytes_toserver
\":2212,\"bytes_toclient\":5318,\"start\":\"2023-02-
27T01:54:58.624874+0000\"}}",
      "timestamp": "2023-02-27T01:59:53.809+0000"
    },
    "fields": {
      "data.timestamp": [
        "2023-02-27T17:54:49.468Z"
      ],
      "timestamp": [
        "2023-02-27T17:54:49.909Z"
      ]
    },
    "highlight": {
      "rule.id": [
        "@kibana-highlighted-field@100201@/kibana-highlighted-field@"
      ]
```

```
  },
  "sort": [
    1677520489908
  ]
}
```

2.17.3 Output

The Incident Response provides outputs to different systems, such as a firewall (firewall drop rule in a monitored endpoint), ticketing systems, email or instant messaging applications. The following figure shows an example of a published response message in MS Teams:



[30/10 11:45] RE4DY Response

New Case generated: ~245772392

RE4DY Nmap scripting engine detected.

- Severity: 3
- Case Id: 1043
- Tags: ['SIEM:Wazuh', 'alert', 'RE4DY']
- mitre-id: T1595
- agent-name: RE4DY agent 1
- rule-description: Nmap scripting engine detected.
- event-type: alert
- automated: False
- mitre-technique:
- mitre-tactic:
- asset_name:
- asset_id:
- agent-ip: 192.168.15.112
- alert-category: Generic Protocol Command Decode
- alert-signature: SURICATA ICMPv4 unknown code
- alert-action: allowed
- source-api: SIEM:Wazuh
- source-rule-level: 12
- source-alert-severity: 3
- source-agent-id: 005
- source-rule-id: 100201
- source-alert-id: 1677520489.9846755

More Info: http://167.71.53.46:9000/index.html#!/case/~245772392/details

Figure 41: Incident response: Teams publication respnse example

2.17.4 Information Flow

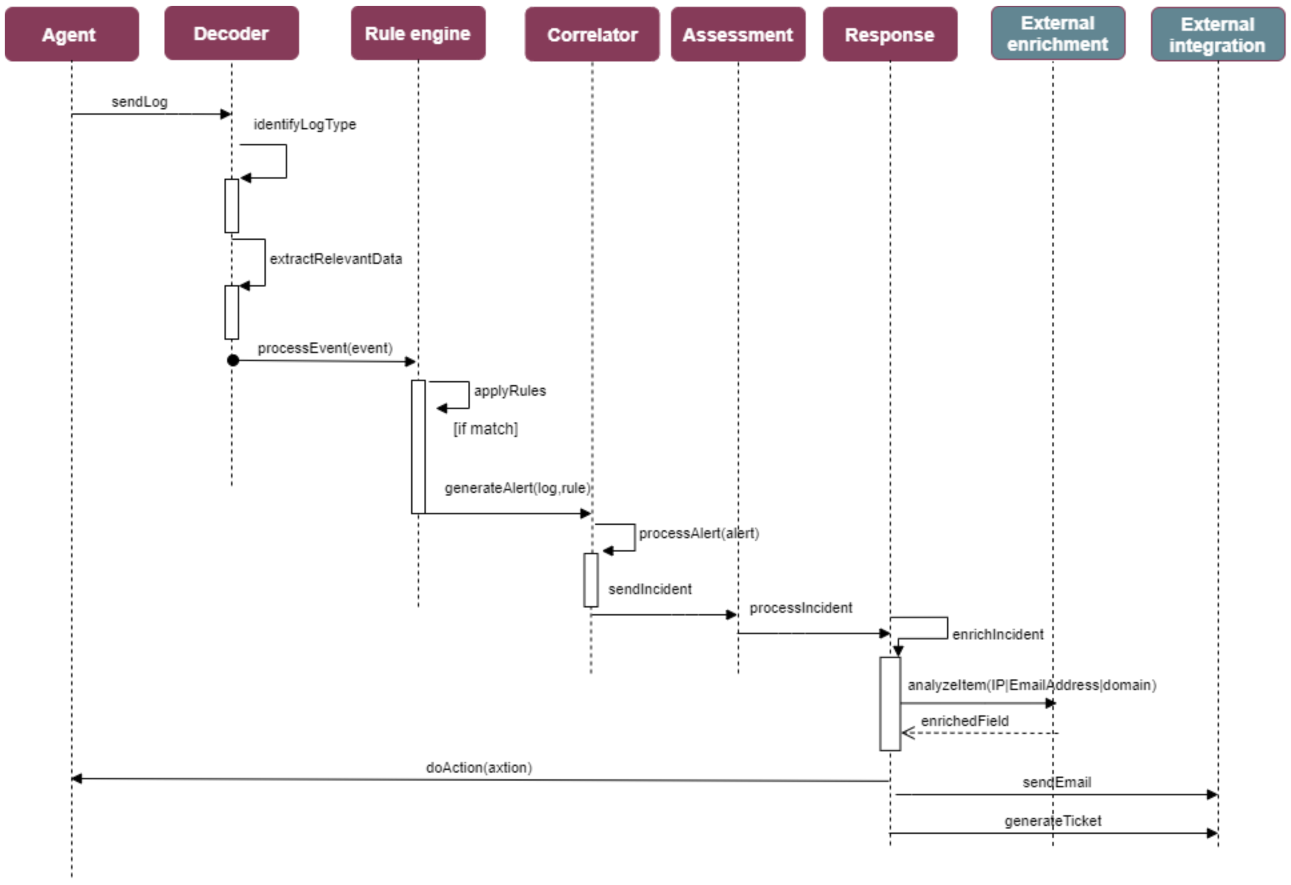


Figure 42: Incident detection and response information flow diagram

- Incident Detection will receive logs and information from the agents deployed.
- Incident Detection will decode the log, identify the type of log and extract some useful fields.
- Incident Detection will have a ruleset to be applied to the received logs.
- Incident Detection will apply the active rules to the received log, and if there is a match, it will generate an alert.
- Incident Response will normalize the alert event and correlate until determine if it is only a simple alert or a real incident.
- Incident Response will enrich the incident with useful information, to facilitate the assignment of the risk level of the incident and the response actions to be taken.
- Incident Response can take predefined actions for incident mitigation depending on the incident, such as communicate with the agent so that it performs an action, send an email or send the incident to a ticketing system.

Incident Detection will update information on a GUI so that the admin user can see the status of the agents and the alert/incident information.

2.17.5 Internal Architecture

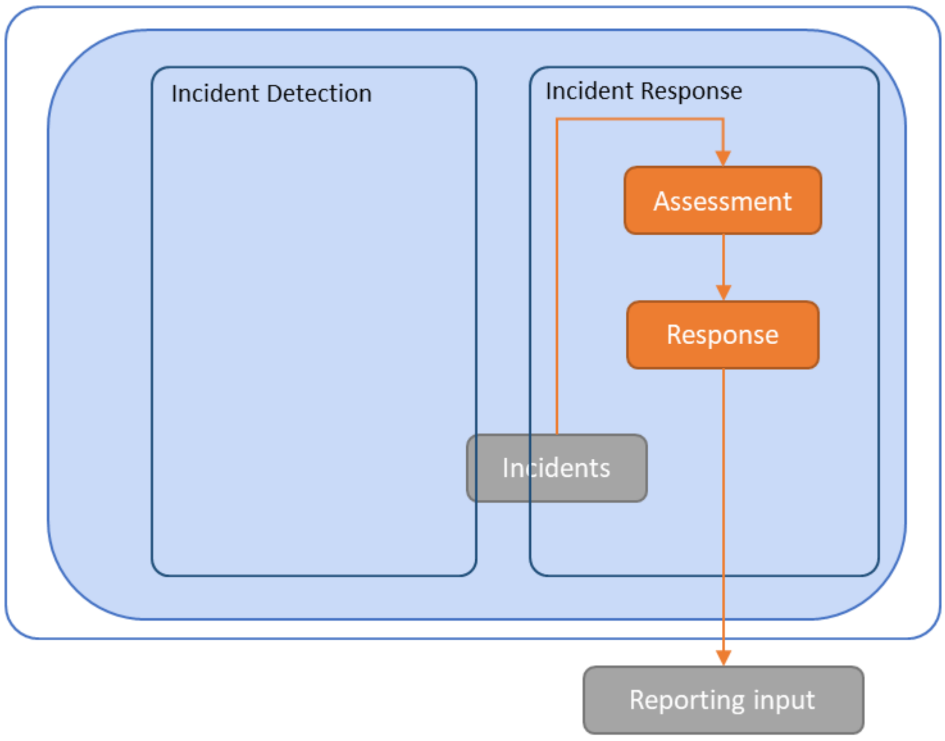


Figure 43: Incident response component architecture

Figure 43 shows how the different blocks are connected in the component. The left side shows the incident detection component, which sends the generated incident reports to the incident response component.

The right side shows the incident response component which collects the incident reports generated in the previous step and assesses them to finally generate a response to the incident.

2.17.6 API

METHOD	DESCRIPTION	ENDPOINT
POST	Authenticate an user and get session cookie	{SOAR}/api/v1/login
POST	Create an organisation	{SOAR}/api/v1/organisation

GET	List all user profiles	{SOAR}/api/v0/profile
POST	Create a new profile	{SOAR}/api/v0/profile
GET	Get information of the given profile	{SOAR}/api/v0/profile/{profile}
PATCH	Update profile	{SOAR}/api/v0/profile/{profile}
DELETE	Remove the profile	{SOAR}/api/v0/profile/{profile}
POST	Create a new user	{SOAR}/api/v1/user
GET	Show information of the current user	{SOAR}/api/v1/user/current
GET	Show information of the given user	{SOAR}/api/v1/user/{user}
PATCH	Update information of the given user	{SOAR}/api/v1/user/{user}
DELETE	Remove an user	{SOAR}/api/v1/user/{user}/force
POST	Set the user password	{SOAR}/api/v1/user/{user}/password/set
POST	Change the user password	{SOAR}/api/v1/user/{user}/password/change
GET	Get the user API key	{SOAR}/api/v1/user/{user}/key
DELETE	Remove the user API key	{SOAR}/api/v1/user/{user}/key
POST	Renew the user API key	{SOAR}/api/v1/user/{user}/key/renew

2.17.7 Implementation Technology

TheHive is open source software which can be installed and executed on bare metal, on a virtual machine, as well as in a containerized manner using Docker or Kubernetes. We have chosen the container-based approach for all services related to incident response in the RE4DY platform. Therefore, Docker is installed on a virtual machine running Ubuntu 22.04

Linux OS and the compose plugin is used to orchestrate the deployment of the necessary services. We use the official TheHive Docker image as the base for our Incident Response instance.

2.17.8 Comments

This component will be integrated with the Incident Detection component.

2.18 Component Name: Create MyVirtual Machine

2.18.1 Description

Create MyVirtual Machine is a virtual CNC system that simulates a SINUMERIK ONE on a user PC. The hardware components of the control are modeled as software components and represent a complete image of a real CNC. With Create MyVirtual Machine, a user can develop and test the next control generation in the development phase of a CNC machine, or NCK, PLC and HMI software without requiring any hardware. Parts of the machine commissioning are preconfigured on the virtual model. It is possible to shorten the commissioning time of the real machine by configuring the system using the virtual model. Furthermore, the created machine projects can be made available by Create MyVirtual Machine for processing in Run MyVirtual Machine for work preparation. Create MyVirtual Machine includes Create MyVirtual Machine /Operate, Create MyVirtual Machine /Open and Create MyVirtual Machine /3D.

Create MyVirtual Machine /Operate contains the TIA Portal data file for configurating the PLC, HMI, and the Numeric Controller. All these components represent the physical components and provide all functionalities from the real products. With Create MyVirtual Machine /Open the user can connect further applications to Create MyVirtual Machine. Create MyVirtual Machine /3D demonstrates a visual simulation of the machine and workpiece where the user can test visually how the system will run in the operating system.

As part of the RE4DY project, Create MyVirtual Machine will be used in the GF and Fraisa pilot. With Create MyVirtual Machine, the physical machine (Mill P 800 U S) from GF will be virtually created. This includes the 3D simulation of the real machine, the implementation of the control code and the virtual numerical code. This setup is necessary to build the digital twin of the real machine.

2.18.2 Input

Create MyVirtual Machine /Operate

The operation of a Create MyVirtual Machine corresponds to that of a real control equipped with a SINUMERIK Operate user interface and machine control panel. Therefore, Create MyVirtual Machine simulates the peripherals through direct reading and writing in the PLC I/O image. Parts of the PLC I/O image are also used for internal communications. Create MyVirtual Machine /Operate allows the user to control the virtual machine like a real physical machine. To do this, the virtual machine must first be created using STL data and a TIA project data file.

Create MyVirtual Machine /Open

The Open Interface with Create MyVirtual Machine /Open allows external applications to control the Create MyVirtual Machine system and to communicate at runtime. The Create MyVirtual Machine application can be started, operated, and stopped by other applications. To do this, the virtual machine must first be created using STL data and a TIA project data file.

Create MyVirtual Machine /3D

Create MyVirtual Machine visualizes the machining process and machine movements by means of a 3D simulation. A user can simulate the processing of NC programs in AUTOMATIC mode, for example, or manual traversing movements and tool changes in JOG mode. To do this, the virtual machine must first be created using STL data and a TIA project data file.

2.18.3 Output

Create MyVirtual Machine allows the user to develop and test the control in the development phase of a CNC machine or NCK, PLC and HMI software without the need for hardware. Create MyVirtual Machine simulates the virtual machine and tests the functionality of the hardware. This is used to analyze and optimize the engineering. On the one hand, while the virtual machine is running in Create MyVirtual Machine, the user gets transparency about the engineering, which he can use for optimization. On the other hand, data from the virtual controllers is available, which can be made available for other applications.

2.18.4 Information Flow

Installation and configuration

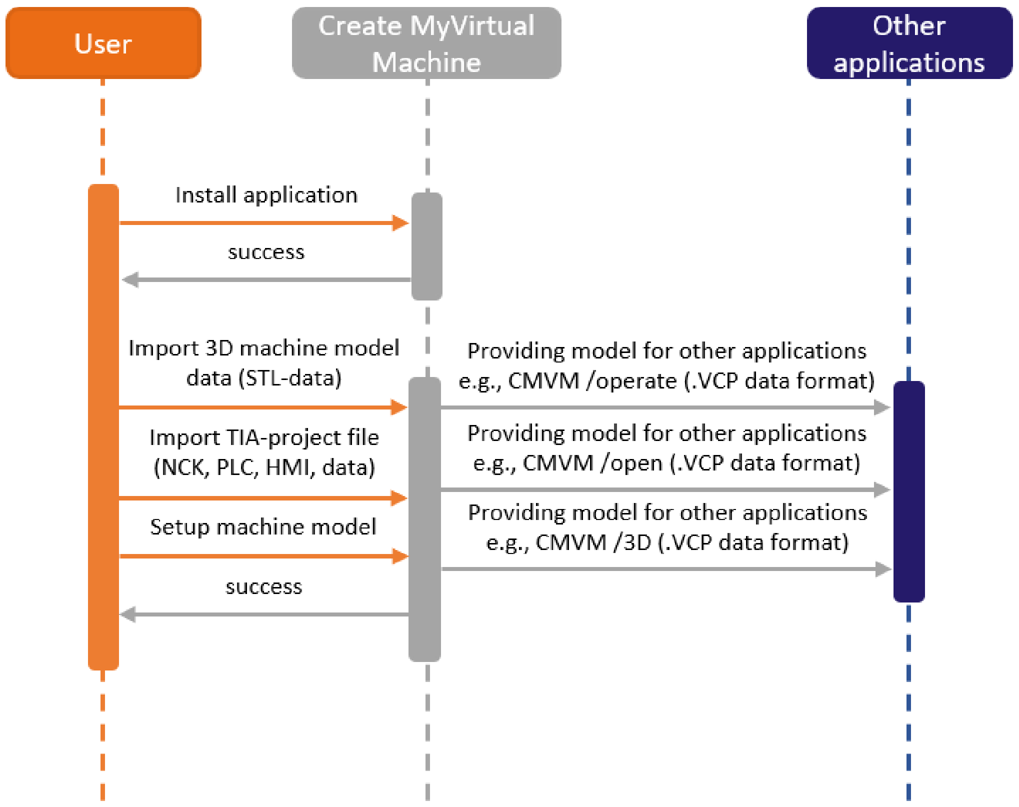

User
Create MyVirtual Machine
Other applications
Install application
success
Import 3D machine model data (STL-data)
Import TIA-project file (NCK, PLC, HMI, data)
Setup machine model
success
Providing model for other applications e.g., CMVM /operate (.VCP data format)
Providing model for other applications e.g., CMVM /open (.VCP data format)
Providing model for other applications e.g., CMVM /3D (.VCP data format)

Figure 44: Installation and configuration

The installation of the Create MyVirtual Machine application is necessary to create and build the virtual machine of an existing real machine. To create the virtual machine, input files such as STL-data for a 3D simulation are required. Also, a TIA project data file is needed, which contains information like numerical control code or control code. After successful creation of the virtual machine, the virtual machine can be exported as a .VCP file to be used in Run MyVirtual Machine.

Virtual operate

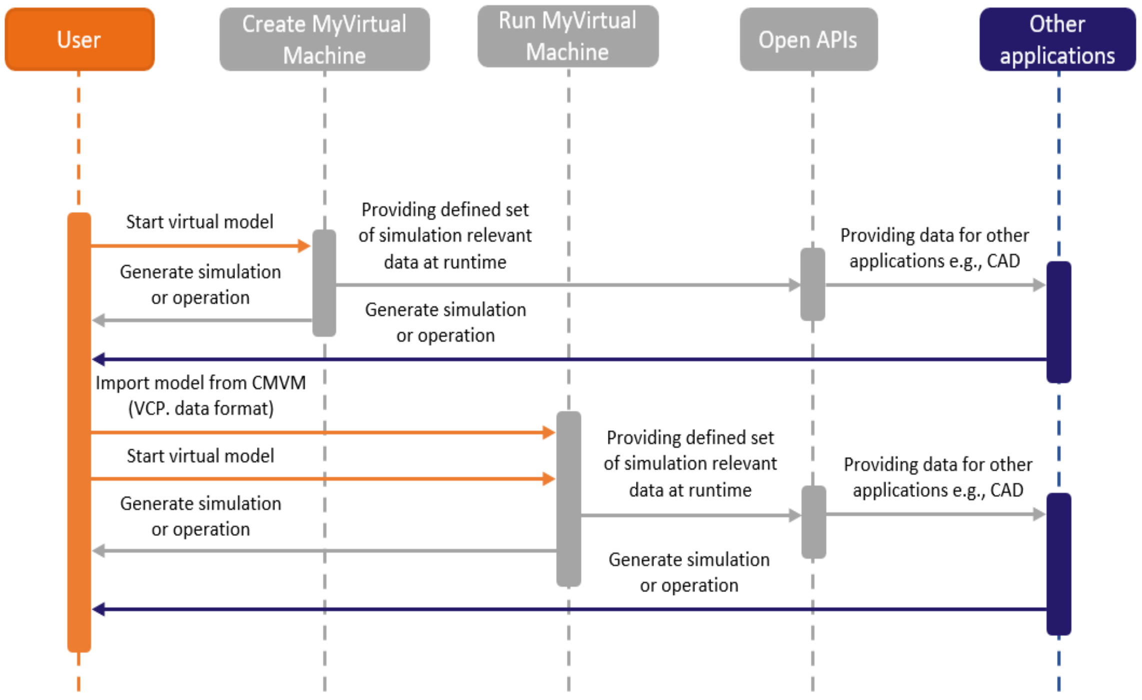

User
Create MyVirtual Machine
Run MyVirtual Machine
Open APIs
Other applications
Start virtual model
Generate simulation or operation
Providing defined set of simulation relevant data at runtime
Generate simulation or operation
Providing data for other applications e.g., CAD
Import model from CMVM (VCP. data format)
Start virtual model
Generate simulation or operation
Providing defined set of simulation relevant data at runtime
Generate simulation or operation
Providing data for other applications e.g., CAD

Figure 45: Virtual operate

The virtual machine can be run within Create MyVirtual Machine. A user can run, simulate, configure, and analyze the virtual machine within the application. To do this, the user must import or run the .VCP data file from Create MyVirtual Machine. Simulation-related data is generated during runtime. This data can be used for analysis and optimization and can also be made available for other applications for analysis and optimization.

2.18.5 Internal Architecture

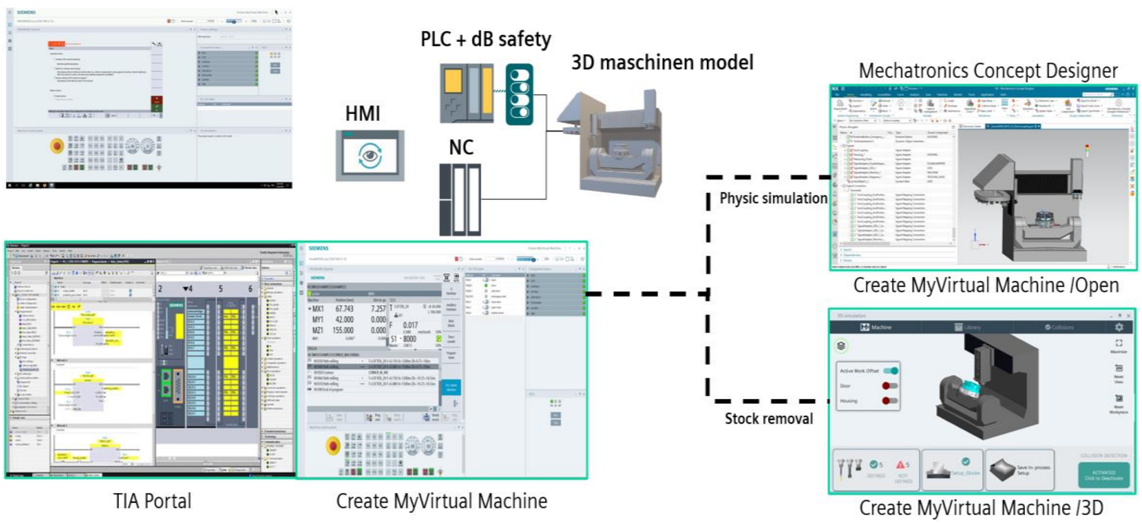


Figure 46: Internal Architecture

To create a virtual machine the application, Create MyVirtual Machine must be installed. Out of the TIA Portal a .VCP file can be exported, which contains the configurating of the controller code, Human Machine Interface, and the Numeric Controller. All these components represent the physical components and provide all functionalities from the real products.

Create MyVirtual Machine/3D is a license that allows the user to visualize the movement of the machine. Create MyVirtual Machine /Open is a license that allows users to connect further applications to Create MyVirtual Machine.

2.18.6 API

The open interface allows an external application to control the SINUMERIK ONE system and to communicate with the SINUMERIK ONE Application during runtime. The task scope of the open interface addresses various use cases (selection):

- Remote control of the SINUMERIK ONE application for example, in automatic inspection and test units
- Cyclic exchange of runtime information, for example, to connect external simulation systems (I/Os, virtual machine applications etc.)

Create MyVirtual Machine or Run MyVirtual Machine provides a defined set of simulation relevant data at runtime.

Further information: https://support.industry.siemens.com/cs/attachments/109817568/AppNote_Run_MyVirtual_Machine_Open_en.pdf

2.18.7 Implementation Technology

Licenses:

Create MyVirtual Machine /Operate

Create MyVirtual Machine /Open

Create MyVirtual Machine /3D

Operating system:

Microsoft Windows 10 Professional/Enterprise/IoT Enterprise/Home (64 Bit)

2.18.8 Comments

None.

2.19 Component Name: SIMATIC Energy Manager with Insights Hub

2.19.1 Description

Insights Hub is a state of the art industrial IoT as a service solution. With Insights Hub users can ingest and visualize immediate real-time data and analytic results in one centralized location with no development required to unleash digital potential.

The Simatic Energy Manager App is a cloud-based Application in the Siemens Cloud "Insights Hub" to visualize energy data from any kind of energy source. The app offers the possibility to see when, where, and how much energy was consumed online at any time:

- Users have access to the consumption data of machines and plants worldwide.
- With individual KPI types and user-specific dashboards, a user can create a complete overview of the energy consumption of his machines, plants, or entire production facilities from which the user can subsequently derive measures for energy efficiency.
- Users can create a KPI instance directly at the asset in the Energy Manager without having defined a KPI type beforehand.
- In the detail view of the widgets, it is possible to create quick media analyses.
- Transparent listing of energy costs, energy consumption and CO2 emissions of individual machines and all production facilities around the globe. (Energy media analysis)
- Users obtain valuable information on peaks in energy consumption, for example, to make informed decisions regarding optimization of energy efficiency and reducing energy costs.
- The Energy Manager is certified in accordance with ISO 50001.

Within the RE4DY project, SIMATIC Energy Manager with Insights Hub, will be implemented at the Swiss Smart Factory. By including this component into the Swiss Smart Factory energy related data can be generated.

2.19.2 Input

With the wide range of different energy meters, users can collect and evaluate energy consumptions, such as power or gas and increase energy efficiency, among other things. The SIMATIC Energy Manager App provides predefined acquisition categories:

- Process values (e.g., m^3/h)
- Power values (Power)
- Consumption values (Energy)
- Count values (Counter)

MindConnect offers numerous possibilities and open standards for connecting energy meters, e.g., Modbus TCP or OPC UA. It is also possible to get data from existing SCADA systems like PCS7 or directly as PLC via OPCUA/S7 protocol.

2.19.3 Output

After successful integration of the energy meters, users have the following options for further processing the synchronized data:

- Long-term archiving of the data
- Preparation of the data and visualization in a dashboard or reports
- Analysis of the data as well as KPI calculations

Dashboards are free to configure with many kinds of widgets (Gauge, Pie, Line and more). The dashboards contain a graphical representation of variables or KPI types. Widgets are used for graphical representation.

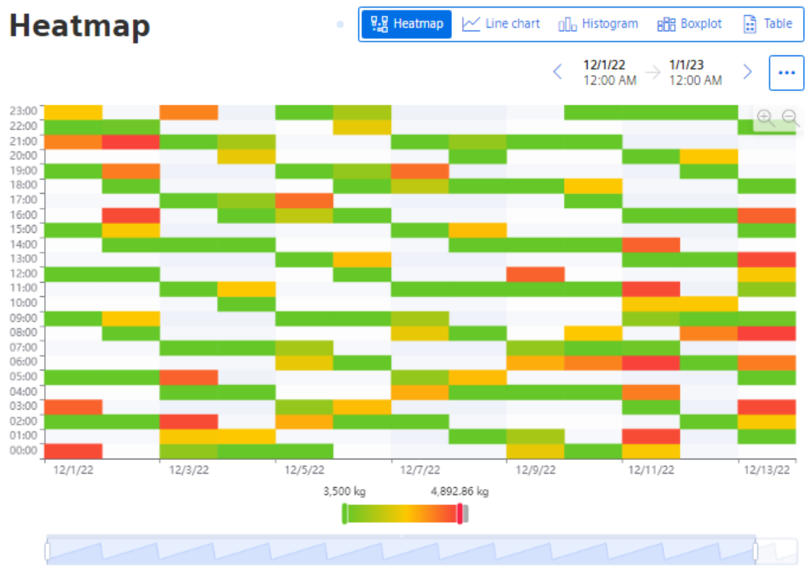


Figure 47: SIMATIC Energy Manager with Insights Hub – Dashboard Heatmap

Reports are a central component of the app. Reports are generated to complement the visualization of KPIs and variables on the dashboards. The reports provide selected information on productivity, energy consumption and costs at regular intervals without the recipient of the reports having to have access to the app. Users can also generate a one-off immediate report whose configuration is not saved. Reports are configured individually and are generated as an Excel spreadsheet.

KPI Calculation is used for indication and analysis of the consumptions. A KPI type is a formula made up of operands, constants and operators. The definition and calculation of KPI types are plant-specific.

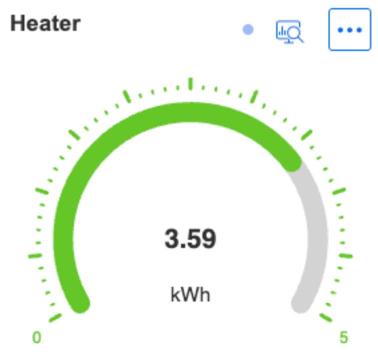


Figure 48: SIMATIC Energy Manager with Insights Hub – Heater Meter

Visualizations using the Sankey diagram, can display the energy flows as arrows whose width is proportional to the flow rate. This makes it easy for users to recognize, for example, how energy is flowing through a plant.

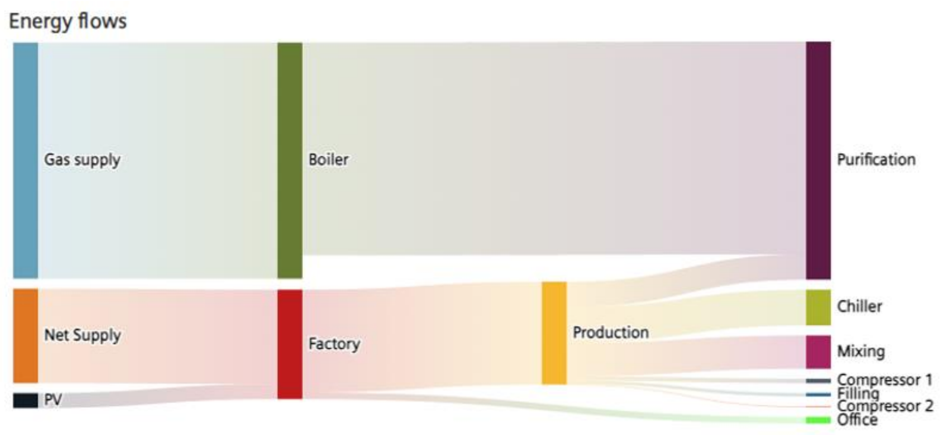


Figure 49: SIMATIC Energy Manager with Insights Hub – Energy Flows Sankey diagram visualization

2.19.4 Information Flow

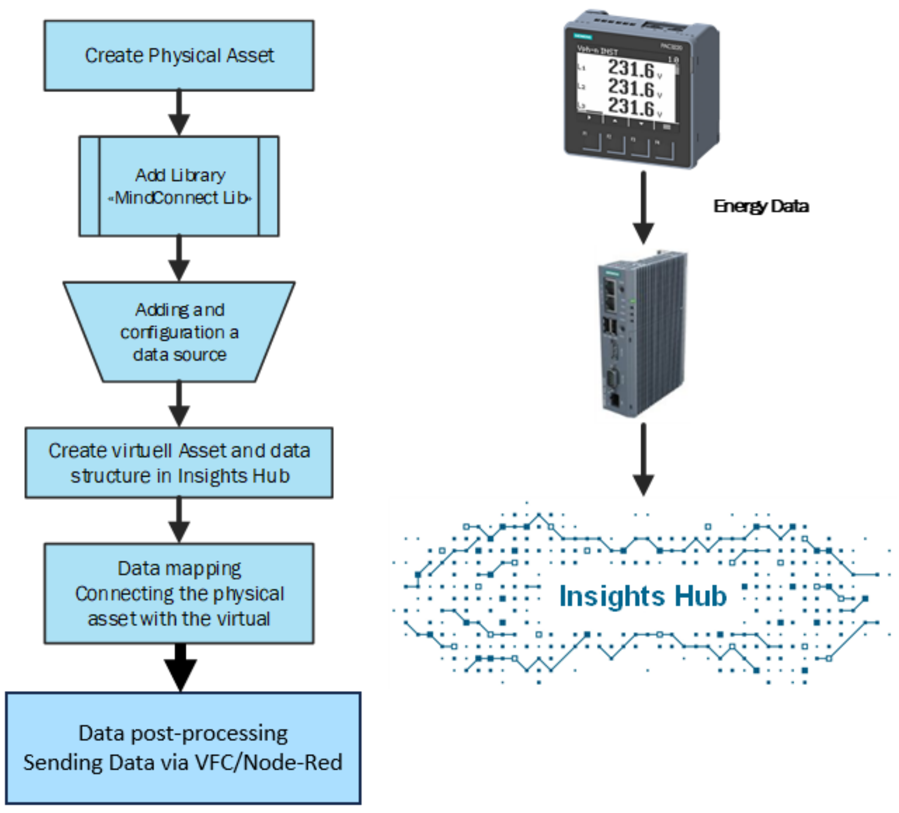

Create Physical Asset
Add Library «MindConnect Lib»
Adding and configuration a data source
Create virtuell Asset and data structure in Insights Hub
Data mapping Connecting the physical asset with the virtual
Data post-processing Sending Data via VFC/Node-Red
Energy Data
Insights Hub

Figure 50: SIMATIC Energy Manager with Insights Hub – Information Flow

In the so-called data mapping, users assign the data points of their physical asset (data source in MindConnect Lib) to those of the virtual asset (data structure in the asset manager). Only after the asset is mapped with a real (physical) asset, the data is stored in the Insights Hub cloud.

Application installation

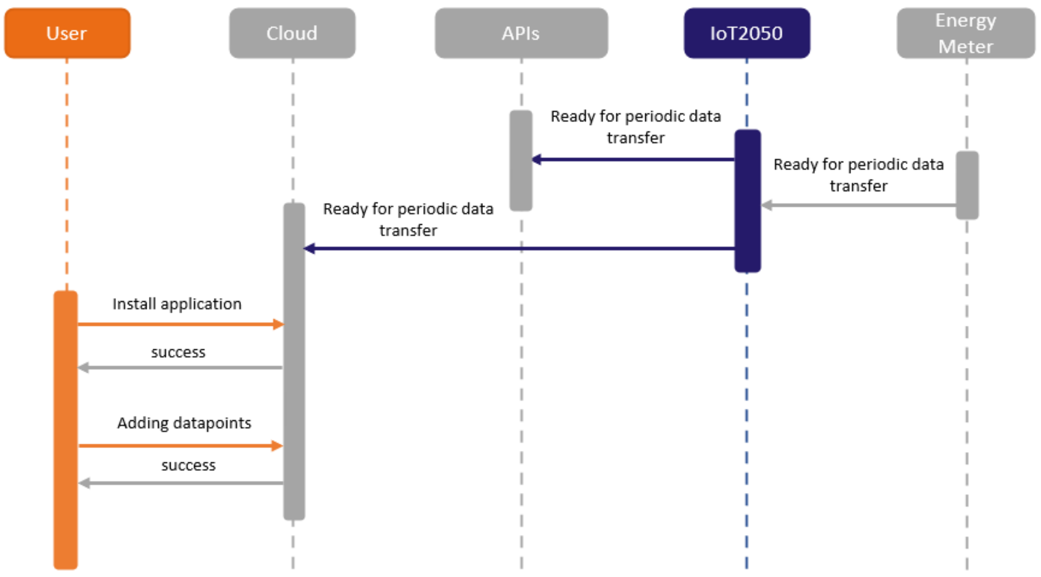

User
Cloud
APIs
IoT2050
Energy Meter
Ready for periodic data transfer
Ready for periodic data transfer
Ready for periodic data transfer
Install application
success
Adding datapoints
success

Figure 51: Application installation

A user can install the Energy Manager application within the Insights Hub cloud platform. Within the application the user can add all data points needed. During operation the Energy Meter will generate data that can be transferred and submitted by a gateway such as the IoT2050. The data can be accessed for example within the Insights Hub cloud platform.

Operation visualization

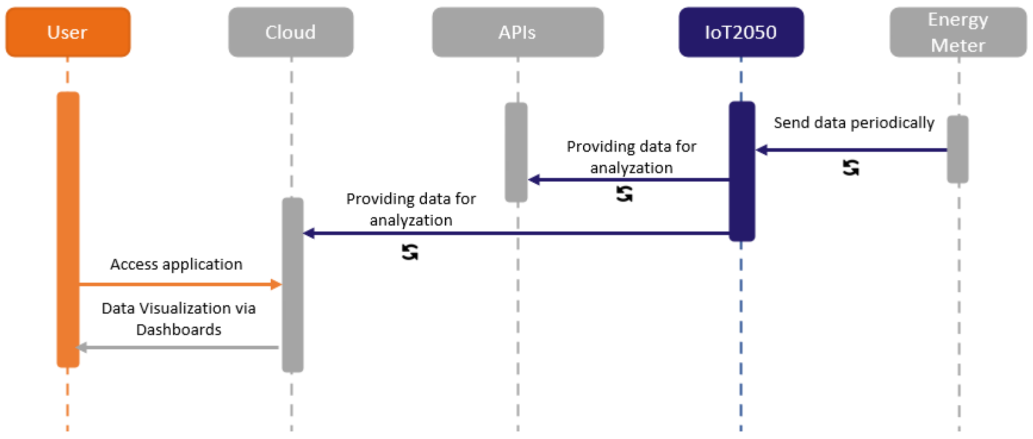

User
Cloud
APIs
IoT2050
Energy Meter
Send data periodically
Providing data for analyzation
Providing data for analyzation
Access application
Data Visualization via Dashboards

Figure 52: Operation visualization

The data, generated by the Energy Meter, can be made available by using a gateway such as the IoT2050. Within the Insights Hub cloud platform, a user can visualize the data by using the possibility of creating dashboards. This allows users to analyze and optimize the operation of the connected asset.

Operation analyzation

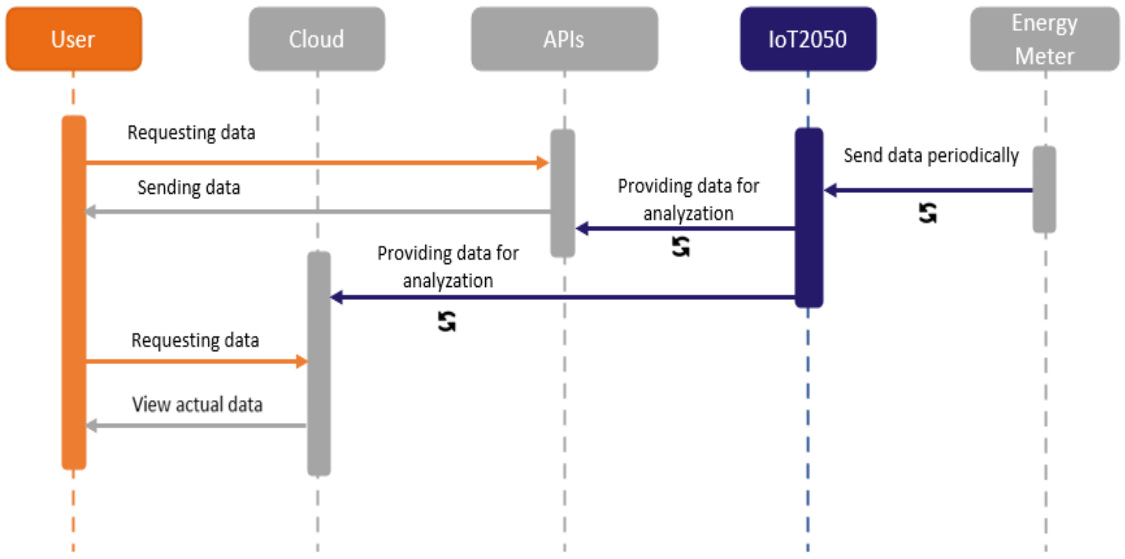


Figure 53: Operation analyzation

The data, generated by the Energy Meter, can be made available by using a gateway such as the IoT2050. Within the Insights Hub cloud platform, a user can analyze the data. This allows to optimize the operation of the connected asset.

2.19.5 Internal Architecture

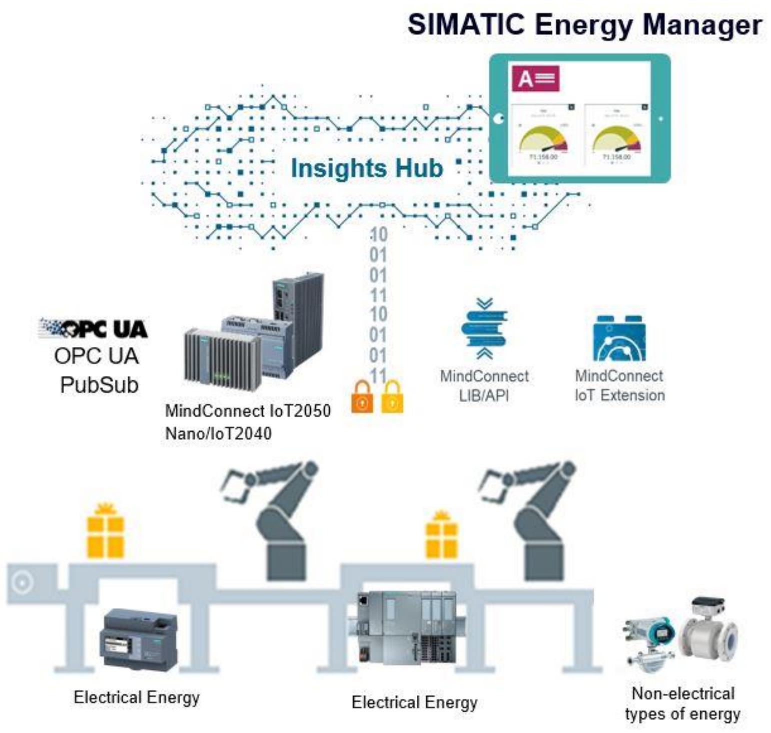


Figure 54: Internal Architecture

At factory level various assets can be connected to the Energy Manger. By using gateways such as MindConnect IoT2050 data can be transferred to further applications. The available data can be analyzed and visualized at cloud level or user preferred applications.

Scalability

By simply adding more assets and connecting them to Insights Hub, energy data can be monitored on different sites. This brings the advantage of fast and easy scalability.

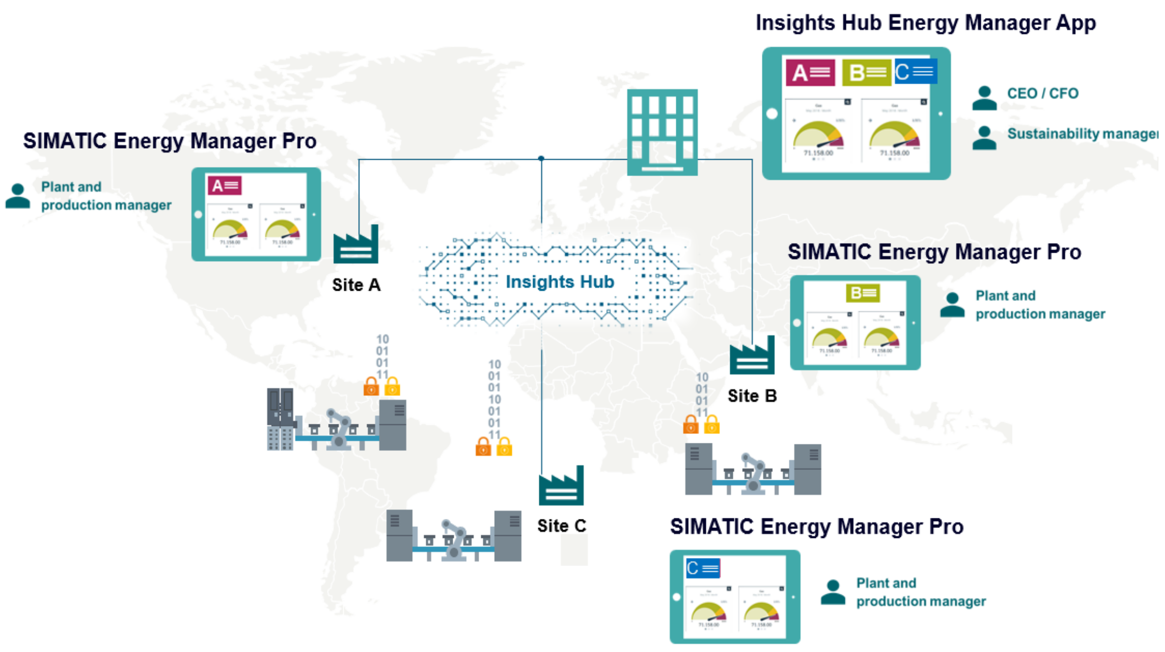

Insights Hub Energy Manager App
CEO / CFO
Sustainability manage
SIMATIC Energy Manager Pro
Plant and production manager
Site A
Insights Hub
SIMATIC Energy Manager Pro
Plant and production manager
Site B
Site C
SIMATIC Energy Manager Pro
Plant and production manager

Figure 55: Scalability

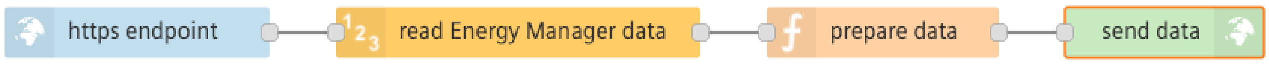
Energy Manager can be connected and configured to a various number of assets. Assets from different locations and sites can be connected to collect data simultaneously. This allows users to gain insight into various plants for making decisions.

2.19.6 APIs

Various interfaces are available for the SIMATIC Energy Manager to exchange data. The following options are possible via another App in “Insights Hub” called “Visual Flow Creator”:

- REST API to get data from an https-endpoint; opposite is providing data by our request.
- REST API to get data via an https-endpoint on our site; opposite site is sending data actively.
- MQTT broker to provide data (opposite site must subscribe or can publish data).
- Data exchange form/to a POSTGRES database located on opposite site.

This is an example of an https-endpoint in Visual Flow Creator to provide data for a cloud2cloud solution:



Further information:

https://documentation.mindsphere.io/resources/pdf/visual-flow-creator-en.pdf

2.19.7 Implementation Technology

Licenses:

SIMATIC Energy Manager

The SIMATIC Energy Manager is provided as a cloud-based Application, which we have implemented in Insights Hub. The main goal is to visualize energy data from any kind of energy source.

As a Gateway, for example Mindconnect IoT2050, is used for the connectivity to the Insights Hub. The IoT2050 collects the data from smart meters or any other system with a previously described interface and sends it to the cloud (Insights Hub). For the application a smart meter with Modbus TCP is used to provide the energy data to the gateway. Due to the easy integration of the devices, no programming skills are needed. We used the MindConnect Library to connect the physical asset with the virtual asset in the cloud.

Insights Hub provides additional Apps, like VisualFlowCreator (Node Red) to consolidate the data before the visualization with Energy Manager is done. After integration of the energy data in the cloud, we have implemented different dashboards to visualize the consumption.

2.19.8 Comments

None.

2.20 Component Name Industrial Edge for Machine Tools

2.20.1 Description

Industrial Edge for Machine Tools is bringing new capabilities to the machine tool, for immediate processing of high-frequency data right where the data is being generated. By decoupling data processing tasks and automation, safe machine operation is always guaranteed. In this secure environment, customers can run their applications – for example to ensure workpiece quality and increase machine availability and productivity. Cloud-based services for Industrial Edge for Machine Tools make it possible to distribute updates and new applications within the shortest possible time. Entire machine parks can thus follow shorter and shorter innovation cycles with maximum efficiency. In addition, machine manufacturers or service providers can integrate their own edge apps into the secure open platform.

As part of the RE4DY project, Industrial Edge for Machine Tools, might be used in the GF and Fraisa pilot. For collecting and exchanging data out of the machine tool from GF (Mill P 800 U S) to a cloud platform, Industrial Edge for Machine Tools, can help make this possible. With the purpose of gaining insight in the operation of the machine tool, the described component can help achieve this.

2.20.2 Input

With the range of ready-to-use apps, it is possible to collect and evaluate machine tool data, monitor the quality of workpieces (including the application of artificial intelligence), and perform condition-based maintenance, among other things.

2.20.3 Output

The industrial Edge for machine tools offers the possibility of an analysis of the available data in the applications provided. Depending on the application the output can vary from alerts to diagrams to data.

2.20.4 Information Flow

Application installation

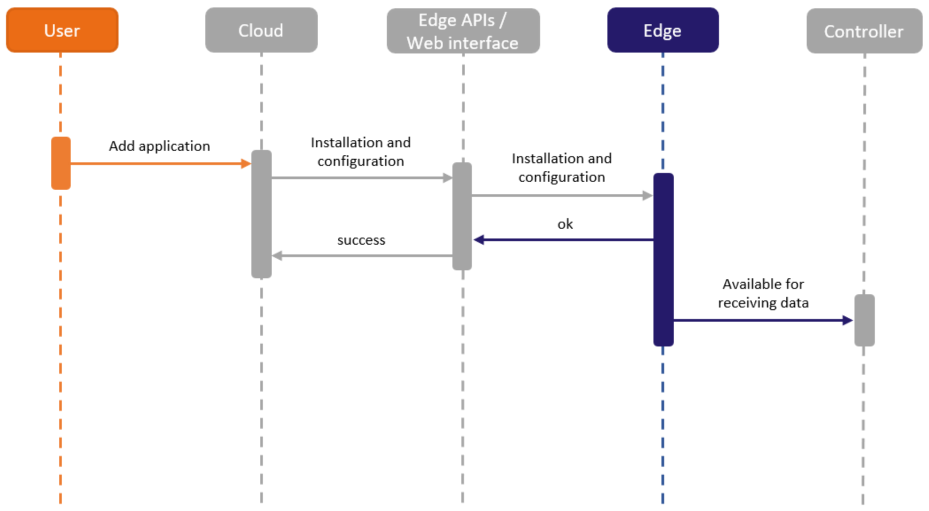

User
Cloud
Edge APIs / Web interface
Edge
Controller
Add application
Installation and configuration
Installation and configuration
ok
success
Available for receiving data

Figure 56: Application installation

An edge hardware such as Siemens SIMATIC IPC427E is required, on which users can install multiple applications. Cloud platforms such as Siemens Insights Hub can be used to install and configure applications at the edge hardware. After successful installation and configuration of the application, data can be submitted from a machine to the installed application.

Application operation

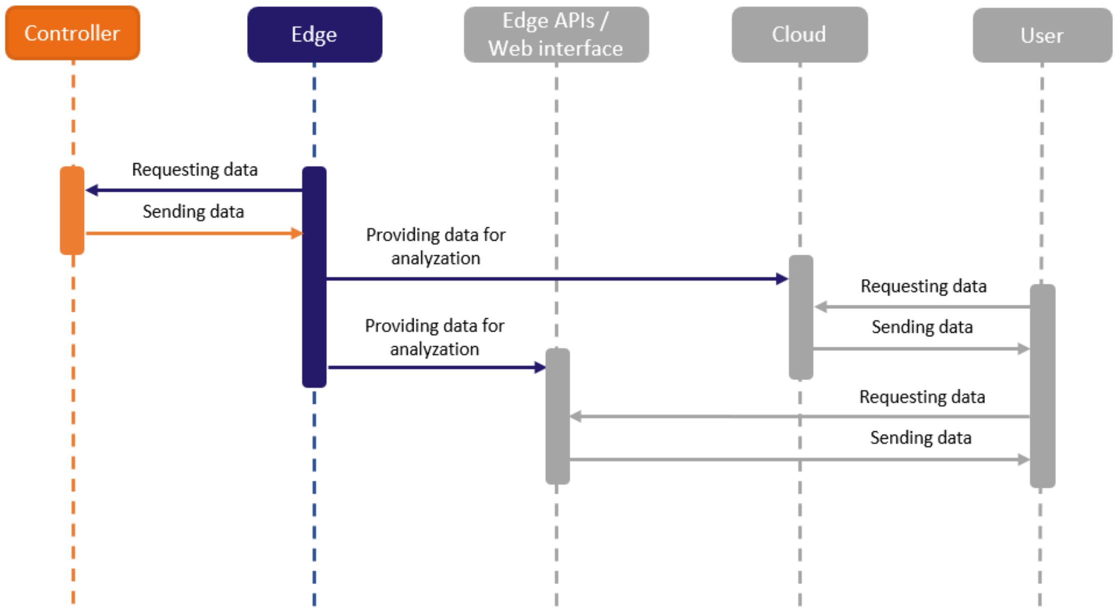


Figure 57: Application operation

The controller of a machine exchanges data with the edge hardware. This data can then be used to analyze and optimize the operation mode of the machine. Furthermore, the data can be made available for the user either by the cloud platform or edge APIs.

Application update

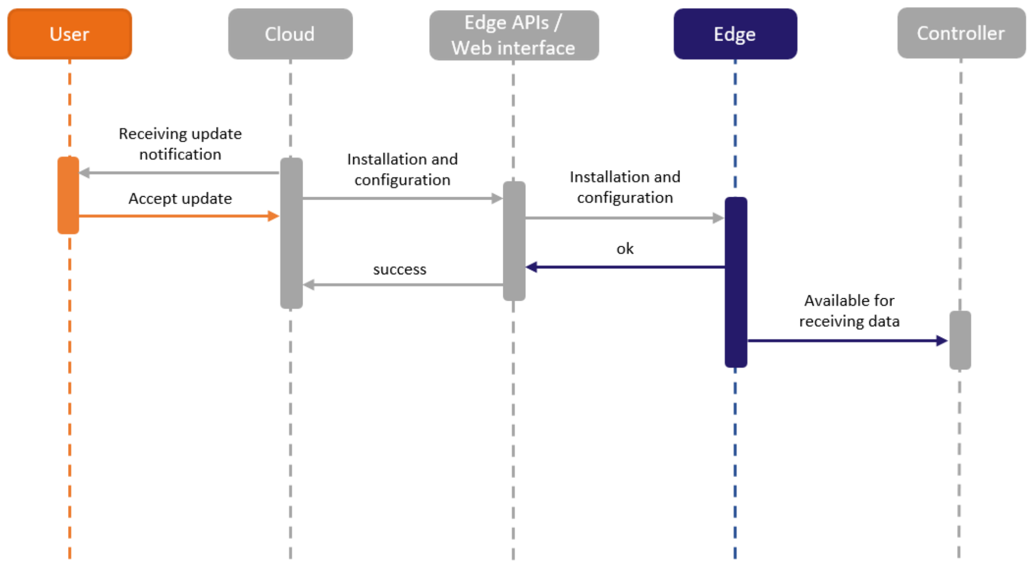

User
Cloud
Edge APIs / Web interface
Edge
Controller
Receiving update notification
Accept update
Installation and configuration
Installation and configuration
ok
success
Available for receiving data

Figure 58: Application update

The installed application on the edge hardware can be updated. Cloud platforms such as Siemens Insights Hub can be used to update applications. As soon as the update process is done successfully, data can be submitted again from a machine to the installed and updated application.

2.20.5 Internal Architecture

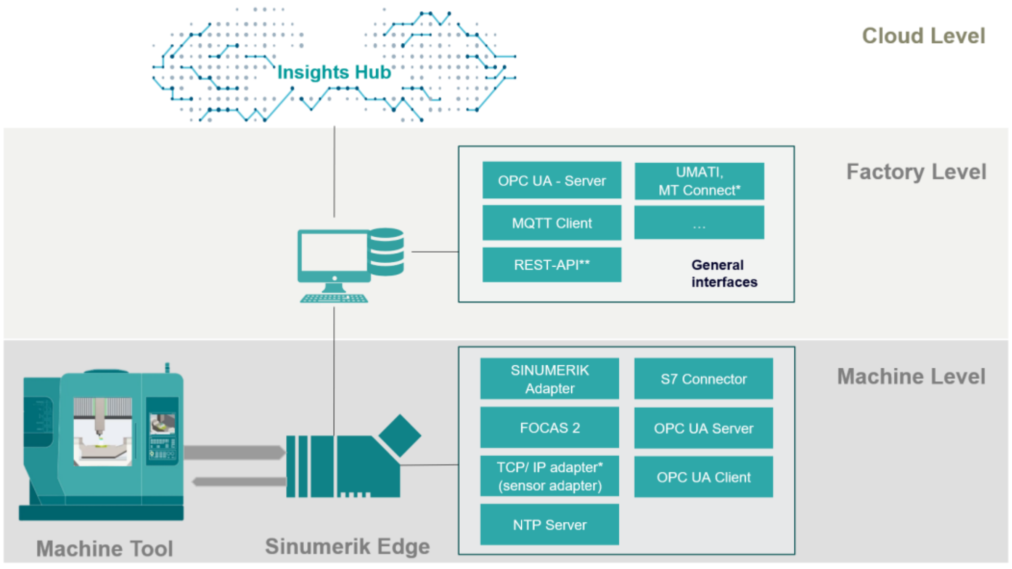


Figure 59: Internal Architecture

The machine tool is located at the machine level. The edge hardware is connected directly to the machine tool for accessing machine data. The edge hardware can send the received data to the cloud level by using common interfaces such as OPC UA. At the cloud level data can be exchanged with other cloud platforms by using API interfaces.

2.20.6 API

Flexible connectivity to any MES and ERP systems, i.e., PLC, OPC UA and umati

Within the industrial edge for machine tools the following applications are available:

- Analyze MyWorkpiece /Capture
- Analyze MyWorkpiece /Monitor
- Protec MyMachine /Setup
- Analyze MyMachine /Condition
- Protect MyMachine /3D Twin

Further information for API connection:

https://developer.siemens.com/apis.html?productLine=Industrial+Edge

2.20.7 Implementation Technology

As an example, a Siemens SIMATIC IPC427E can be used for the operation of the Sinumerik Edge platform. The operating system of the Edge platform is Linux.

Siemens Insights Hub is the cloud platform on which all application installations and configurations are running.

2.20.8 Comments

It is possible to create one's own application based on common docker systems e.g., Python, Matlab, Node-RED.

2.21 Component Name: NX Part Manufacturing

2.21.1 Description

A digital map of the production process is also referred to as a "digital twin" – because the real and the digitalized process should be as similar as possible. By combining the NX CAD/CAM system and the SINUMERIK CNC system, Siemens already offers a digital twin for the entire machining process. By taking a closer look at this process, NX CAM uses exactly the same model that was previously created in CAD. With NX CAM, users can speed up machining and achieve high accuracy through advanced toolpath technology.

By using Manufacturing Resource Library (MRL), the programmer quickly creates new tool assemblies and quickly finds the appropriate tool through a variety of attributes in Teamcenter or directly in NX CAM. Complex measuring operations are considerably simplified by the use of On Machine Probing (OMP). Users can verify measuring cycles directly in NX CAM, via machine simulation to achieve collision avoidance thanks to true NC code and cycle simulation.

For the most accurate machine simulation possible, the manufacturer's exact machine kinematics are stored in the 3D model. When it comes to post-processing, NX CAM Post Hub delivers the modern cloud-based solution. Free for NX CAM users, Post Hub enables a streamlined process for creating production-ready CNC programs for users' applications. With just a few clicks, Post Hub provides the machine kit (MK) and the corresponding post processor (PP). With Run MyVirtual Machine (RMVM) the hardware components of the control are modeled as software components and represent a complete image of a real CNC control.

By combining RMVM and the real NC code from NX CAM, it is possible to achieve a true 1:1 simulation. After verification, the program is released for production in Teamcenter (TC). Afterwards, the NC program can be transferred directly to the machine using Shop Floor Connect (SFC) and all production information is accessible via a device with a browser.

As part of the RE4DY project, NX Part Manufacturing is being used in the GF and Fraisa pilot. With NX Part Manufacturing, a CAD designed workpiece can be prepared for production using elements such as NX CAD/CAM system and post processor. All generated data during this process can be used for optimization, quality improvement and to build a transparent data process.

2.21.2 Input

NX CAD/CAM:

- 3D-models of the design part, raw material, machine with machine kinematics, tools and fixtures etc.
- CAM program includes all the information, operation order, tool selection, toolpath movements, feeds and speeds etc.

MRL:

- Import DIN vendor catalog data into Teamcenter and map to MRL component classes
- Import 3D model (and convert to NX parts)
- Create connection rules

OMP:

- A selection of the necessary tool and zero-point adjustments and control structures. Measuring operations and measuring activities are seamlessly integrated into the graphically oriented programming as easy-to-use NX operations. Measurement objects are selected on the 3D model, are associative to the component and measuring cycles are displayed as normal tool paths.

MK / Post Hub:

- A simple selection of the machine, controller, or manufacturing technology and Post Hub will present the applicable postprocessors

RMVM:

- In order to use Run MyVirtual Machine /3D correctly, the machine project must be provided with the 3D model of the machine including kinematics so that the project can be put into operation accordingly.
- The NC program generated from the Post Hub

SFC:

- After a verified simulation, the CAM-object in Teamcenter (including NC-programs, setup sheets, clamping plans, tool lists, drawings or 3D models) receives the status "approved" and is now available on the machine

2.21.3 Output

NX CAD/CAM:

- Run tool path verification
- Reusable process and program

MRL:

- Get automatic generated tool assemblies
- Browse for available tools in the library
- Select and use resources in NX CAM session

OMP:

- The 3D simulation simulates and visualizes measuring processes with touch trigger probes. The measuring process is identical to the process on a real machine and has the same prerequisites.
- Continuous quality assurance during production

MK / Post Hub:

- NC-program
- Setup and tool list document for the shop floor

RMVM:

- Full machine simulation with collision check
- Machine simulation with material removal

SFC:

- Secure and professionally managed data and information
- Download and Upload of NC Package (NC pgr, shopdoc and toollist)
- Available on every device via browser

2.21.4 Information Flow

Installation for NX part manufacturing

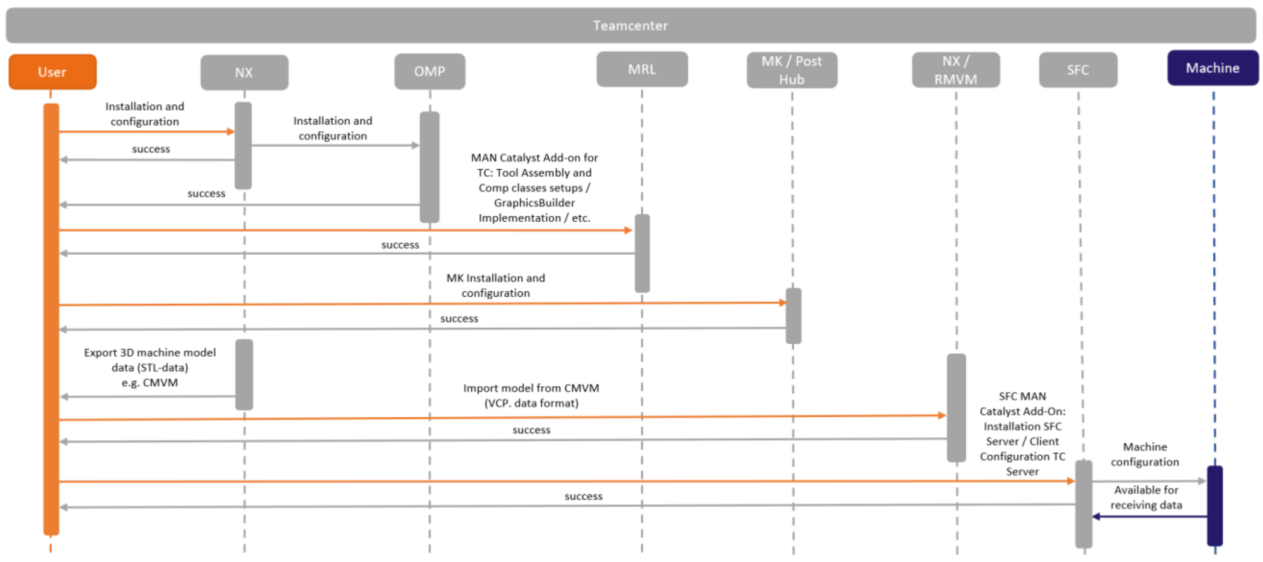

Teamcenter
User
NX
OMP
MRL
MK / Post Hub
NX / RMVM
SFC
Machine
Installation and configuration
Installation and configuration
success
success
MAN Catalyst Add-on for TC: Tool Assembly and Comp classes setups / GraphicsBuilder Implementation / etc.
success
MK Installation and configuration
success
Export 3D machine model data (STL-data) e.g. CMVM
Import model from CMVM (VCP. data format)
success
SFC MAN Catalyst Add-On: Installation SFC Server / Client Configuration TC Server
Machine configuration
success
Available for receiving data

Figure 60: Installation for NX part manufacturing

The user needs to install the NX CAD application. Created NX CAD drawings can be exported as STL data files that can be used by the Create MyVirtual Machine application.

Within the NX CAD application further applications such as On Machine Probing can be installed and configured. Further elements such as Manufacturing Resource Library, Machine Kit and Shop Floor Connect must be prepared and installed to make the part manufacturing available.

2.21.5 Internal Architecture

NX CAD/CAM / OMP / MRL / RMVM / SFC:

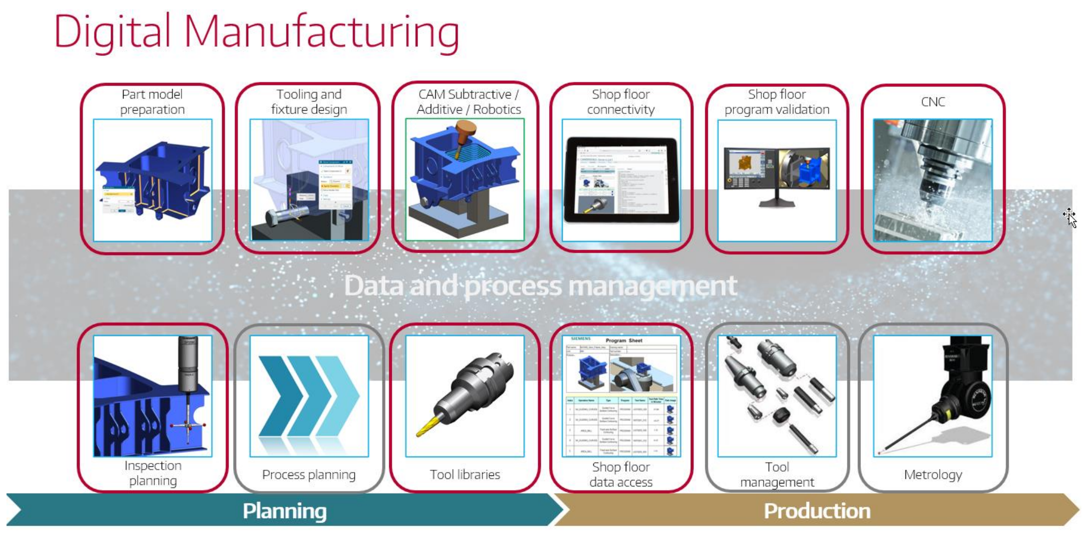


Figure 61: Internal Architecture

In the digital manufacturing process several applications must be installed and configured. The manufactured part is designed in NX. In addition, applications and elements such as tool libraries, shop floor connectivity, inspection planning or tooling and fixture design are needed to complete the digital manufacturing process.

2.21.6 API

NX CAD/CAM:

NX software provides an automation architecture that serves as the foundation for all NX APIs as well as for a new journaling utility. Called the Common API, it combines the power of journaling and automation with the freedom of a language-neutral platform. Integrated within the core NX architecture, the Common API is the foundation for all NX solutions and is fully compatible with the existing Open C API.

MRL: N/A

OMP: N/A

MK / Post Hub: N/A

RMVM:

The open interface allows an external application to control the SINUMERIK ONE system and to communicate with SINUMERIK ONE Application during runtime. The task scope of the open interface addresses various use cases (selection):

- Remote control of the SINUMERIK ONE application for example, in automatic inspection and test units

- Cyclic exchange of runtime information, for example, to connect external simulation systems (I/Os, virtual machine applications etc.)

SFC:

The connection of all NC controls works directly via network or serial via Com-Server++.

2.21.7 Implementation Technology

NX CAD/CAM:

Provides a native .NET API that supports all .NET languages including Visual Basic .NET and C#. The Java and Open C++ APIs support the full range of Common API capabilities.

MRL: Teamcenter application

OMP: Python based

MK / Post Hub: Directly integrated in NX CAM

RMVM:

License: Run MyVirtual Machine /3D Operating system: Microsoft Windows 10 Professional/Enterprise/IoT Enterprise/Home (64 Bit)

SFC:

Shop Floor Connect (SFC) supports Microsoft SQL Server 2017, 2019, 2022 or Oracle 12c as well as Windows 10 and 11. The following browsers are supported, Mozilla Firefox, Google Chrome and Microsoft Edge. The existing standard interface for connection to NX CAM allows the data and information generated there to be stored automatically.

Further information:

NXOpen.CAM Package — NXOpen Python API Reference 10.0.0 documentation (siemens.com)

2.21.8 Comments

N/A

2.22 Component Name: Run MyVirtual Machine

2.22.1 Description

Run MyVirtual Machine is an NC programming workstation with identical controls on the PC for machine tools. The hardware components of the control are modeled as software components and represent a complete image of a real CNC. With Run MyVirtual Machine, a user can simulate and test the next control generation in the development phase of a CNC machine, or NCK, PLC and HMI software without requiring any hardware. Parts of the machine commissioning are preconfigured on the virtual model. It is possible to shorten the commissioning time of the real machine by configuring the system on the virtual model. Run MyVirtual Machine requires a .VCP data file to be imported into the Run MyVirtual Machine software application. This .VCP data file must first be exported from Create MyVirtual Machine.

Run MyVirtual Machine /Operate uses the .VCP data file from Create MyVirtual Machine to activate the operation. Create MyVirtual Machine /Open allows the user to connect additional applications to their machine. Create MyVirtual Machine /3D shows a visual simulation of the machine and workpiece, allowing the user to visually test how the system will run in the operating system.

As part of the RE4DY project, Run MyVirtual Machine is being used in the GF and Fraisa pilot. With Run MyVirtual Machine, GF's physical machine (Mill P 800 U S) can be run virtually during operation.

2.22.2 Input

Run MyVirtual Machine /Operate

The operation of Run MyVirtual Machine corresponds to that of a real control equipped with a SINUMERIK Operate user interface and machine control panel. Run MyVirtual Machine therefore simulates the periphery by direct reading and writing in the PLC I/O image. Parts of the PLC I/O image are also used for internal communications. Run MyVirtual Machine /Operate allows the user to control the virtual machine like a real physical machine. To do this, the virtual machine must first be imported as a .VCP data file from Create MyVirtual Machine.

Run MyVirtual Machine /Open

Run MyVirtual Machine /Open is an additional option to Run MyVirtual Machine /Operate. Run MyVirtual Machine /Open can be used to include external software applications such as a user machine room simulation.

Run MyVirtual Machine /3D

Run MyVirtual Machine /3D is an additional option to Run MyVirtual Machine /Operate. This option extends Run MyVirtual Machine /Operate with an integrated 3D machining and material removal simulation. This allows the user to visually evaluate machine movements and verify collision-free operation. With the material removal simulation, the machining of workpieces can be tested in advance by simulation. The 3D simulation is also ideal for training setup procedures and running machines on a virtual model without exposure to any risks whatsoever.

2.22.3 Output

Run MyVirtual Machine /Operate

Checking the NC program ensures machine instructions are error-free. Furthermore, estimating machining time aids production planning and it helps OEM applications on the HMI optimizing machine operations. It also contains a postprocessor verification to ensure the code compatibility.

Run MyVirtual Machine /Open

The .Net/C++ programming interface provides a powerful foundation for creating automation solutions. Leveraging this interface, the realization of test automation becomes efficient and adaptable, enabling comprehensive testing across diverse applications. Its flexibility allows seamless integration into existing 3D simulation environments, enhancing their capabilities and accuracy and providing an open environment for running a 3D simulation.

Run MyVirtual Machine /3D

Simulation is instrumental in the manufacturing process, particularly in NC program verification with kinematics simulation, where it enables a virtual test to run machining operations, ensuring precision and accuracy. Collision detection, a key element of this simulation, serves as a safety net by identifying potential clashes between machine components and materials, mitigating costly accidents and damage. Additionally, having a library of tools, clamping devices, and blanks within the simulation environment streamlines setup and programming, further enhancing the efficiency and reliability of CNC machining operations.

2.22.4 Information Flow

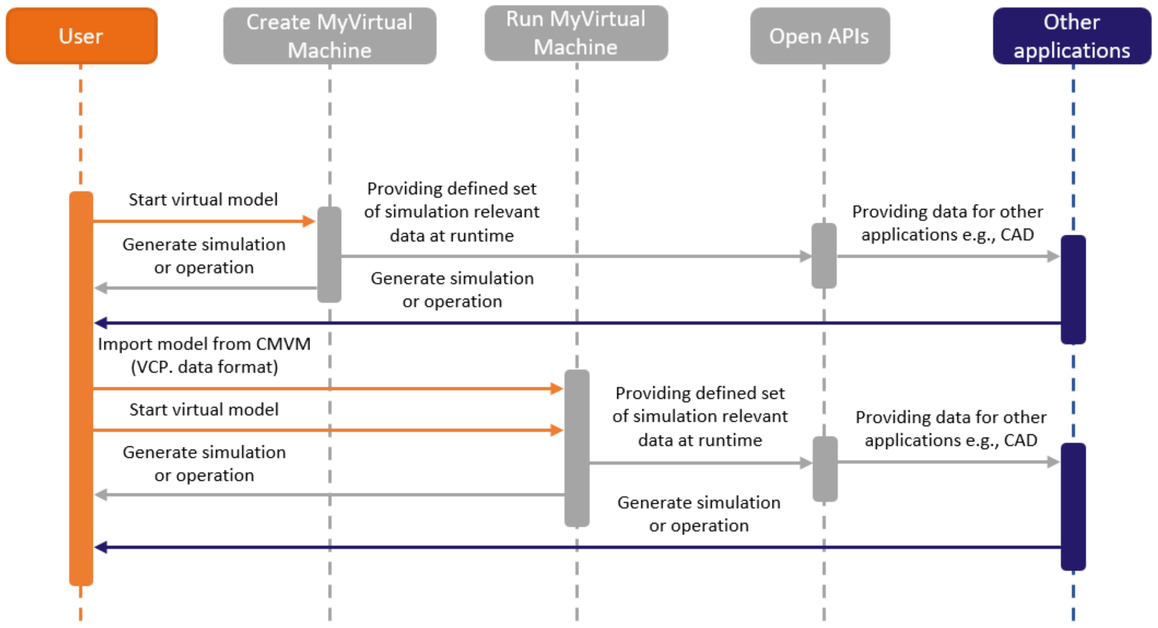

User
Create MyVirtual Machine
Run MyVirtual Machine
Open APIs
Other applications
Start virtual model
Generate simulation or operation
Providing defined set of simulation relevant data at runtime
Generate simulation or operation
Providing data for other applications e.g., CAD
Import model from CMVM (VCP. data format)
Start virtual model
Generate simulation or operation
Providing defined set of simulation relevant data at runtime
Generate simulation or operation
Providing data for other applications e.g., CAD

Figure 62: Virtual operate

The virtual machine can be run in Run MyVirtual Machine. A user can run, simulate, and analyze the virtual machine within the application. To do this, the user must import the .VCP data file from Create MyVirtual Machine. Simulation-related data is generated during runtime. This data can be used for analysis and optimization and can also be made available for other applications for analysis and optimization.

2.22.5 Internal Architecture

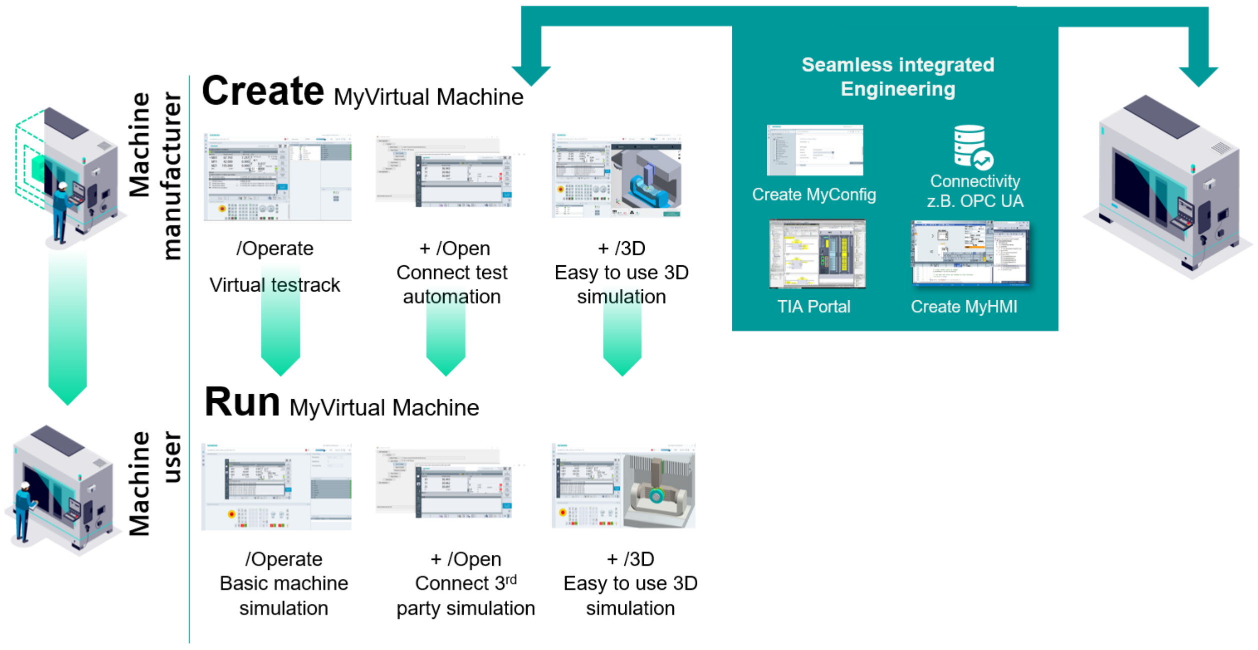


Figure 63: Internal Architecture

On the one hand, a machine manufacturer can create a real machine virtually. To do this, he must provide and integrate engineering data such as control code or CAD-STL data. With Create MyVirtual Machine /Operate a user can operate the virtual machine within the application. In addition, Create MyVirtual Machine /3D can be used to visualize the 3D simulation to get a better understanding of how the machine moves.

On the other hand, the user of the machine can run the digital twin of a real machine using Run MyVirtual Machine. To do this, a .VCP data file must be imported into the Run MyVirtual Machine application. Run MyVirtual Machine /Operate allows the user to simulate and run the virtual machine during runtime. Run MyVirtual Machine /3D can be used to visualize the simulation to get a better understanding of how the machine moves.

2.22.6 API

The Open Interface of Run MyVirtual Machine allows external applications to control the Run MyVirtual Machine system and to communicate at runtime. Possible applications of the Open Interfaces are:

- Remotely controlling Run MyVirtual Machine. An external application starts, operates and exits Run MyVirtual Machine.
- Operating Run MyVirtual Machine in an external simulation product.

- Connecting to an external peripheral simulation.

Create MyVirtual Machine or Run MyVirtual Machine provides a defined set of simulation relevant data at runtime.

Further information: https://support.industry.siemens.com/cs/attachments/109817568/AppNote_Run_MyVirtual_Machine_Open_en.pdf

2.22.7 Implementation Technology

Licenses:

Run MyVirtual Machine /Operate

Run MyVirtual Machine /Open

Run MyVirtual Machine /3D

Operating system:

Microsoft Windows 10 Professional/Enterprise/IoT Enterprise/Home (64 Bit)

2.22.8 Comments

None.

2.23 Component Name: Data Analytics and Visualization Environment

2.23.1 Description

The Data Analytics and Visualization Environment allows the creation of data analytics workflows in a dynamic way. This solution enables the creation of AI workflows in a simple and intuitive, code-free manner, building workflows using a visual programming environment to place the components through drag-and-drop interface.

The Data Analytics and Visualization Environment solution comprises four major components. The first one is the Apache Airflow platform, responsible for the orchestration of the workflows, meaning the scheduling, execution, monitoring and storage of the workflows. The second component is the set of supporting technologies, which encompasses all the necessary supporting technologies for workflow creation, as well as

AI, Data Analytics and machine learning algorithms. The third component is the set of operators (connectors, transformers and analytics) used to create workflows, and finally, the last component is the User Interface (UI), which guides users through the dynamic definition of workflows, by enabling the creation of workflows that can connect to data sources (data source configuration, connection and storage), perform data preparation and pre-processing (data filtering, aggregation, harmonisation and semantic enrichment) and apply AI methods for AI model training, updating and serving and Data Analytics (selection and configuration of AI methods).

In the RE4DY project this tool is used to create and automate data integration tasks and create and automate analytic optimization tasks, as well as provide data visualization capabilities.

2.23.2 Input

The Data Analytics Environment requires the selection of operators from the user in order to form the desired workflows through the user guided interface.

The tool is equipped with multiple operators to support the input of datasets in multiple formats like Excel or CSV files, databases or message brokers.

2.23.3 Output

Workflows that can be saved to be re-utilized or edited in the future.

Trained AI models - Models that were trained in the workflows.

Modified Datasets - Datasets resulting from the transformations applied in the workflows.

Dashboards – Data dashboards can be created to visualize data as well as results from the workflows.

2.23.4 Information Flow

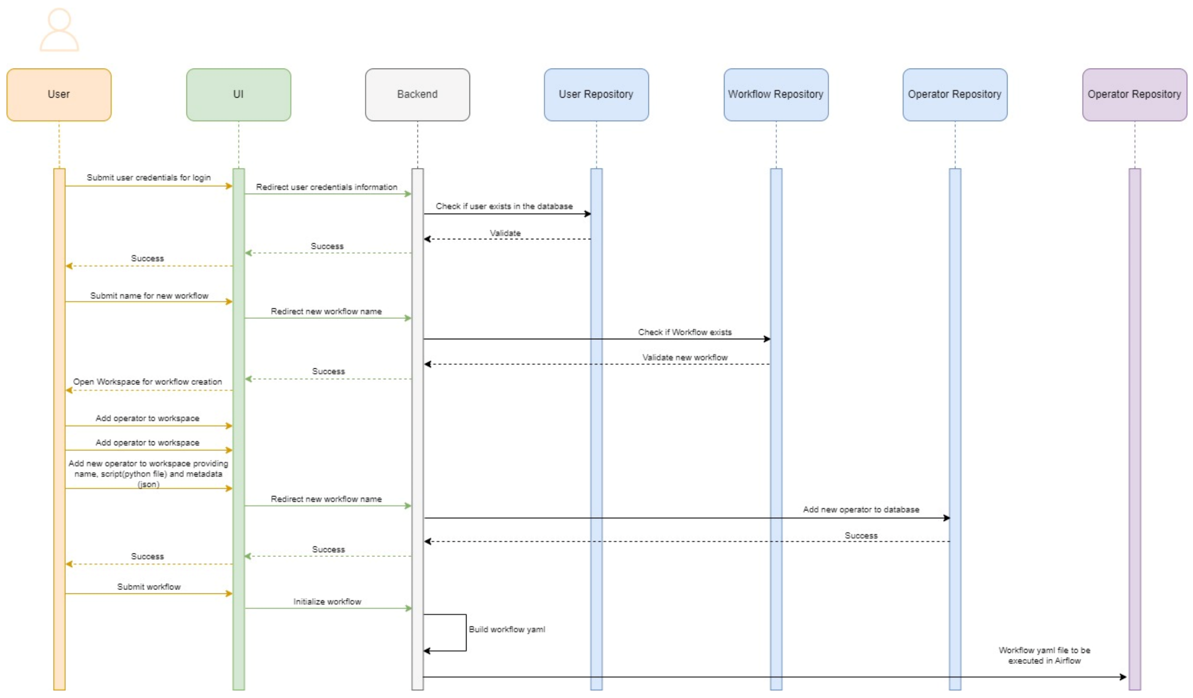

User
UI
Backend
User Repository
Workflow Repository
Operator Repository
Operator Repository
Submit user credentials for login
Redirect user credentials information
Check if user exists in the database
Validate
Success
Success
Submit name for new workflow
Redirect new workflow name
Check if Workflow exists
Validate new workflow
Success
Open Workspace for workflow creation
Add operator to workspace
Add operator to workspace
Add new operator to workspace providing name, script(python file) and metadata (json)
Redirect new workflow name
Add new operator to database
Success
Success
Success
Submit workflow
Initialize workflow
Build workflow yaml
Workflow yaml file to be executed in Airflow

Figure 64: DAVE information flow

Sample operator JSON metadata:

```
{
"op_name": "op_conn_rdbms",
"op_pres_name": "RDBMS Connection Operator",
"op_pres_desc": "Connector for RDBMS databases",
"op_type": "connector",
"airflow_op_type": "PythonOperator",
"visibility": "public",
"op_params":{
    "python_callable_file":"op_conn_rdbms.py",
    "python_callable_name":"main",
    "op_kwargs": [{
        "name": "driver",
        "pres_name":"Database Driver",
        "pres_desc":"Database Driver for the RDBMS database",
        "type":"enum_string",
        "accepted_values": ["org.postgresql.Driver", "com.mysql.jdbc.Driver"]
    }, {
        "name": "url",
        "pres_name":"Database URL",
        "pres_desc":"URL of the RDBMS database",
        "type":"string"
    }, {
        "name": "port",
        "pres_name":"Database Port",
        "pres_desc":"Port of the RDBMS database",
        "type":"int"
    }, {
        "name": "db",
        "pres_name":"Database Name",
        "pres_desc":"Name of the RDBMS database",
        "type":"string"
    }, {
        "name": "dbtable",
        "pres_name":"Database Table Name",
        "pres_desc":"Name of the RDBMS database table",
        "type":"string"
    }, {
        "name": "user",
        "pres_name":"Username",
        "pres_desc":"Username of the account to login to the database",
        "type":"string"
    }, {
        "name": "password",
        "pres_name":"Password",
        "pres_desc":"Password of the account to login to the database",
        "type":"string"
    }]
}
}
```

Workflow JSON metadata:

```
{
    "wf_meta": "workflow_name",
    "wf_meta_pres_name": "Workflow name",
    "wf_meta_pres_desc": "Name of the workflow given by the user.",
    "default_args": [{
        "name": "owner",
        "pres_name": "Workflow owner",
        "pres_desc": "Owner/creator of the workflow",
        "type": "string"
    }, {
        "name": "start_data",
        "pres_name": "Start Date",
        "pres_desc": "Scheduled starting date of the workflow.",
        "type": "date",
        "format": "yyyy-MM-dd"
    }, {
        "name": "retries",
        "pres_name": "Retries",
        "pres_desc": "Number of retries in case of a failed run.",
        "type": "int",
        "default": 1
    }, {
        "name": "retry_delay_sec",
        "pres_name": "Retry delay",
        "pres_desc": "Delay, in seconds, between retries",
        "type": "int",
        "default": 300
    }],
    "args": [{
        "name": "schedule_interval",
        "pres_name": "Scheduled Interval",
        "pres_desc": "Interval between the scheduled runs of the workflow",
        "type": "enum_string",
        "accepted_values": ["@hourly" ,"@daily", "@weekly"],
        "default": "@daily"
    }, {
        "name": "concurrency",
        "pres_name": "Concurrency",
        "pres_desc": "Maximum number of concurrent tasks running in the workflow.",
        "type": "int",
        "default": 1
    }, {
        "name": "max_active_runs",
        "pres_name": "Maximum Active Runs",
        "pres_desc": "Maximum number of active runs of the workflow.",
        "type": "int",
        "default": 1
    }, {
        "name": "dagrun_timeout_sec",
        "pres_name": "Dagrun Timeout",
        "pres_desc": "Time. in seconds, the workflow can run before being timed out.",
        "type": "int",
        "default": 3600
    }, {
        "name": "default_view",
        "pres_name": "Default View",
        "pres_desc": "Default view of the workflow in the Airflow UI.",
        "type": "enum_string",
        "accepted_values": ["LR", "TB", "RL", "BT"],
        "default": "LR"
    }]
}
```

2.23.5 Internal Architecture

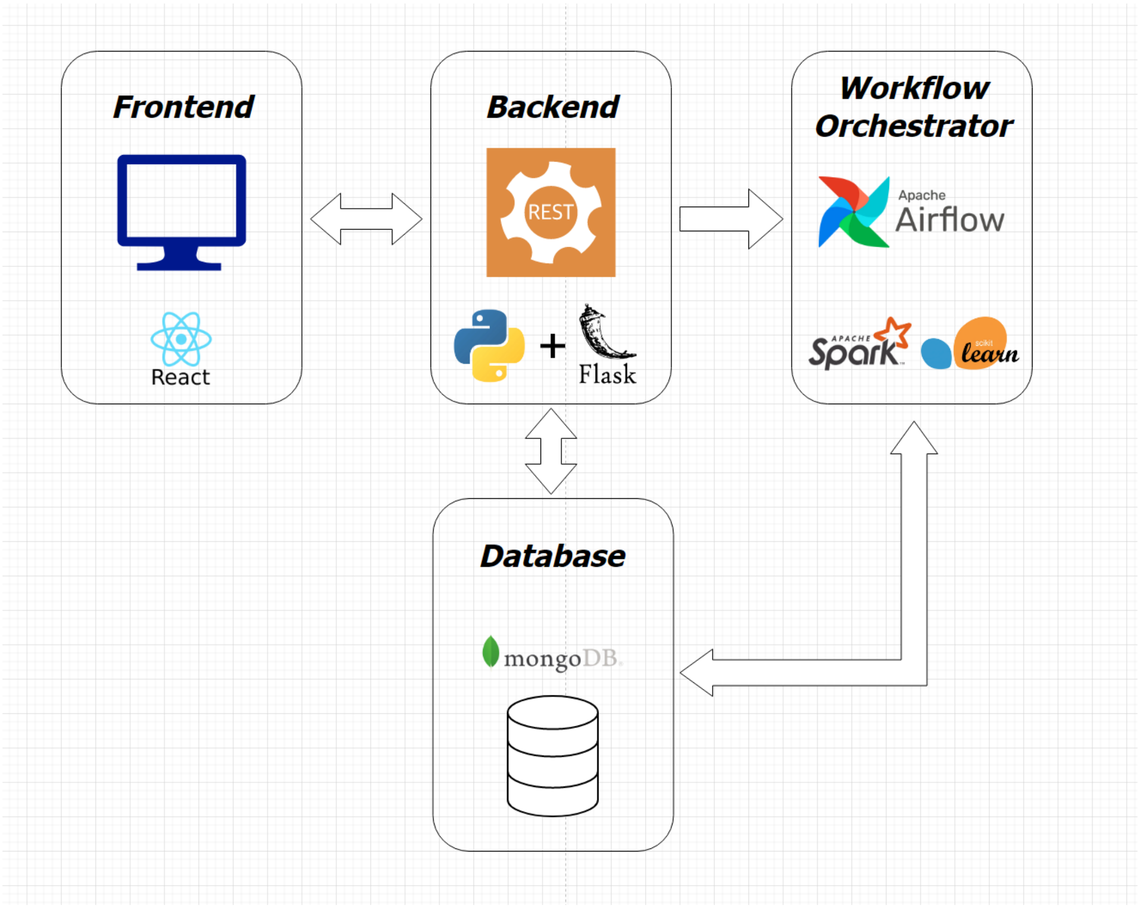


Figure 65: DAVE Architecture

2.23.6 API

The internal API was designed to be used by our frontend and integration with other tools. It is not supposed to be used by outside connections.

Create workflow	
Description	Creates an empty workflow JSON file, where the operators will then be added.
HTTP Method	POST
Endpoint URL	http://host:port/analytics/create_workflow
Parameters	Check Workflow JSON metadata above.

Outputs	HTTP Response Code: 201 Created

Update workflow	
Description	Edits the parameters of the chosen workflow, such as the starting date of the execution, the frequency of the execution and other parameters.
HTTP Method	POST
Endpoint URL	http://host:port/analytics/update_workflow
Parameters	Check Workflow JSON metadata above.
Outputs	HTTP Response Code: 201 Updated

Initialise workflow execution	
Description	Converts the workflow JSON file, with the added operators, to a YAML file, initializes the workflow in Airflow, using the DAG factory library, and saves the workflow to an intermediate storage database.
HTTP Method	POST
Endpoint URL	http://host:port/analytics/init_workflow
Parameters	Check Workflow JSON metadata above.
Outputs	HTTP Response Code: 201 Initiated -> Airflow UI

Get workflow metadata	
Description	Gets the JSON metadata for a workflow. This metadata contains all the accepted inputs for the fields required for the creation of the DAG workflow.
HTTP Method	GET

Endpoint URL	http://host:port/analytics/get_workflow_metadata
Parameters	N.A.
Outputs	HTTP Response Code: 200 OK -> Check Workflow JSON metadata above.

Delete workflow	
Description	Deletes a workflow from the intermediate storage database.
HTTP Method	POST
Endpoint URL	http://host:port/analytics/delete_workflow
Parameters	Workflow Name
Outputs	HTTP Response Code: 201 Deleted

Add operator	
Description	Adds an operator to the current workflow according to the input parameters. Gets the operator type, all the necessary parameters for that operator type and the workflow where the operator will be added.
HTTP Method	POST
Endpoint URL	http://host:port/analytics/add_operator
Parameters	Workflow name Check Sample operator JSON metadata above.
Outputs	HTTP Response Code: 201 Added

Get operators metadata

Description	Gets the metadata for all the available operators. This metadata contains all the parameters that each operator accepts, as well as all the accepted inputs.
HTTP Method	GET
Endpoint URL	http://host:port/analytics/get_operators_metadata
Parameters	N.A.
Outputs	HTTP Response Code: 200 OK -> Check Sample operator JSON metadata above.

Delete operator	
Description	Deletes a selected Operator from a given workflow.
HTTP Method	POST
Endpoint URL	http://host:port/analytics/delete_operator
Parameters	Operator Name, Workflow Name
Outputs	HTTP Response Code: 201 Deleted

Furthermore, Airflow also provides a RESTful API that can be used, for instance, to trigger workflows on demand. The Airflow REST API can be used in a number of situations by users, such as in the case of triggering a new workflow execution or listing the set of available workflows for a particular user. The Airflow REST API documentation is available through the following link: https://airflow.apache.org/docs/apache-airflow/stable/stable-rest-api-ref.html.

2.23.7 Implementation Technology

The Data Analytics and Visualization Environment is distributed in Linux containers. The backend consists of the Apache Airflow backend and a Python Flask API for the different operators. The frontend was developed using the React framework.

The available operators use multiple libraries and frameworks and database connections:

- Keras

- Pandas
- scikit-learn
- Apache Superset
- Apache Spark
- Apache Kafka
- PostgreSQL
- MongoDB
- MSSQL
- MonetDB

2.23.8 Comments

None.

2.24 Component Name: Data Container

2.24.1 Description

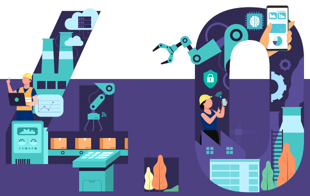
The Data Containers (DC) provide access to the data following the Data as a Product approach while, at the same time, creating an abstraction layer that will hide all the underlying technical complexity from the users. DC will allow the applications to access the data in a trusted and reliable way regardless of the location (Edge or Cloud) and particularities of the data sources. The DC will provide the data according to the consumer's requirements in terms of format and data quality, with the proper considerations about security, privacy, and performance.

2.24.2 Input

The DC should be able to obtain the data from the sources (directly or indirectly) and call the necessary components of the RE4DY architecture in order to serve the data following the Data as a Product approach.

Inputs of the DC API:

- Dataset id
- Format: defines the output data format (for example, JSON, Parquet or CSV).



- Rules: allow the data consumer to select a subset of the available data. Rules can also be used to remove outliers from the data. A rule consists of three elements:
  - Subject: variable where the rule is applied.
  - Operator: the allowed operators are the following: "<=", ">=", "<", ">", "==", "!=", "or", "++" (include only the columns whose names are specified in the "object" value), and "—" (do not include the columns whose names are specified in the "object" value).
  - Object: value.

2.24.3 Output

The Data Containers will provide data and metadata.

Metadata request

The metadata will be sent in the HTTP response in JSON format.

Data request

- API response: indicates whether the request was completed successfully or not.
- Notification: is sent to the specified HTTP endpoint once the process is completed and data is available (in the prototype, the URL of this endpoint is defined in the DC configuration). Example:

  ```
  {
    "message": "Data sent successfully to FTP server",
    "success": "true"
  }
  ```

- Data: The requested data will be served following the consumer's requirements in terms of format and data quality. The current version makes the data available for the consumer using an FTP server.

2.24.4 Information Flow

Obtaining cleansed and preprocessed historical data from storage

The flow shown in Figure 66Figure 66: UML diagram of the process of retrieving data from the storage and serving it to the client. describes a particular scenario in which a cleansed and preprocessed dataset is available in the data storage. In that case, the DC can retrieve the data from the storage, apply the requested filtering rules and format and serve it to the client. This scenario has been implemented in the prototype. To facilitate the management of big datasets, the data is sent to a repository (FTP or SFTP server) and the client receives a notification when the data is available.

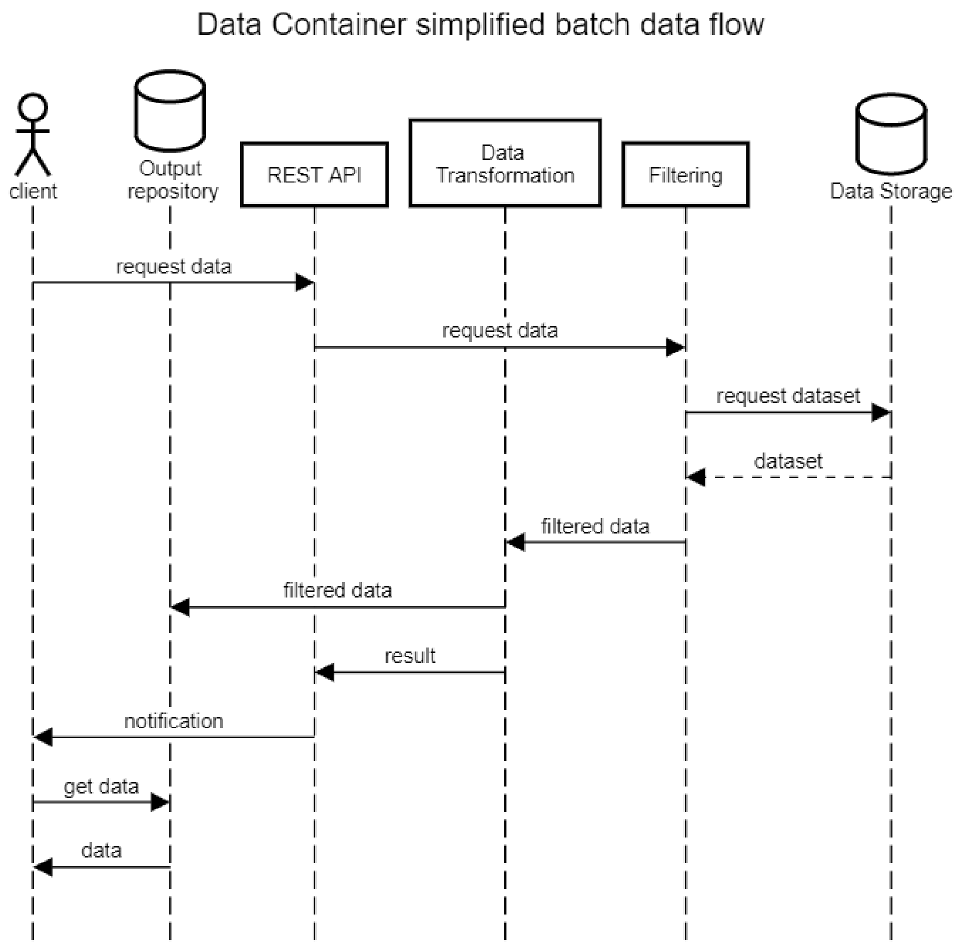


Figure 66: UML diagram of the process of retrieving data from the storage and serving it to the client.

Obtaining historical data from the source

The flow shown in Figure 67 illustrates the overall process. Hence, it includes calls to other blocks of the RE4DY architecture, such as Data Ingestion and Semantic Transformation. To facilitate the management of big datasets, the data can be sent to a repository (such as an FTP or SFTP server). In that case, the DC sends a notification to the client when the data is available.

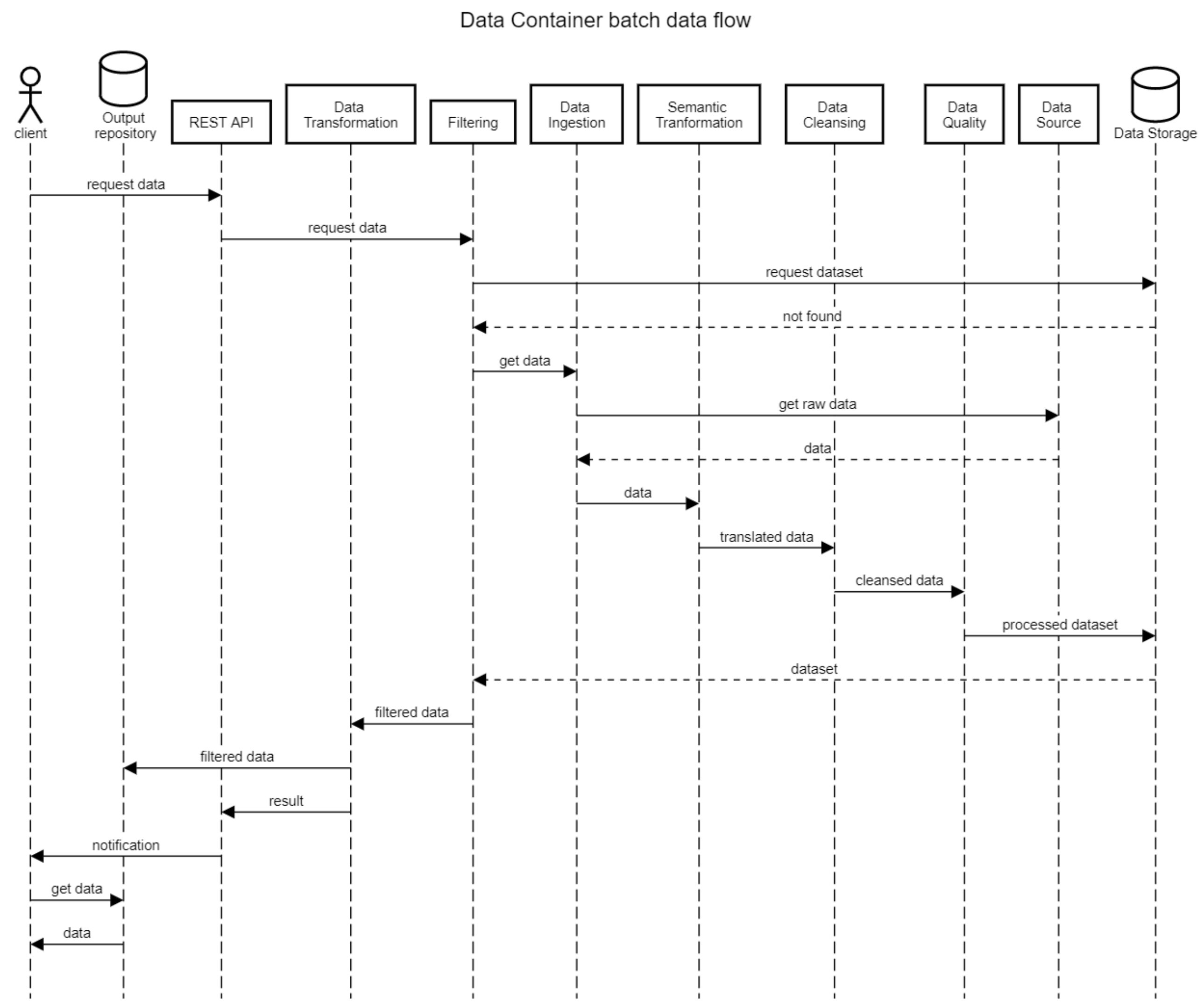


Figure 67: UML diagram of the complete process of retrieving and serving batch data in a generic Data Container.

2.24.5 Internal Architecture

Data Containers expose a REST API to enable access to the data and metadata. The control module manages the DC configuration and ensures that the conditions defined in the DCP are applied. When a request is received, the Data Processing module performs the necessary calls to other components of the RE4DY architecture in order to retrieve the requested data and ensure that the client's requirements are met. This module can also apply filtering rules to select only a subset of the available data. Finally, the Data Transformation module prepares the data and converts it into the format requested by the client.

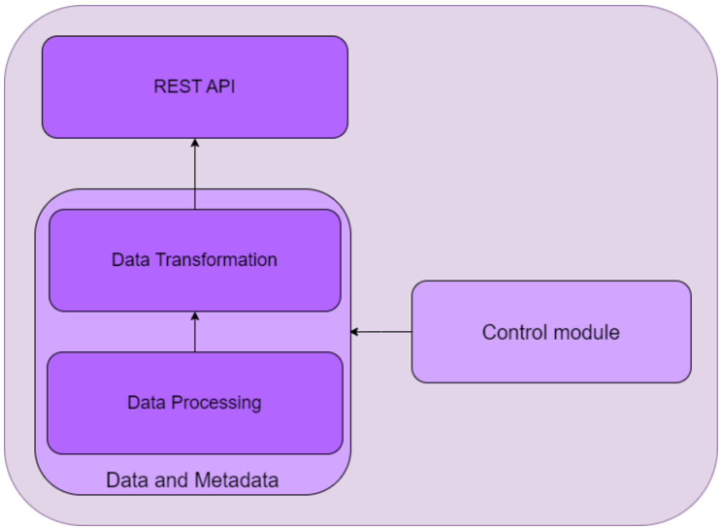


Figure 68: Internal architecture of a generic Data Container.

2.24.6 API

```
•      GET /metadata

   -   Description
This endpoint provides metadata associated to the datasets provided
by the DC. The input parameter "dataset_id" identifies the dataset
for which the metadata is requested.

   -   Input
       dataset_id

   -   Output
{
  "variables": 11,
  "rows": 1,
  "size": 1291,
  "lastUpdate": "2023-06-05T15:31:16Z",
  "variablesNames": [ "var1", "var2", … ]
}

•      POST /vdc

   -   Description
This endpoint is used by the data consumer to request data from the
DC. The input parameter "dataset_id" identifies the dataset. The
parameter "format" defines the selected output format. The JSON
array "rules" defines the filtering rules that will be applied to
the selected dataset. A notification will be sent to the URL
specified in the parameter "notification_url" when the dataset is
available in the output repository (FTP server).

   -   Input
{
  "dataset_id": "FaultySteelPlates",
  "rules": [
        {
```

```
        "name": "test-rule",
        "rule": {
          "subject_column": "TypeOfSteel_A300",
          "operator": "==",
          "object": 1
        }
      }
    ],
    "format": "json",
    "notification_url": "http://example.com/notify "
  }

     -   Output
  {
    "result rows": 777,
    "message": "Request received successfully. Preparing results...",
    "success": true
  }
```

2.24.7 Implementation Technology

Through the Data Containers, it should be possible to define flows that connect different elements of the RE4DY architecture to enable access to the data following the DaaP principles. Thus, the preferred implementation for the DCs is based on flow-oriented programming frameworks such as Apache NiFi or Node-RED. The current implementation has been created using NiFi and is deployed on Docker.

2.24.8 Comments

The current implementation is a prototype. Specific Data Containers will be developed for each use case.

At this point, only the flows for historical data have been defined. The use of DC with near real-time data will be defined in future versions.

2.25 Component Name: 5G User Equipment (UE) Digital Twin or 5G UE AAS

2.25.1 Description

This component represents the digital twin or Asset Administration Shell of a 5G User Equipment. An AAS is a digital model or a set of submodels that provides relevant information and features of an asset. AASs describe the technical functionalities exposed by the assets, ensuring interoperability between systems managing manufacturing processes. Asset Administration Shells (AAS) are then standardized digital representations or digital twins of assets and play a crucial role in the management and administration of assets within manufacturing environments.

The digital twin or AAS of a 5G system Is integrated by two main components: the digital twin of the 5G UE, which is the endpoint of a 5G link, and the digital twin of the 5G network that encompasses all the nodes and functions within the 5G Radio and Core networks. This component focuses on the 5G UE digital twin.

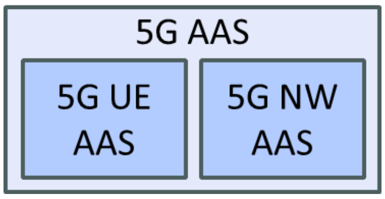


Figure 69: Representation of the main components of a 5G Digital Twin or AAS.

2.25.2 Input

The 5G UE Digital Twin obtains information from the 5G UE. The information obtained shows the current status of the 5G UE and about the different 5G connections established by the UE. In particular, the 5G UE Digital Twin obtains information about technical capabilities of the 5G UE, identification data, network access restrictions, connection status, experienced communication performance, and localization information. The 5G UE Digital Twin uses the interfaces and exposure functions enabled by the 5G systems to make this data available, such as the NEF (Network Exposure Function), and SEAL (Service Enabler Architecture Layer for Verticals) servers.

The 5G UE Digital Twin can also receive input data from other Digital Twins. Specifically, the 5G Network Digital Twin can provide information about the status and operation of the 5G network. Moreover, the Digital Twin of the industrial device to which the UE is related will provide input data about the traffic generated by the industrial application and the status of the device.

2.25.3 Output

The 5G UE Digital Twin exposes and simplifies access to 5G UE data, allowing for easier management and utilization of the 5G technology within manufacturing systems. The 5G

UE Digital Twin structures and organizes the data obtained from the 5G UE and makes it accessible following a standardized approach that facilitates its seamless integration with the digital twin of manufacturing systems.

2.25.4 Information Flow

2.25.4.1 QoS established for a PDU Session

When a new connection or PDU Session is established by a UE in a 5G network, the UE informs the 5G network about the communication requirements of the traffic generated by the industrial application. Based on these communication requirements, the 5G network decides the QoS parameters associated to the PDU session executing several QoS parameters mapping functions at different nodes of the 5G network. The main purpose of these mapping functions is the conversion of QoS parameters from one format to another. This process is depicted in Figure 70 and described below:

1. The application requirements are communicated and stored in the AF (Application Function) using a session description language (SDI), e.g. Session Description Protocol (SDP) or Media Presentation Description (MPD).
2. The AF maps the application specific information into the appropriate QoS format to be understood by the 5G network.
3. The PCF (Policy Control Function) is in charge of mapping the service information received from the AF into QoS parameters (e.g. Guaranteed Bit Rate, Maximum Bit Rate, Allocation and Retention Priority, etc.) authorized to the UE based on the UE subscription. The PCF combines per direction the individual Authorized QoS parameters per flow.
4. The SMF (Session Management Function) then maps the Authorized QoS parameters received from the PCF to Authorized Access-specific QoS parameters, that is for each QoS flow. The Authorized Access-specific QoS parameters are communicated to the UE.

As shown in Figure 70, the UE has information about the QoS parameters related to each connection established. The 5G UE AAS accesses the information about the established connections and stores the QoS parameters related to each connection. The 5G Network Digital Twin will also model the mapping functions executed at each node of the 5G core network. If this is the case, the 5G UE AAS will interact with the 5G Network Digital Twin to model the QoS mapping process in the digital world.

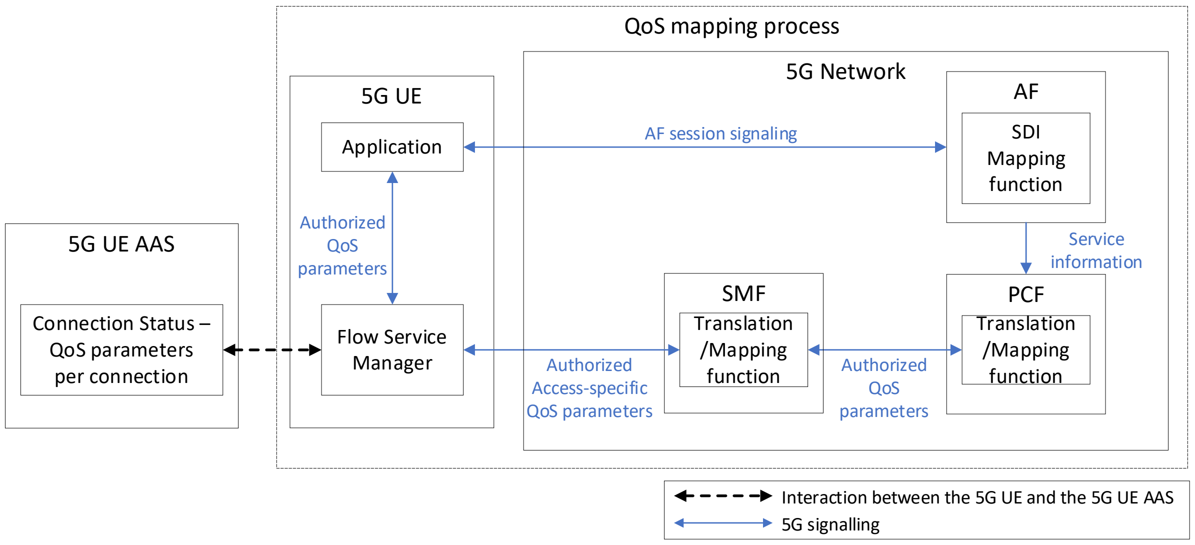

QoS mapping process
5G Network
5G UE
Application
AF
SDI Mapping function
AF session signaling
Authorized QoS parameters
Service information
5G UE AAS
Connection Status – QoS parameters per connection
Flow Service Manager
SMF
Translation /Mapping function
PCF
Translation /Mapping function
Authorized Access-specific QoS parameters
Authorized QoS parameters
Interaction between the 5G UE and the 5G UE AAS
5G signalling

Figure 70: Information exchange between the 5G UE and the 5G UE Digital Twin or AAS.

2.25.4.2 PDU session modification

After a PDU is established, the UE can request a PDU session modification to, for example, request the modification of the QoS parameters related to the PDU session (see Figure 71). To this end, the UE sends a 'PDU session modification request' message to the SMF containing the requested QoS parameters. If the request is accepted, it will be granted only if the network can support the petition. The SMF will then send a 'PDU session modification command' message to the UE, containing the Authorized QoS parameters.

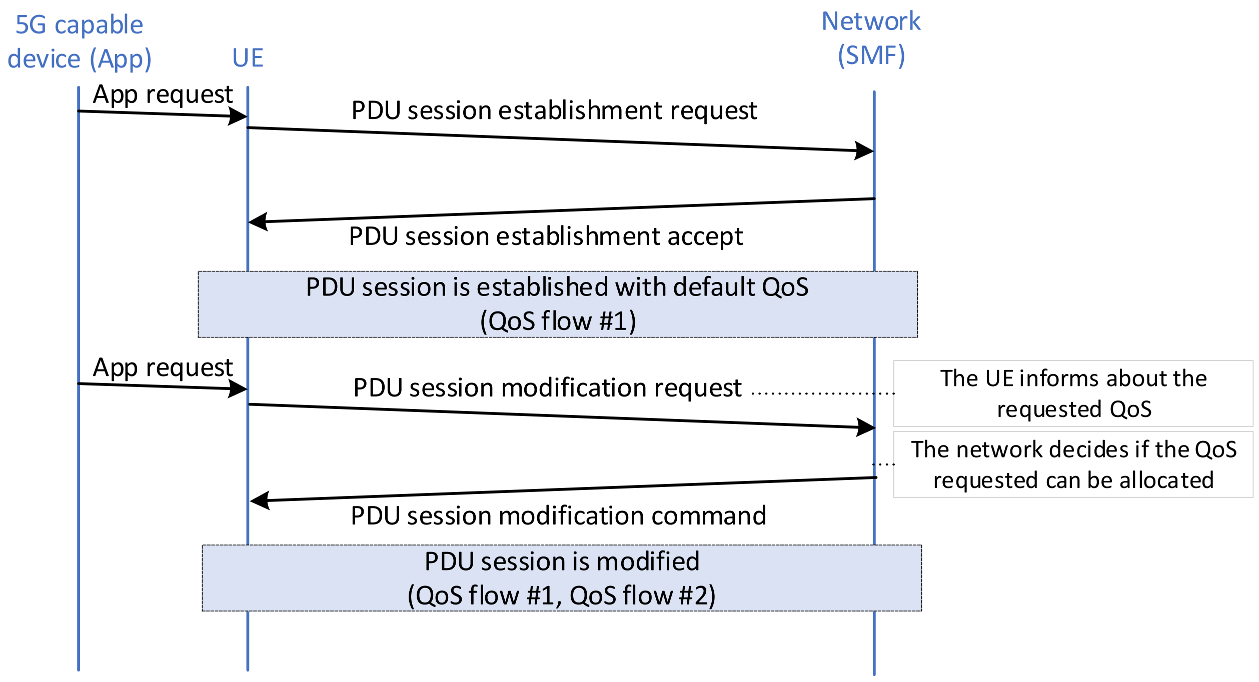

5G capable device (App)
UE
Network (SMF)
App request
PDU session establishment request
PDU session establishment accept
PDU session is established with default QoS (QoS flow #1)
App request
PDU session modification request
The UE informs about the requested QoS
The network decides if the QoS requested can be allocated
PDU session modification command
PDU session is modified (QoS flow #1, QoS flow #2)

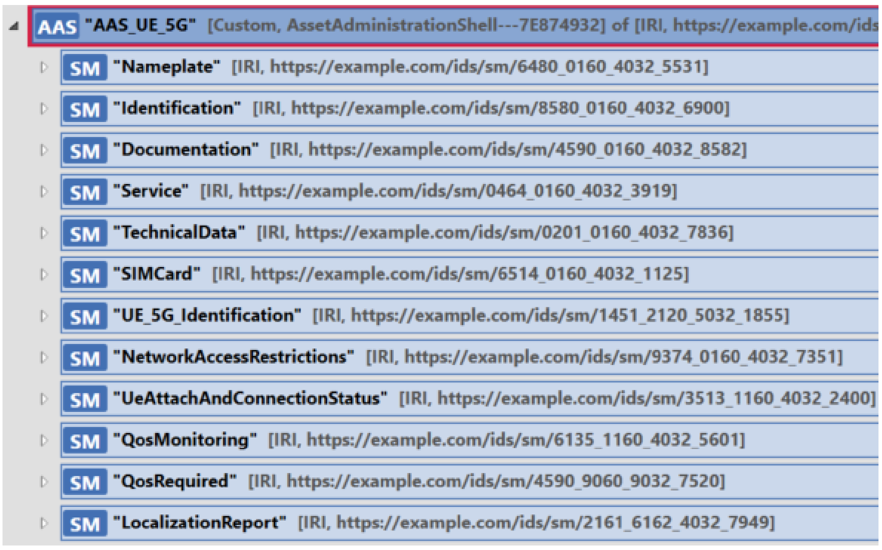
Figure 71: PDU session modification procedure.

This PDU session modification procedure can also be modelled in the digital twin of the 5G network. Before to take an action, it could be interesting to know if the network will be capable to support the new requested QoS. In this case, the digital twin of the 5G capable device will interact with the 5G UE Digital Twin. Upon receiving the request, the 5G UE Digital Twin will process the request and will send a PDU session modification request to the 5G Network Digital Twin which model all the session and QoS management functions. The 5G Network Digital Twin will evaluate the request and will decide if the QoS requested can be allocated. The 5G Network Digital Twin will inform the 5G UE Digital Twin about their decision. By modelling this procedure in the digital world, it is possible to evaluate the results and consequences of the request before taking an action in the physical word.

2.25.5 Internal Architecture

The information and functions within the 5G UE Digital Twin are organized in different submodels. Submodels are technically isolated from each other. Each submodel comprises submodel elements, which serve as the components suitable for describing and distinguishing assets. These submodel elements can be properties, operations, and collections, enabling a hierarchical structure for asset differentiation.

The defined 5G UE Digital Twin relies on a comprehensive understanding of 5G networks, their operation and functions, and 3GPP standards. In addition, the definition of the 5G UE Digital Twin or AAS is done based on the specifications defined by Platform Industry 4.0 for the definition of AASs, and the indications provided by 5G-ACIA. Figure 72 shows the defined 5G UE Digital Twin and the different submodels at which the information is organized. The submodels included in the 5G UE Digital Twin are next presented. Some of these submodels are default submodels that all AASs need to include based on the specifications of Platform Industry 4.0. Other submodels have been defined to include the relevant information, data and functions of a 5G UE.



AAS "AAS_UE_5G" [Custom, AssetAdministrationShell---7E874932] of [IRI, https://example.com/ids

SM "Nameplate" [IRI, https://example.com/ids/sm/6480_0160_4032_5531]

SM "Identification" [IRI, https://example.com/ids/sm/8580_0160_4032_6900]

SM "Documentation" [IRI, https://example.com/ids/sm/4590_0160_4032_8582]

SM "Service" [IRI, https://example.com/ids/sm/0464_0160_4032_3919]

SM "TechnicalData" [IRI, https://example.com/ids/sm/0201_0160_4032_7836]

SM "SIMCard" [IRI, https://example.com/ids/sm/6514_0160_4032_1125]

SM "UE_5G_Identification" [IRI, https://example.com/ids/sm/1451_2120_5032_1855]

SM "NetworkAccessRestrictions" [IRI, https://example.com/ids/sm/9374_0160_4032_7351]

SM "UeAttachAndConnectionStatus" [IRI, https://example.com/ids/sm/3513_1160_4032_2400]

SM "QosMonitoring" [IRI, https://example.com/ids/sm/6135_1160_4032_5601]

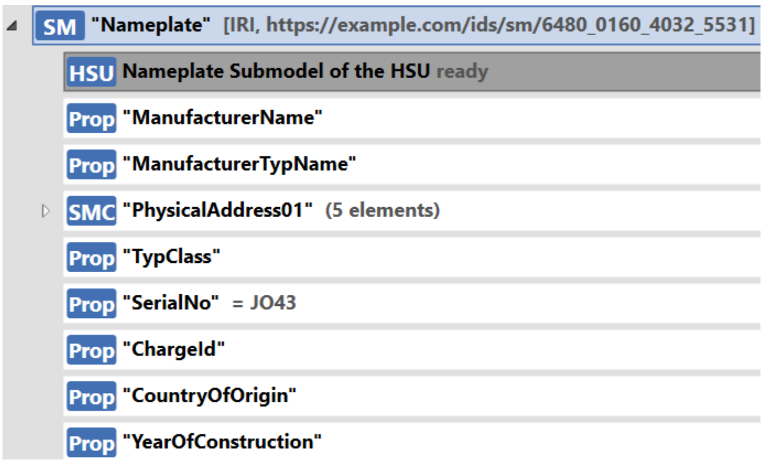
SM "QosRequired" [IRI, https://example.com/ids/sm/4590_9060_9032_7520]

SM "LocalizationReport" [IRI, https://example.com/ids/sm/2161_6162_4032_7949]

Figure 72: Submodels of the 5G UE Digital Twin or AAS.

2.25.5.1 Standardized or default submodels.

- Nameplate submodel provides identifying information about an asset, such as the manufacturer's name, model type, and serial number.



SM "Nameplate" [IRI, https://example.com/ids/sm/6480_0160_4032_5531]

HSU Nameplate Submodel of the HSU ready

Prop "ManufacturerName"

Prop "ManufacturerTypName"

SMC "PhysicalAddress01" (5 elements)

Prop "TypClass"

Prop "SerialNo" = JO43

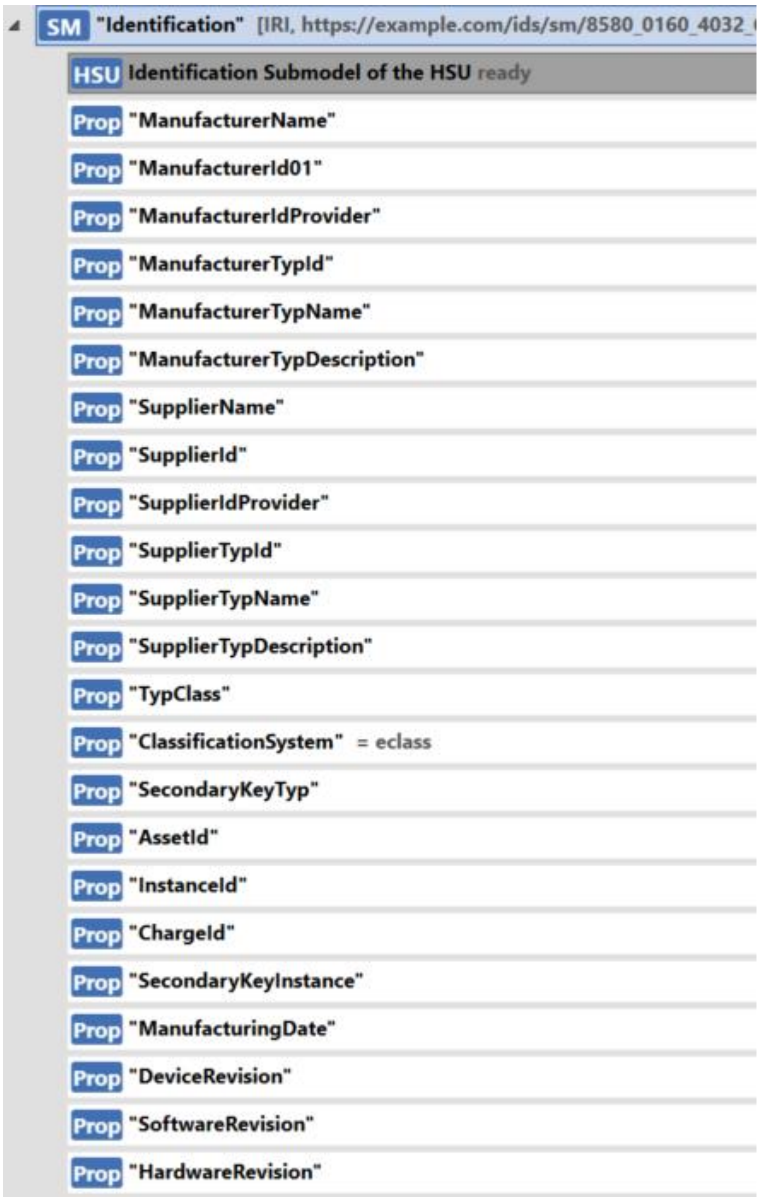
Prop "ChargeId"

Prop "CountryOfOrigin"

Prop "YearOfConstruction"

Figure 73: Nameplate submodel of a 5G UE Digital Twin or AAS.

- Identification submodel is utilized for property recognition. It comprises elements such as the manufacturer's name, supplier's name, asset ID, manufacturing date, device revision, software revision, hardware revision, etc. Despite some overlapping elements with the Nameplate submodel, these serve different functions in varied scenarios.



SM "Identification" [IRI, https://example.com/ids/sm/8580_0160_4032_

HSU Identification Submodel of the HSU ready

Prop "ManufacturerName"

Prop "ManufacturerId01"

Prop "ManufacturerIdProvider"

Prop "ManufacturerTypId"

Prop "ManufacturerTypName"

Prop "ManufacturerTypDescription"

Prop "SupplierName"

Prop "SupplierId"

Prop "SupplierIdProvider"

Prop "SupplierTypId"

Prop "SupplierTypName"

Prop "SupplierTypDescription"

Prop "TypClass"

Prop "ClassificationSystem" = eclass

Prop "SecondaryKeyTyp"

Prop "AssetId"

Prop "InstanceId"

Prop "ChargeId"

Prop "SecondaryKeyInstance"

Prop "ManufacturingDate"

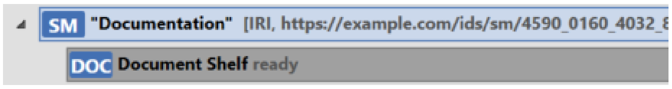
Prop "DeviceRevision"

Prop "SoftwareRevision"

Prop "HardwareRevision"

Figure 74: Identification submodel of a 5G UE Digital Twin or AAS.

- Documentation submodel organizes and categorizes pertinent documents, making it easier to locate them. This encompasses items like data sheets, maintenance manuals, and user guides.

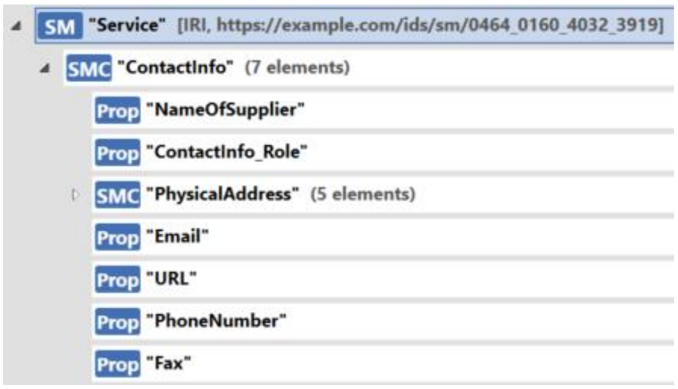


SM "Documentation" [IRI, https://example.com/ids/sm/4590_0160_4032_8

DOC Document Shelf ready

Figure 75: Documentation submodel of a 5G UE Digital Twin or AAS.

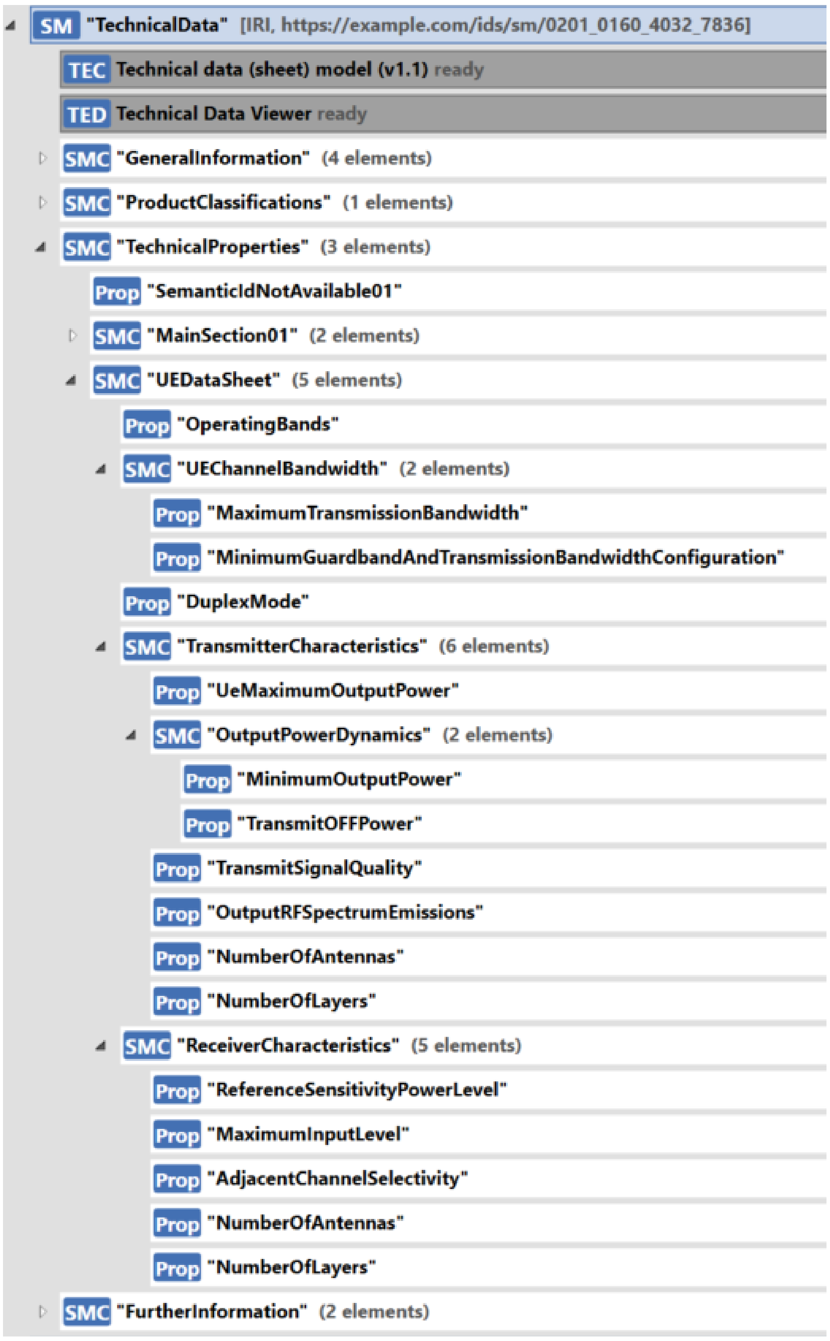
- Service submodel contains support and maintenance information. It offers contact details for necessary support. Elements within this submodel may include supplier name, contact info role, email, or phone number.



SM "Service" [IRI, https://example.com/ids/sm/0464_0160_4032_3919]
SMC "ContactInfo" (7 elements)
Prop "NameOfSupplier"
Prop "ContactInfo_Role"
SMC "PhysicalAddress" (5 elements)
Prop "Email"
Prop "URL"
Prop "PhoneNumber"
Prop "Fax"

Figure 76: Service submodel of a 5G UE Digital Twin or AAS.

- TechnicalData submodel aims to provide interoperable technical data that describes the asset of the respective Asset Administration Shell. The "TechnicalData" submodel has been extended to include specific technical characteristics of a 5G UE, such as the operating bands, channel bandwidth, duplex mode, and technical characteristics of the transmitter and receiver.



```
SM "TechnicalData" [IRI, https://example.com/ids/sm/0201_0160_4032_7836]
  TEC Technical data (sheet) model (v1.1) ready
  TED Technical Data Viewer ready
  SMC "GeneralInformation" (4 elements)
  SMC "ProductClassifications" (1 elements)
  SMC "TechnicalProperties" (3 elements)
    Prop "SemanticIdNotAvailable01"
    SMC "MainSection01" (2 elements)
    SMC "UEDataSheet" (5 elements)
      Prop "OperatingBands"
      SMC "UEChannelBandwidth" (2 elements)
        Prop "MaximumTransmissionBandwidth"
        Prop "MinimumGuardbandAndTransmissionBandwidthConfiguration"
      Prop "DuplexMode"
      SMC "TransmitterCharacteristics" (6 elements)
        Prop "UeMaximumOutputPower"
        SMC "OutputPowerDynamics" (2 elements)
          Prop "MinimumOutputPower"
          Prop "TransmitOFFPower"
        Prop "TransmitSignalQuality"
        Prop "OutputRFSpectrumEmissions"
        Prop "NumberOfAntennas"
        Prop "NumberOfLayers"
      SMC "ReceiverCharacteristics" (5 elements)
        Prop "ReferenceSensitivityPowerLevel"
        Prop "MaximumInputLevel"
        Prop "AdjacentChannelSelectivity"
        Prop "NumberOfAntennas"
        Prop "NumberOfLayers"
  SMC "FurtherInformation" (2 elements)
```

Figure 77: TechnicalData submodel of a 5G UE Digital Twin or AAS.

2.25.5.2 5G UE related submodels

- SIMCard submodel includes all the information related to the SIM (Subscriber Identity Module) card used by the UE. The SIMCard submodel contains the IMSI (International Mobile Subscriber Identity), ICCID (Integrated Circuit Card ID), PIN (Personal Identification Number). The submodel also includes the Service Provider Name (SPN) that identifies the name of the mobile operator providing the service, and the Authentication Key that is a unique value for each SIM card that authenticates the UE in the network.

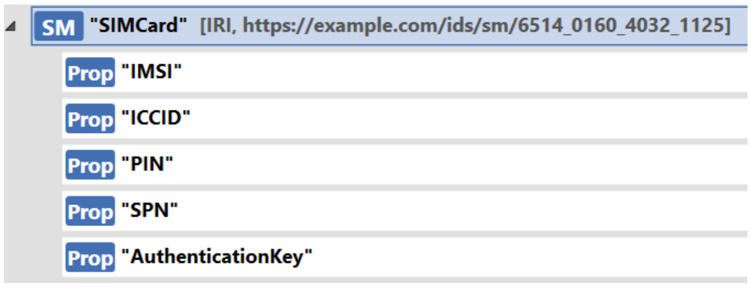

SM "SIMCard" [IRI, https://example.com/ids/sm/6514_0160_4032_1125]
Prop "IMSI"
Prop "ICCID"
Prop "PIN"
Prop "SPN"
Prop "AuthenticationKey"

Figure 78: SIMCard submodel of a 5G UE Digital Twin or AAS.

- UE_5G_Identification submodel contains data that is used to uniquely identify a UE within the 5G network such as the Permanent Equipment Identifier, UE identity GPSI, the certificate of authentication and the certificate status.

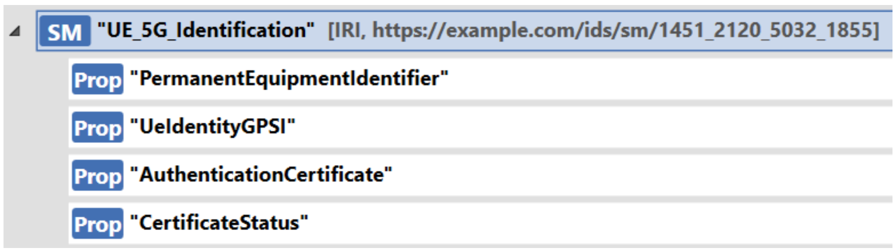

SM "UE_5G_Identification" [IRI, https://example.com/ids/sm/1451_2120_5032_1855]
Prop "PermanentEquipmentIdentifier"
Prop "UeIdentityGPSI"
Prop "AuthenticationCertificate"
Prop "CertificateStatus"

Figure 79: UE_5G_Identification submodel of a 5G UE Digital Twin or AAS.

- NetworkAccessRestrictions submodel includes physical and logical access restrictions to the network. It contains a list of cells and slices to which the UE can access.

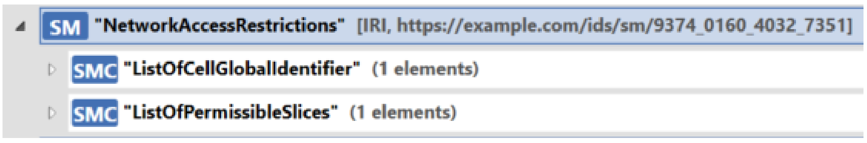
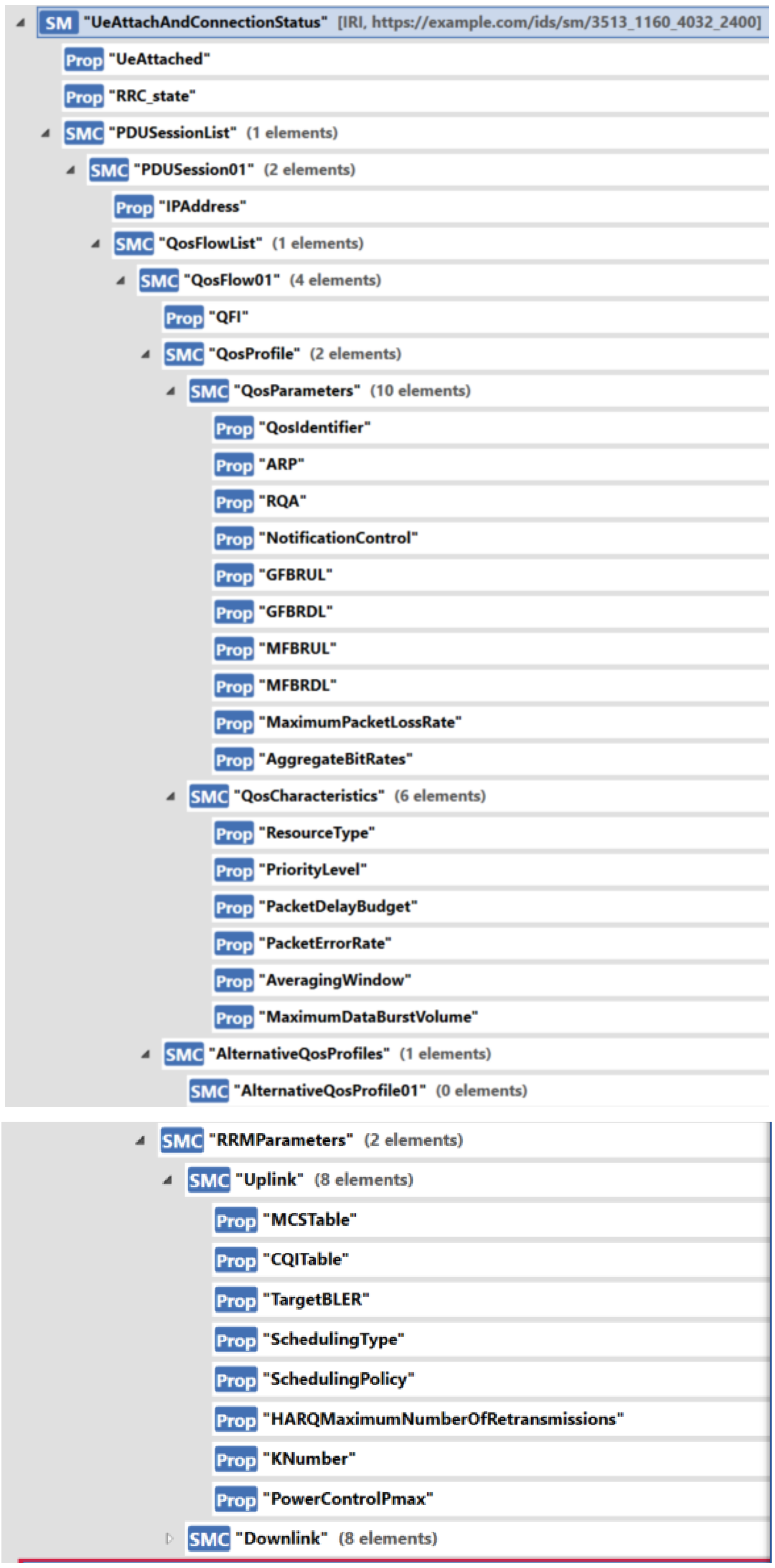

SM "NetworkAccessRestrictions" [IRI, https://example.com/ids/sm/9374_0160_4032_7351]
SMC "ListOfCellGlobalIdentifier" (1 elements)
SMC "ListOfPermissibleSlices" (1 elements)

Figure 80: NetworkAccessRestrictions submodel of a 5G UE Digital Twin or AAS.

- UeAttachAndConnectionStatus submodel contains information about the connection status of the UE and a list of the connections or Protocol Data Unit (PDU) sessions established with the UE (acting as a source or destination of the communication). For each PDU session, the submodel stores the information about the QoS parameters that need to be supported, and the Radio Resource Management (RRM) parameters that are used to meet the established QoS.

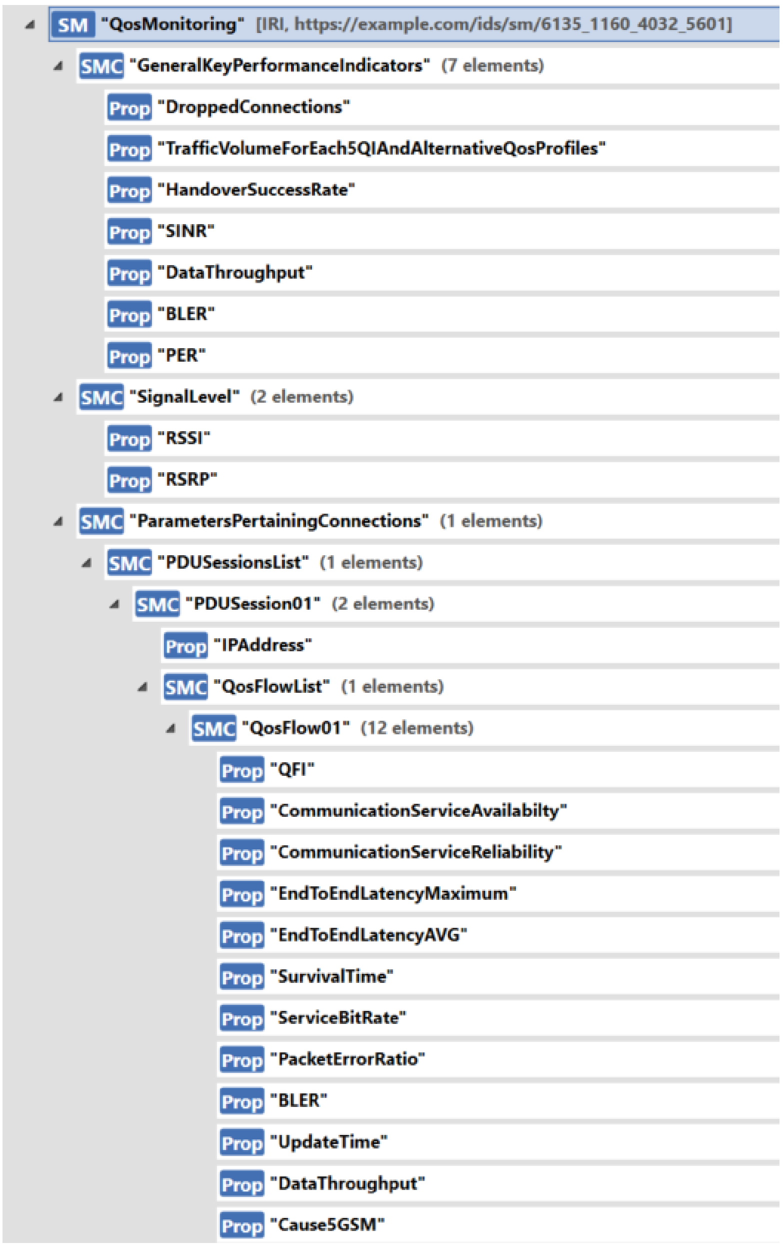


```
SM "UeAttachAndConnectionStatus" [IRI, https://example.com/ids/sm/3513_1160_4032_2400]
  Prop "UeAttached"
  Prop "RRC_state"
  SMC "PDUSessionList" (1 elements)
    SMC "PDUSession01" (2 elements)
      Prop "IPAddress"
      SMC "QosFlowList" (1 elements)
        SMC "QosFlow01" (4 elements)
          Prop "QFI"
          SMC "QosProfile" (2 elements)
            SMC "QosParameters" (10 elements)
              Prop "QosIdentifier"
              Prop "ARP"
              Prop "RQA"
              Prop "NotificationControl"
              Prop "GFBRUL"
              Prop "GFBRDL"
              Prop "MFBRUL"
              Prop "MFBRDL"
              Prop "MaximumPacketLossRate"
              Prop "AggregateBitRates"
            SMC "QosCharacteristics" (6 elements)
              Prop "ResourceType"
              Prop "PriorityLevel"
              Prop "PacketDelayBudget"
              Prop "PacketErrorRate"
              Prop "AveragingWindow"
              Prop "MaximumDataBurstVolume"
          SMC "AlternativeQosProfiles" (1 elements)
            SMC "AlternativeQosProfile01" (0 elements)

          SMC "RRMParameters" (2 elements)
            SMC "Uplink" (8 elements)
              Prop "MCSTable"
              Prop "CQITable"
              Prop "TargetBLER"
              Prop "SchedulingType"
              Prop "SchedulingPolicy"
              Prop "HARQMaximumNumberOfRetransmissions"
              Prop "KNumber"
              Prop "PowerControlPmax"
            SMC "Downlink" (8 elements)
```

Figure 81: UeAttachAndConnectionStatus submodel of a 5G UE Digital Twin or AAS.

- QosMonitoring submodel contains data about the performance experienced by the UE. In particular, it shows key performance indicators (KPIs) that provide information about the general performance experienced by the UE, and also KPIs related to the particular performance achieved in each connection or PDU session established by the UE. This submodel also contains a list of the connectivity-related events to which the UE is subscribed in the 5G network. Some examples of the events to which the UE can subscribe are: maximum latency exceeded, minimum service bit rate not achieved, communication service availability fell below the requested value, connectivity lost/ re-established. The model also shows and saves the registered events as a result of the subscriptions.



```
SM "QosMonitoring" [IRI, https://example.com/ids/sm/6135_1160_4032_5601]
  SMC "GeneralKeyPerformanceIndicators" (7 elements)
    Prop "DroppedConnections"
    Prop "TrafficVolumeForEach5QIAndAlternativeQosProfiles"
    Prop "HandoverSuccessRate"
    Prop "SINR"
    Prop "DataThroughput"
    Prop "BLER"
    Prop "PER"
  SMC "SignalLevel" (2 elements)
    Prop "RSSI"
    Prop "RSRP"
  SMC "ParametersPertainingConnections" (1 elements)
    SMC "PDUSessionsList" (1 elements)
      SMC "PDUSession01" (2 elements)
        Prop "IPAddress"
        SMC "QosFlowList" (1 elements)
          SMC "QosFlow01" (12 elements)
            Prop "QFI"
            Prop "CommunicationServiceAvailabilty"
            Prop "CommunicationServiceReliability"
            Prop "EndToEndLatencyMaximum"
            Prop "EndToEndLatencyAVG"
            Prop "SurvivalTime"
            Prop "ServiceBitRate"
            Prop "PacketErrorRatio"
            Prop "BLER"
            Prop "UpdateTime"
            Prop "DataThroughput"
            Prop "Cause5GSM"
```

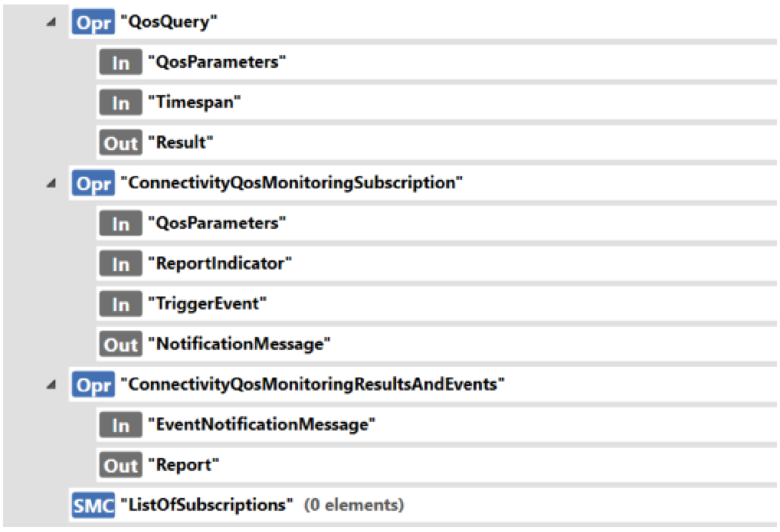

Opr "QosQuery"
In "QosParameters"
In "Timespan"
Out "Result"
Opr "ConnectivityQosMonitoringSubscription"
In "QosParameters"
In "ReportIndicator"
In "TriggerEvent"
Out "NotificationMessage"
Opr "ConnectivityQosMonitoringResultsAndEvents"
In "EventNotificationMessage"
Out "Report"
SMC "ListOfSubscriptions" (0 elements)

Figure 82: QosMonitoring submodel of a 5G UE Digital Twin or AAS.

- LocalizationReport submodel includes a list of localization events to which the UE is subscribe in the 5G network, as for example. In addition, this submodel provides information about localization events to which the UE is subscribed.

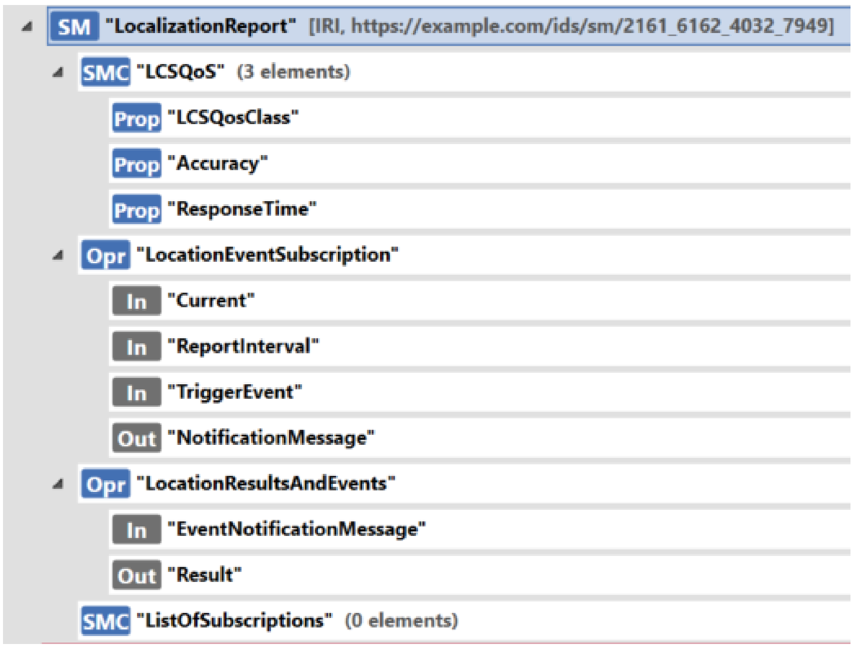

SM "LocalizationReport" [IRI, https://example.com/ids/sm/2161_6162_4032_7949]
SMC "LCSQoS" (3 elements)
Prop "LCSQosClass"
Prop "Accuracy"
Prop "ResponseTime"
Opr "LocationEventSubscription"
In "Current"
In "ReportInterval"
In "TriggerEvent"
Out "NotificationMessage"
Opr "LocationResultsAndEvents"
In "EventNotificationMessage"
Out "Result"
SMC "ListOfSubscriptions" (0 elements)

Figure 83: LocalizationReport submodel of a 5G UE Digital Twin or AAS.

2.25.6 API

The 5G UE Digital Twin is one of the two components that integrate a 5G Digital Twin. The second component is the 5G Network Digital Twin that digital models all the components and functions of the 5G Radio and Core networks. In this context, the 5G Digital Twin closely interacts with the 5G Network AAS to form a robust digital twin of a 5G network.

On the other hand, the 5G UE Digital Twin will closely interact with the digital twin of the industrial 5G-enabled device where the 5G UE is integrated. The digital twin of the industrial 5G-enabled device will provide information about the traffic generated by the industrial application and the status of the device. The 5G UE Digital Twin could provide information about the status of the communications to the 5G-enabled device digital twin in order to take any preventive action.

The interfaces necessary to interact with other digital twins or AASs (with the Digital Twin of the 5G network part and the Digital Twin of industrial devices) are being currently designed. Following the specifications of Platform Industry 4.0, the interfaces are being implemented using OPC-UA following a Client-Server communication approach.

2.25.7 Implementation Technology

The 5G UE Digital Twin or AAS has been developed following the specifications provided by Platform Industry 4.0 for the development of AASs. The first version of the developed 5G UE Digital Twin or AAS focuses on the design of a passive Digital Twin or AAS that only contains documents and static information. To this end, the 5G UE Digital Twin or AAS has been implemented using the AASX Package Explorer software. This software is developed by Platform Industry 4.0 and employs a graphical interface for creating, editing, and viewing AAS in adherence to the standardized specifications of Platform Industry 4.0. It primarily focuses on creating passive AAS with the correct structure in a static manner.

We are currently working on providing dynamism to the 5G UE Digital Twin and achieving an active AAS that can exchange information with others AAS. To this end, we are using Python-based programming tools (in particular, PyI40AAS and basyx-python-sdk libraries) to add active functionalities to the 5G UE AAS

2.25.8 Comments

This component provides the general description of a 5G UE Digital Twin or AAS and describes the different submodels and how the data should be organized to make it accessible to other digital twins or applications. It is important to highlight that the 5G UE Digital Twin or AAS needs to be specifically configured based on the requirements of the current deployment scenario.

2.26 Component Name: DIDI (Dataspace for Industrial Data Intelligence) Data Marketplace

2.26.1 Description

Atos' DIDI Data Marketplace solution leverages on the power of Gaia-X Data Spaces architecture and open-source components, and provides a secure, interoperable, and highly efficient environment for organizations to seamlessly exchange, store, and manage data across a wide range of industries[6]. With advanced features such as data sovereignty, trustworthiness, and data sharing capabilities at its core, DIDI Data Marketplace solution empowers businesses to unlock new opportunities, drive innovation, and foster collaboration while maintaining strict data privacy and security standards. It serves as a versatile platform for companies seeking to harness the full potential of their data, enabling them to thrive in an increasingly data-driven world.

2.26.2 Internal Architecture

The current implementation of DIDI Data Marketplace is composed, as shown in Figure 84, of the following GAIA-X components:

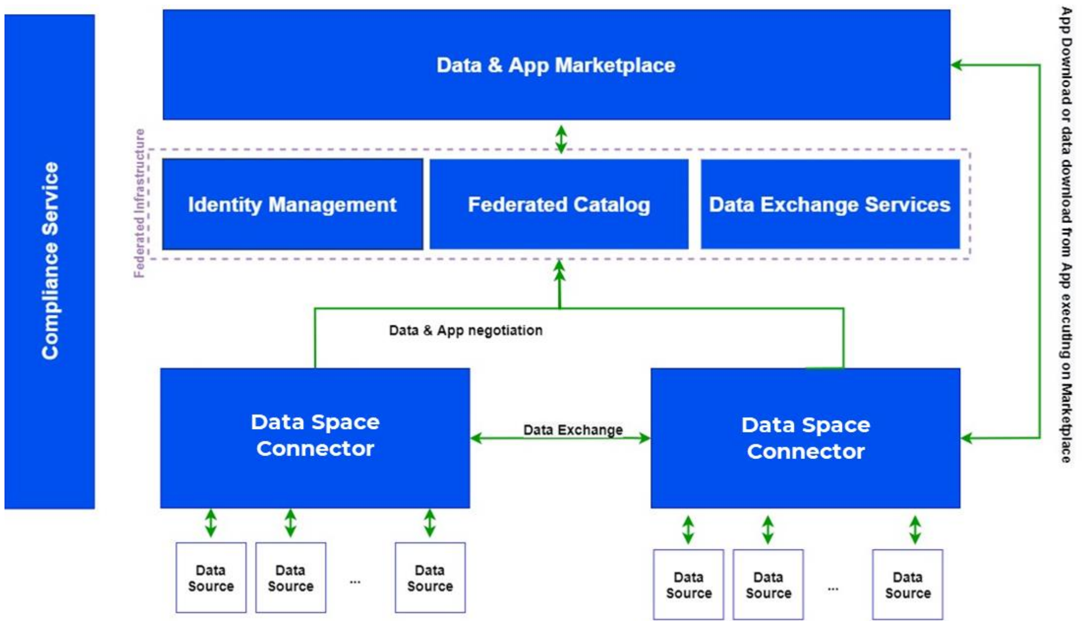


Figure 84: DIDI Data Marketplace High-level Architecture

[6] Note: The 'Dataspace for Industrial Data Intelligence (DIDI) Data Marketplace' asset represents the evolution of former Atos' asset 'AGORA Data Marketplace'.

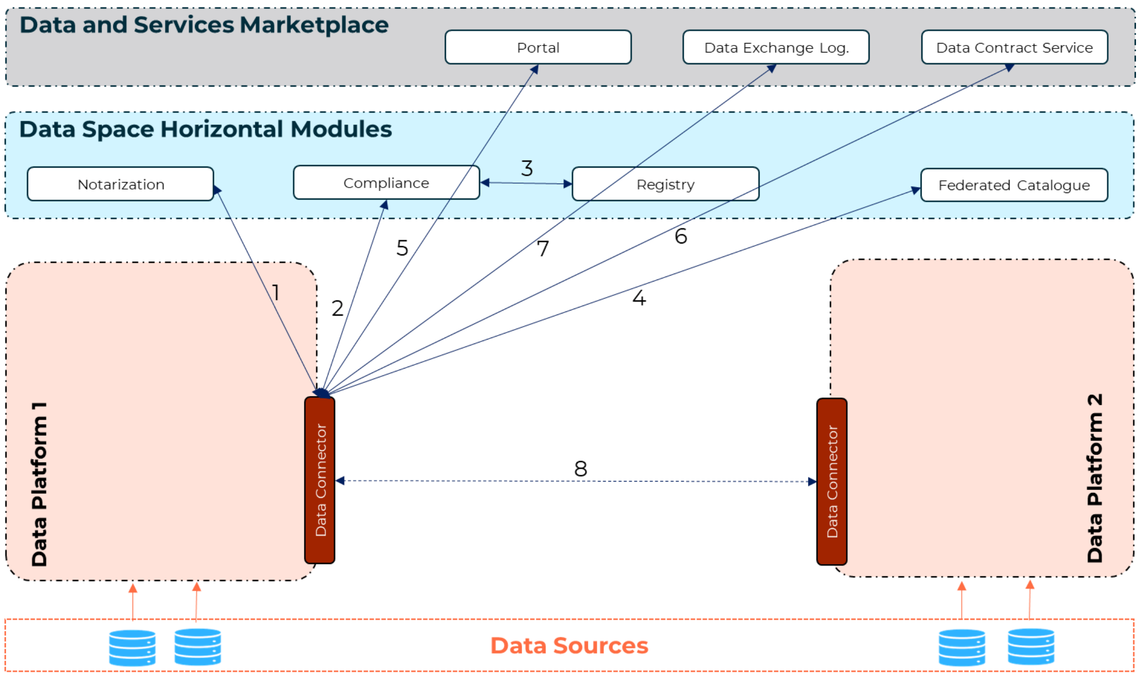


Figure 85: DIDI Data Marketplace High-level Architecture (sequence of interactions between the Data Connector and the rest of components)

<u>Data Space Horizontal Modules:</u>

- *Notarization Service*: it is a tool used to return a Verified Credential (VC) once the API successfully verifies the registration numbers provided by the participant.
- *Compliance Service*: It ensures that the required level of information for the users to take educated decisions is available and the information is verified or verifiable. This service is implemented via the Gaia-X Digital Clearing House, which runs services of the Gaia-X Framework necessary to achieve compliance. It integrates to external TA (Trust Anchors), Identity Verification (like eIDAS), and other TDS (Trusted Data Sources).
- *Registry Service*: it can verify X.509 certificates and public keys of a set of Trust Service Providers according to the Trust Framework document. The module offers the basis for the proof of compliance issued by the Compliance Service and the basis to validate the chain of trust for the Compliance Service.
- *Federated Catalogue:* it is a central component that facilitates the discovery, exchange, and usage of data and services within the GAIA-X ecosystem. The Catalogue Service essentially serves as a registry or index, providing metadata (self-descriptions) and information about available data, services, and resources. The Catalogue Service incorporates the concept of self-descriptions by allowing entities to register and publish metadata about themselves. Thus, entities within the GAIA-X ecosystem, such as data sources or services, can use the Catalogue Service to provide detailed self-descriptions. This information helps users discover and understand the available data and services.

Data and Services Marketplace:

- *Data Contract Service:* it enables and governs data-sharing agreements between different participants in the GAIA-X ecosystem. This includes businesses, organizations, and other entities that contribute, share, or consume data and services.
- *Data Exchange Logging Service:* log and monitor activities related to the exchange of data within the GAIA-X ecosystem. It helps in tracking events and transactions associated with data sharing, providing an audit trail for accountability and compliance purposes. It contributes to regulatory compliance by providing a mechanism for organizations to demonstrate adherence to data protection, privacy, and other relevant standards.
- *Portal*: The frontend provides a user-friendly interface that allows participants to interact with the DIDI Data Marketplace. This includes data providers, data consumers, and any other stakeholders in the ecosystem. Among the features it provides we can find:
    - *Data Discovery*: Users can search and discover available datasets through the frontend. This may involve browsing categories, using search filters, and accessing detailed information about each dataset.
    - *Data Listings*: The frontend displays listings for various datasets, including metadata such as data source, format, size, and any other relevant information. This helps users make informed decisions about which datasets to explore or acquire.
    - *Authentication and Authorization*: The frontend includes features for user authentication and authorization. Participants may need to log in to access certain features, and permissions are enforced based on roles and agreements established in the GAIA-X ecosystem.
    - *Data Contracts and Agreements*: Participants can use the frontend to review and negotiate data-sharing contracts and agreements. This may include specifying terms such as data usage, access controls, and any other conditions agreed upon by the parties involved.
    - *Integration with Catalogue and Logging Services*: The frontend integrates with other GAIA-X services, such as the Catalogue Service and Data Exchange Logging Service, to ensure that data listings are accurate and up-to-date and that data exchange activities are appropriately logged.
    - *Transaction and Payment Management*: If the marketplace involves transactions or payments, the frontend may include features for managing these processes. This could include payment gateways, transaction histories, and billing information.

Data Exchange Services: These services are designed to facilitate secure data exchange and interoperability between different data platforms and providers. They may include data integration, transformation, and messaging services:

- *Data Connector*: it is designed to promote interoperability by providing a standardized mechanism for connecting and exchanging data between different participants in the GAIA-X ecosystem. Besides, it adheres to GAIA-X principles of data sovereignty and security, ensuring that data transfers and integrations comply with privacy regulations and security standards.

2.26.3 Information Flow

This section outlines the basic interaction among various participants and the DIDI Data Marketplace components.

Participants, including Providers and Consumers within the ecosystem, are identified and comprehensively described through valid Self-Descriptions[7], created either before or during the onboarding process. Providers articulate their Service Offerings and make them available in the Federated Catalogue. In parallel, Consumers explore Service Offerings in Gaia-X Catalogues coordinated by Federators and the Gaia-X Registry. Once a Consumer identifies a suitable Service Offering in a Gaia-X Catalogue, Contract negotiation between Provider and Consumer establishes the specific conditions for providing the Service Instance.

The subsequent diagram (Figure 86) illustrates the general workflow for Gaia-X service provisioning and consumption processes.

[7] https://gaia-x.eu/wp-content/uploads/2022/08/SSI_Self_Description_EN_V3.pdf

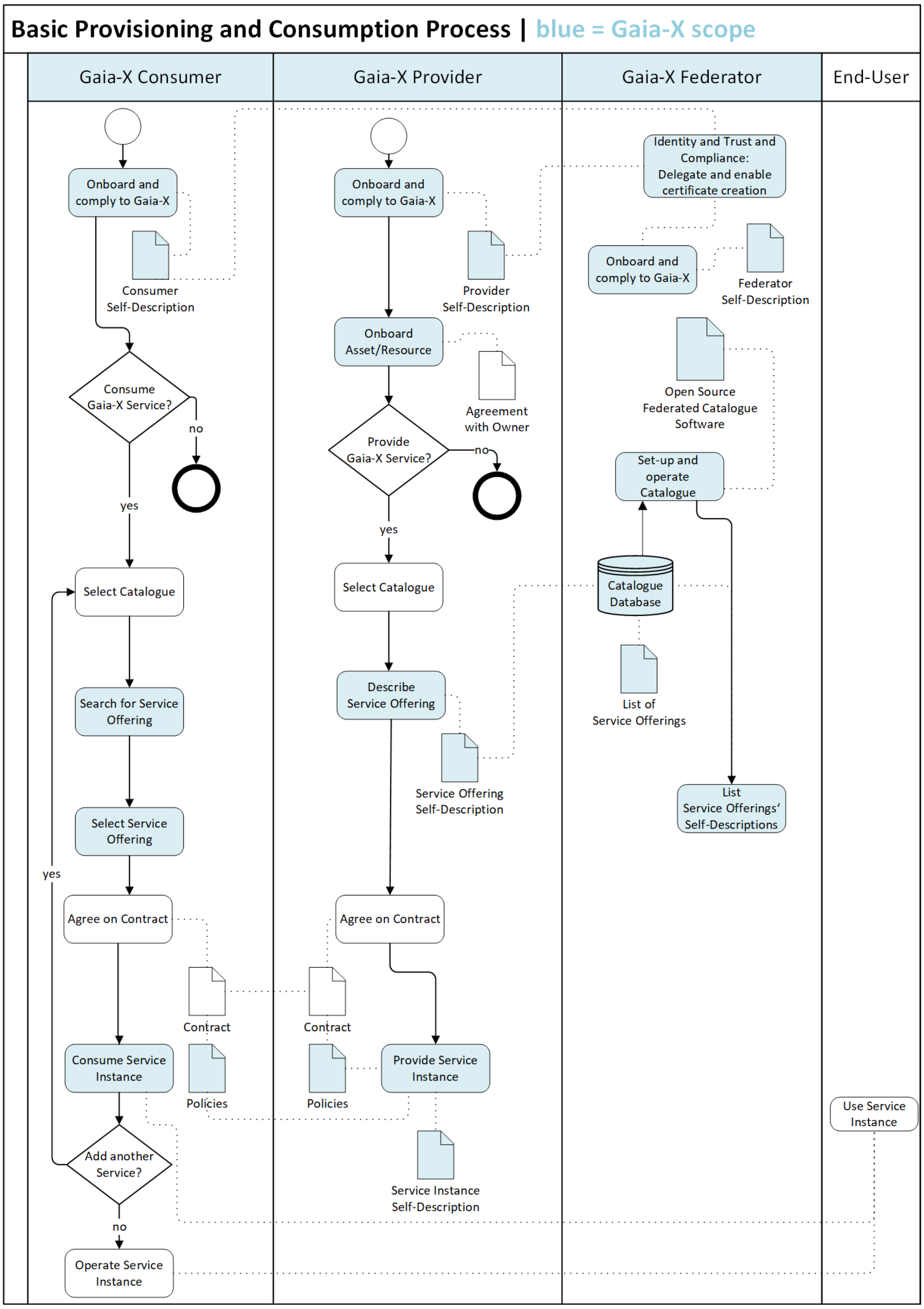

Basic Provisioning and Consumption Process | blue = Gaia-X scope
Gaia-X Consumer
Gaia-X Provider
Gaia-X Federator
End-User
Onboard and comply to Gaia-X
Consumer Self-Description
Consume Gaia-X Service?
no
yes
Select Catalogue
Search for Service Offering
Select Service Offering
yes
Agree on Contract
Contract
Consume Service Instance
Policies
Add another Service?
no
Operate Service Instance
Onboard and comply to Gaia-X
Provider Self-Description
Onboard Asset/Resource
Agreement with Owner
Provide Gaia-X Service?
no
yes
Select Catalogue
Describe Service Offering
Service Offering Self-Description
Agree on Contract
Contract
Policies
Provide Service Instance
Service Instance Self-Description
Identity and Trust and Compliance: Delegate and enable certificate creation
Onboard and comply to Gaia-X
Federator Self-Description
Open Source Federated Catalogue Software
Set-up and operate Catalogue
Catalogue Database
List of Service Offerings
List Service Offerings' Self-Descriptions
Use Service Instance

Figure 86: Basic Provisioning and Consumption Process (source GAIA-X)

For the sake of clarity, some of the flows in the above diagram are broken down and depicted in more detail in separate interaction diagrams illustrating the processes and the information flow among the DIDI Data Marketplace components:

Onboarding process

The onboarding process for registering Participants and Consumers involves several key steps to ensure a smooth and secure integration into the GAIA-X ecosystem. Participants, both Providers and Consumers, initiate the onboarding process by creating a comprehensive Self-Description. This Self-Description document provides essential information about the entity, including identification details, capabilities, and data-related specifications, and it serves as a foundational element for interactions within the GAIA-X ecosystem.

The Self-Description undergoes a validation process to ensure its accuracy and compliance with GAIA-X standards. This step helps maintain the integrity of the information provided by the Participant or Consumer during the onboarding process.

Approved Participants and Consumers receive the necessary credentials and access rights to engage within the GAIA-X ecosystem. This may include authentication tokens, keys, or other secure means of access.

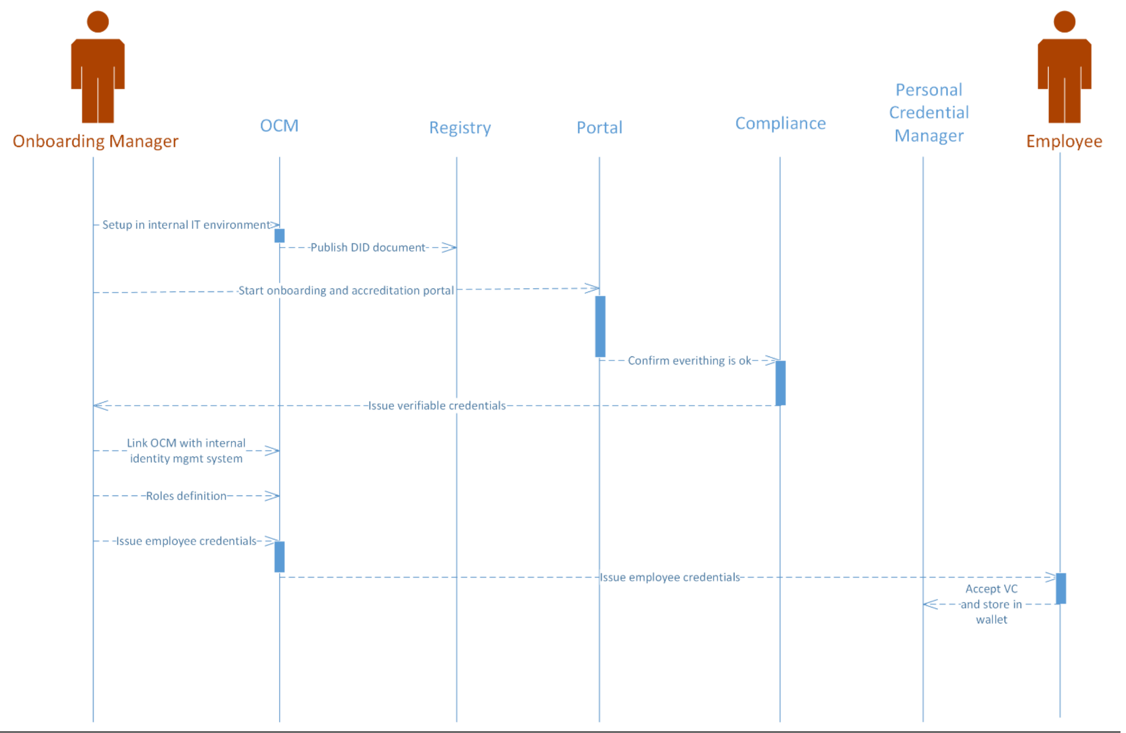

Onboarding Manager
OCM
Registry
Portal
Compliance
Personal Credential Manager
Employee
Setup in internal IT environment
Publish DID document
Start onboarding and accreditation portal
Confirm everithing is ok
Issue verifiable credentials
Link OCM with internal identity mgmt system
Roles definition
Issue employee credentials
Issue employee credentials
Accept VC and store in wallet

Figure 87: Onboarding Process

Federated Catalogue

- Onboarding a new Participant in the Catalogue
  This flow describes the onboarding of a new Participant to the catalogue.

  Once the Approved Participants (in this case, the Providers) receive the necessary credentials and access rights to engage within the GAIA-X ecosystem they can define their Service Offerings. This involves specifying the types of services or data they intend to make available within the GAIA-X Catalogue.

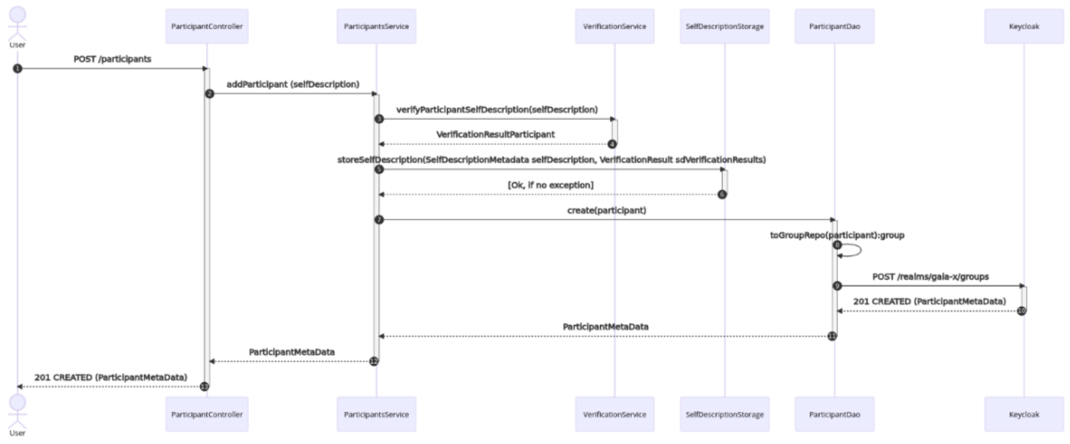

User
ParticipantController
ParticipantsService
VerificationService
SelfDescriptionStorage
ParticipantDao
Keycloak
POST /participants
addParticipant (selfDescription)
verifyParticipantSelfDescription(selfDescription)
VerificationResultParticipant
storeSelfDescription(SelfDescriptionMetadata selfDescription, VerificationResult sdVerificationResults)
[Ok, if no exception]
create(participant)
toGroupRepo(participant):group
POST /realms/gaia-x/groups
201 CREATED (ParticipantMetaData)
ParticipantMetaData
ParticipantMetaData
201 CREATED (ParticipantMetaData)

Figure 88: Onboarding a new Participant (source: GAIA-X[8])

- Adding a Self-Description for an Offering
  The process begins with a Provider who intends to add a new offering to the GAIA-X Catalogue.

  1. Authentication and Authorization: The Provider authenticates their identity using the appropriate credentials. The Catalogue service checks the Provider's authorization level to ensure they have the necessary permissions to add a new Self-Description for the offering.
  2. Create or Update Self-Description: The Provider creates or updates a Self-Description for the offering. This Self-Description includes crucial information such as identification details, capabilities, data specifications, and any other relevant metadata.
  3. Validation: The Catalogue service may perform validation checks on the submitted Self-Description to ensure it meets GAIA-X standards and is consistent with the required format.
  4. Submission to Catalogue: Once the Self-Description is complete and validated, the Provider submits it to the GAIA-X Catalogue. The Catalogue service records the new or updated Self-Description in its database.

[8] https://gaia-x.gitlab.io/data-infrastructure-federation-services/cat/architecture-document/images/diag-a12195ee3006b3f0ab733e721e31291f.png

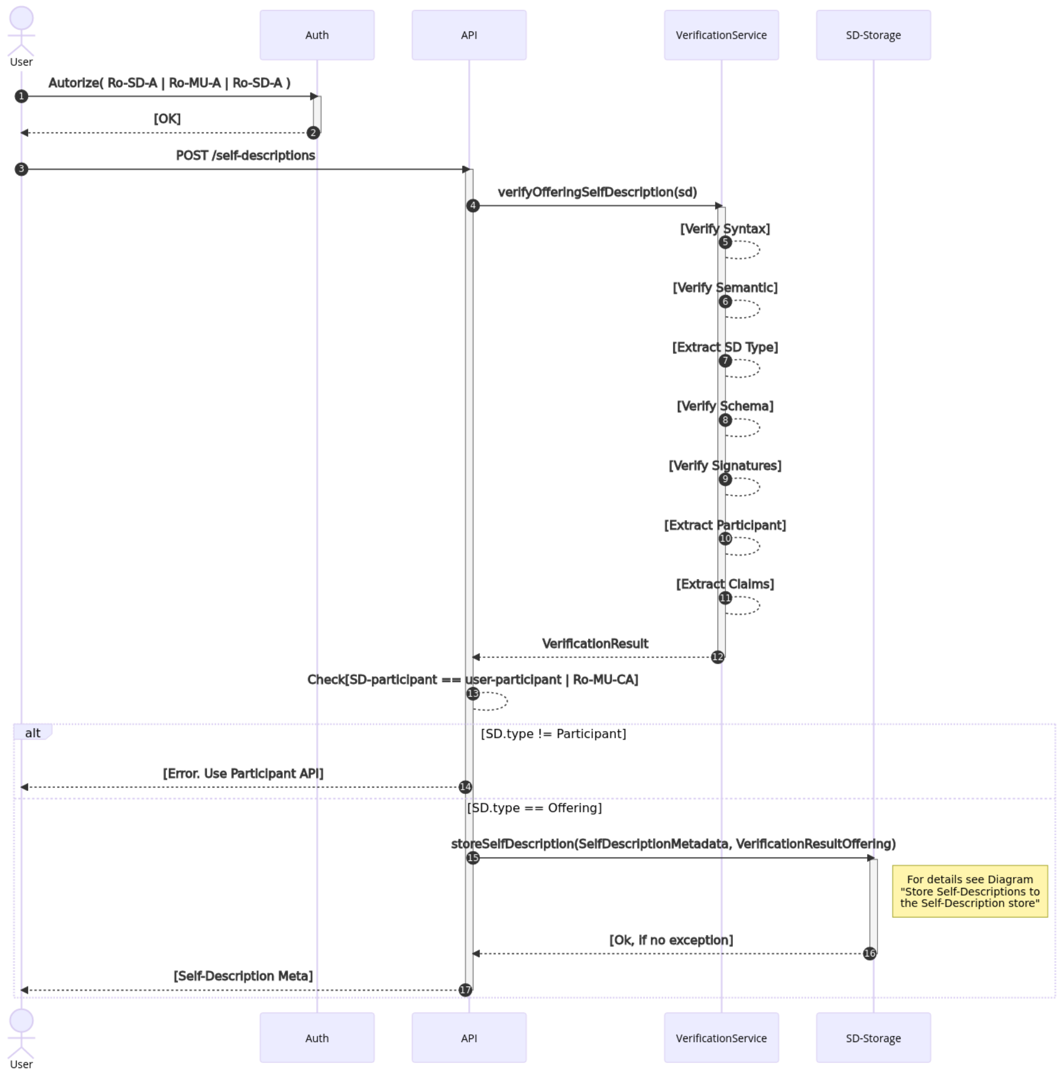

User
Auth
API
VerificationService
SD-Storage
Autorize(Ro-SD-A | Ro-MU-A | Ro-SD-A)
[OK]
POST /self-descriptions
verifyOfferingSelfDescription(sd)
[Verify Syntax]
[Verify Semantic]
[Extract SD Type]
[Verify Schema]
[Verify Signatures]
[Extract Participant]
[Extract Claims]
VerificationResult
Check[SD-participant == user-participant | Ro-MU-CA]
alt
[SD.type != Participant]
[Error. Use Participant API]
[SD.type == Offering]
storeSelfDescription(SelfDescriptionMetadata, VerificationResultOffering)
For details see Diagram "Store Self-Descriptions to the Self-Description store"
[Ok, if no exception]
[Self-Description Meta]

Figure 89: Adding a Self-Description for an Offering (source: GAIA-X[9])

- Requesting all Self-Descriptions
  Once the Approved Participants (in this case, the Consumers) receive the necessary credentials and access rights to engage within the GAIA-X ecosystem they can interact with the GAIA-X Catalogue to explore available Service Offerings. The Catalogue provides a comprehensive view of the offerings, enabling Consumers to make informed decisions.
  The Consumer can narrow down the search within the Catalogue to find specific Self-Descriptions by using specific keywords and filters.

[9] https://gaia-x.gitlab.io/data-infrastructure-federation-services/cat/architecture-document/images/diag-0b8bbfdd68c0886283d848ecbadcf283.png

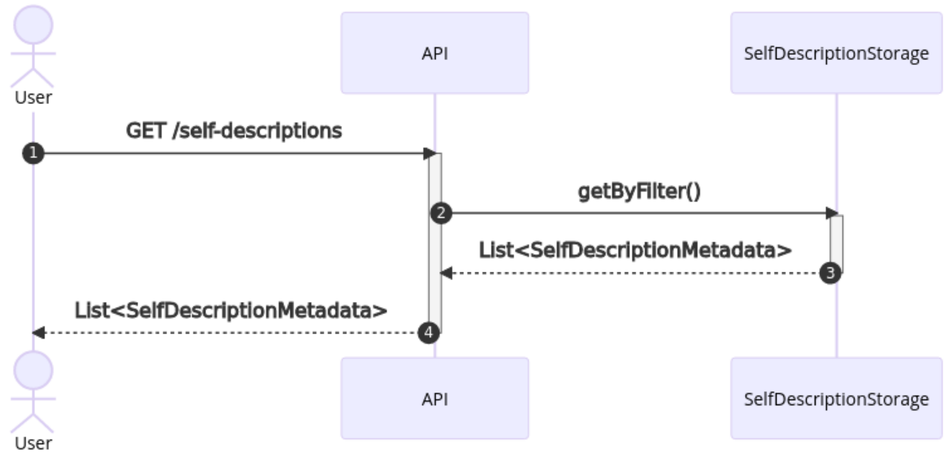


Figure 90: Requesting all Self-Descriptions (source: GAIA-X[10])

- Requesting a specific Self-Description

The Consumer can request access to the detailed Self-Description or additional information from the search results by specifying the concrete Self-Description hash in a new request to the Catalogue.

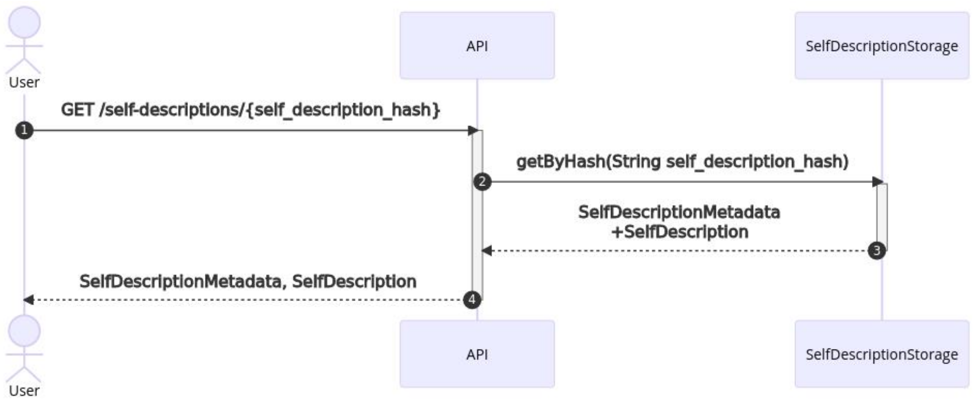

User
API
SelfDescriptionStorage
GET /self-descriptions/{self_description_hash}
getByHash(String self_description_hash)
SelfDescriptionMetadata +SelfDescription
SelfDescriptionMetadata, SelfDescription
1
2
3
4

Figure 91: Requesting information about a specific Self-Description (source: GAIA-X[11])

Contract Negotiation

When a Consumer identifies a relevant Service Offering, they engage in Contract negotiations with the respective Provider. The negotiation process establishes the terms and conditions under which the Service Instance will be provided.

[10] https://gaia-x.gitlab.io/data-infrastructure-federation-services/cat/architecture-document/images/diag-6e1c2d3c7a88f8d142a97e259c67c314.png
[11] https://gaia-x.gitlab.io/data-infrastructure-federation-services/cat/architecture-document/images/diag-5832aadb62702f484c64f9335a6171ae.png

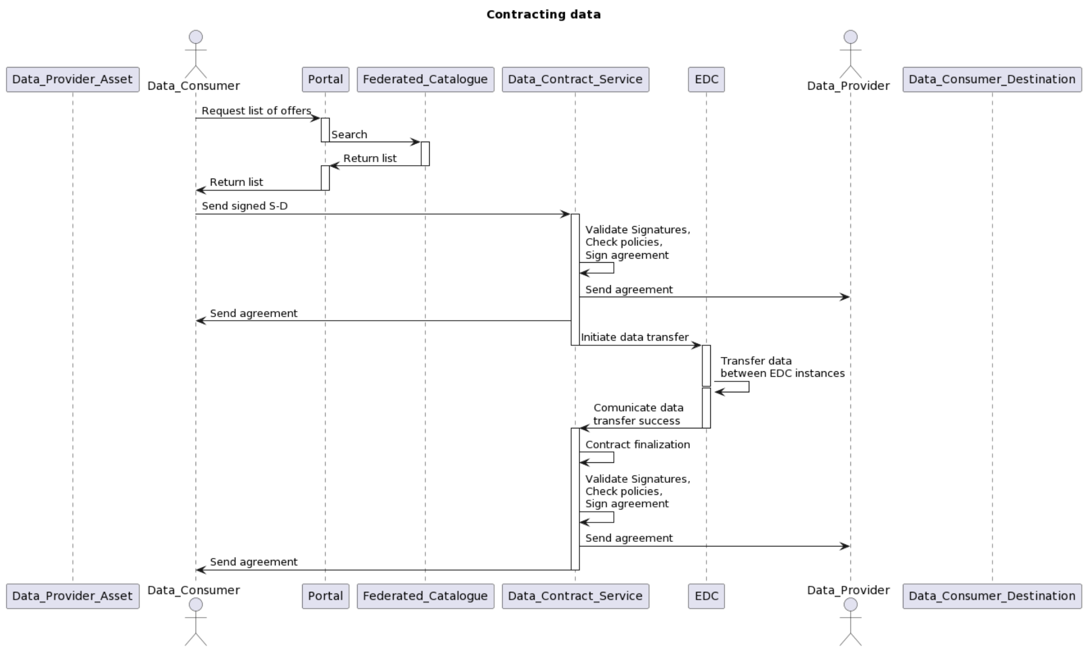


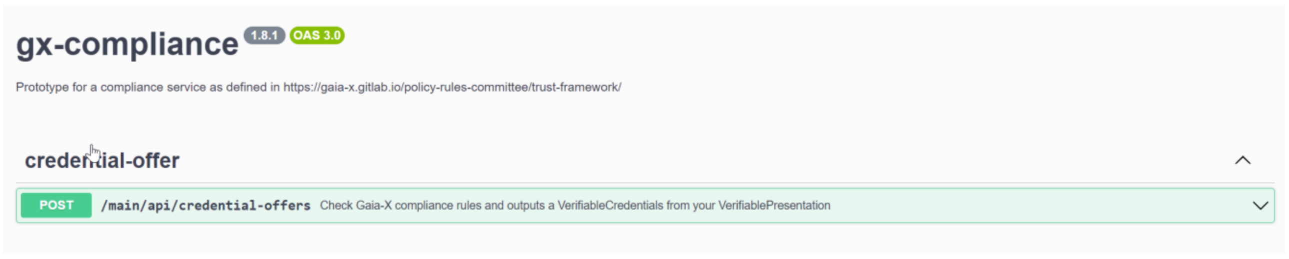
Figure 92: Contract Negotiation of an Offering

2.26.4 API

The following section provides a list of the main APIs of DIDI Data Marketplace components.

Compliance Service

Link to OpenAPI specification: https://compliance.lab.gaia-x.eu/main/docs


gx-compliance 1.8.1 OAS 3.0

Prototype for a compliance service as defined in https://gaia-x.gitlab.io/policy-rules-committee/trust-framework/

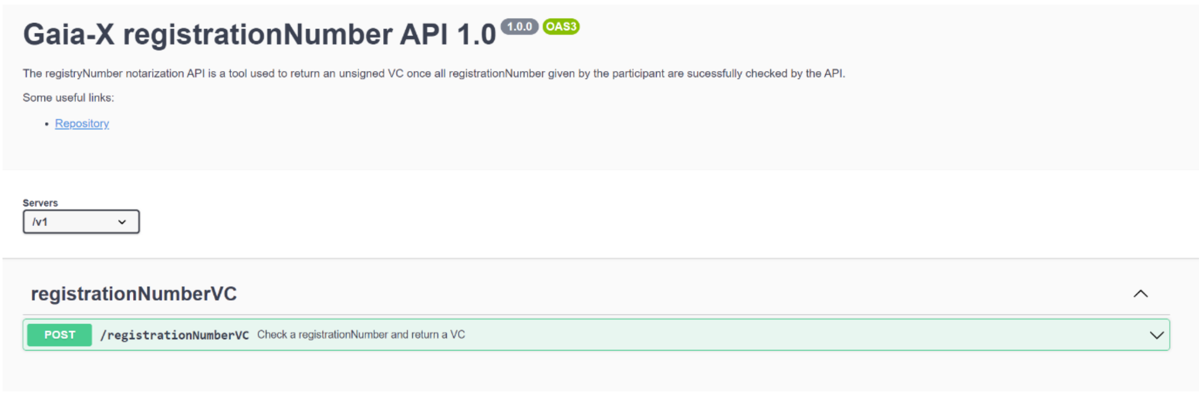
credential-offer

POST /main/api/credential-offers Check Gaia-X compliance rules and outputs a VerifiableCredentials from your VerifiablePresentation

Figure 93: API Endpoint – Compliance Service Prototype

Notarization Service

Link to OpenAPI specification: https://registrationnumber.notary.gaia-x.eu/v1/docs/



Gaia-X registrationNumber API 1.0 1.0.0 OAS3

The registryNumber notarization API is a tool used to return an unsigned VC once all registrationNumber given by the participant are sucessfully checked by the API.

Some useful links:

- Repository

Servers: /v1

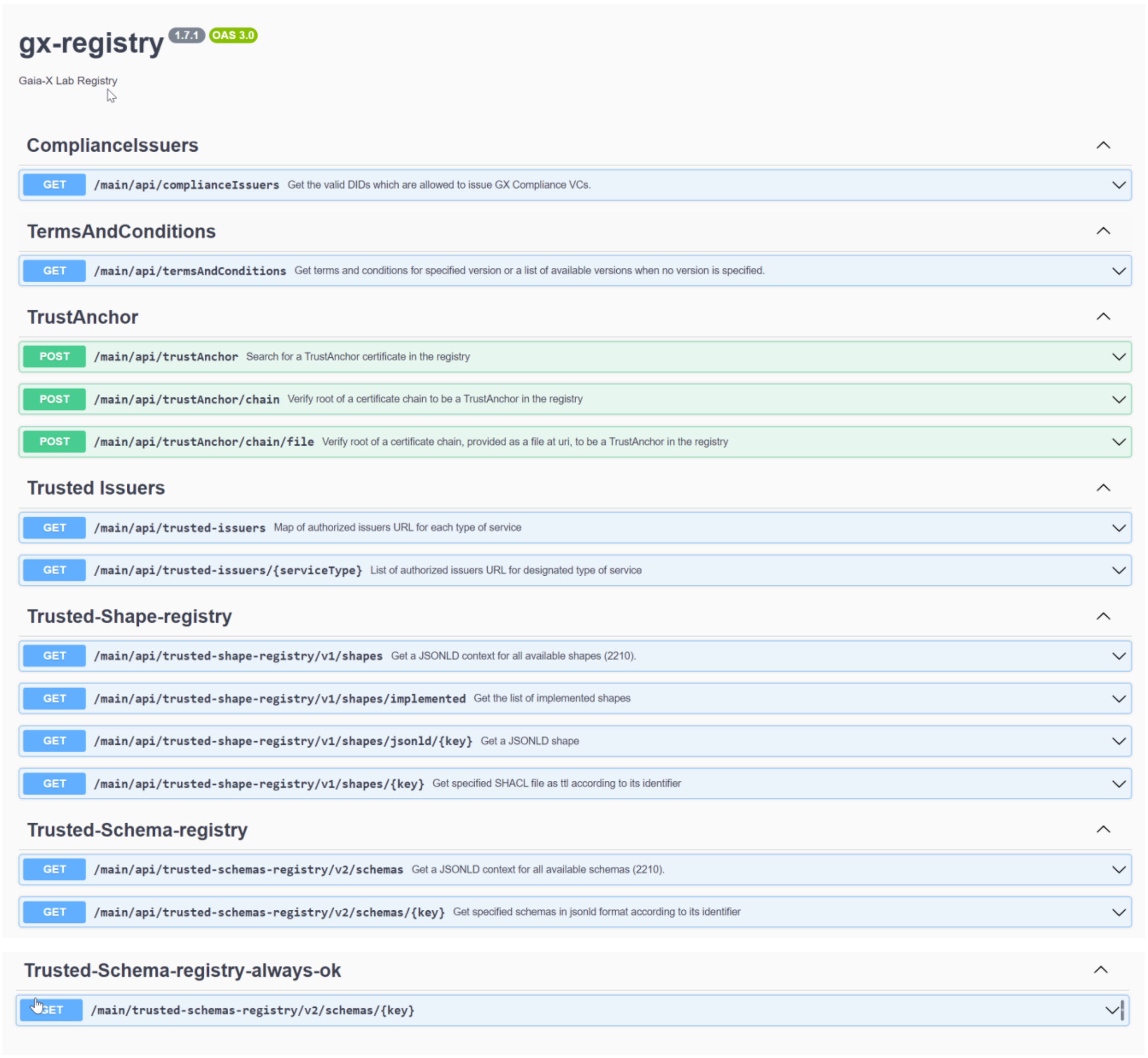
registrationNumberVC

POST /registrationNumberVC Check a registrationNumber and return a VC

Figure 94: API Endpoint – Gaia-X registrationNumber API 1.0

Registry Service

Link to OpenAPI specification: https://registry.lab.gaia-x.eu/main/docs/



gx-registry 1.7.1 OAS 3.0

Gaia-X Lab Registry

ComplianceIssuers

GET /main/api/complianceIssuers Get the valid DIDs which are allowed to issue GX Compliance VCs.

TermsAndConditions

GET /main/api/termsAndConditions Get terms and conditions for specified version or a list of available versions when no version is specified.

TrustAnchor

POST /main/api/trustAnchor Search for a TrustAnchor certificate in the registry

POST /main/api/trustAnchor/chain Verify root of a certificate chain to be a TrustAnchor in the registry

POST /main/api/trustAnchor/chain/file Verify root of a certificate chain, provided as a file at uri, to be a TrustAnchor in the registry

Trusted Issuers

GET /main/api/trusted-issuers Map of authorized issuers URL for each type of service

GET /main/api/trusted-issuers/{serviceType} List of authorized issuers URL for designated type of service

Trusted-Shape-registry

GET /main/api/trusted-shape-registry/v1/shapes Get a JSONLD context for all available shapes (2210).

GET /main/api/trusted-shape-registry/v1/shapes/implemented Get the list of implemented shapes

GET /main/api/trusted-shape-registry/v1/shapes/jsonld/{key} Get a JSONLD shape

GET /main/api/trusted-shape-registry/v1/shapes/{key} Get specified SHACL file as ttl according to its identifier

Trusted-Schema-registry

GET /main/api/trusted-schemas-registry/v2/schemas Get a JSONLD context for all available schemas (2210).

GET /main/api/trusted-schemas-registry/v2/schemas/{key} Get specified schemas in jsonld format according to its identifier

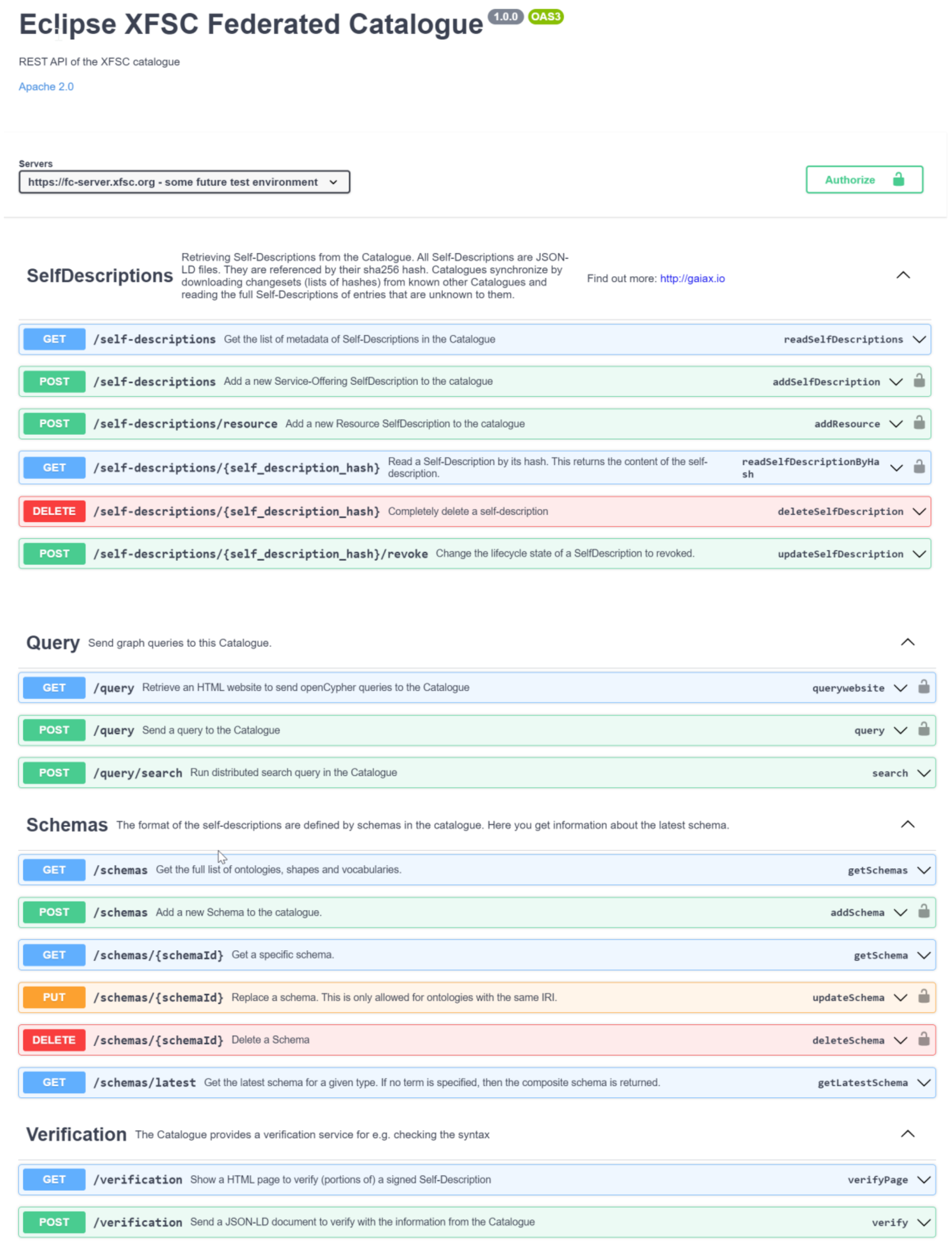
Trusted-Schema-registry-always-ok

GET /main/trusted-schemas-registry/v2/schemas/{key}

Figure 95: API Endpoints – Gaia-X Lab Registry

Federated Catalogue APIs

Link to OpenAPI specification: https://gitlab.eclipse.org/eclipse/xfsc/cat/fc-service/-/blob/main/openapi/fc_openapi.yaml



Eclipse XFSC Federated Catalogue 1.0.0 OAS3

REST API of the XFSC catalogue

Apache 2.0

Servers
https://fc-server.xfsc.org - some future test environment

Authorize

SelfDescriptions Retrieving Self-Descriptions from the Catalogue. All Self-Descriptions are JSON-LD files. They are referenced by their sha256 hash. Catalogues synchronize by downloading changesets (lists of hashes) from known other Catalogues and reading the full Self-Descriptions of entries that are unknown to them. Find out more: http://gaiax.io

GET	/self-descriptions	Get the list of metadata of Self-Descriptions in the Catalogue	readSelfDescriptions
POST	/self-descriptions	Add a new Service-Offering SelfDescription to the catalogue	addSelfDescription
POST	/self-descriptions/resource	Add a new Resource SelfDescription to the catalogue	addResource
GET	/self-descriptions/{self_description_hash}	Read a Self-Description by its hash. This returns the content of the self-description.	readSelfDescriptionByHash
DELETE	/self-descriptions/{self_description_hash}	Completely delete a self-description	deleteSelfDescription
POST	/self-descriptions/{self_description_hash}/revoke	Change the lifecycle state of a SelfDescription to revoked.	updateSelfDescription

Query Send graph queries to this Catalogue.

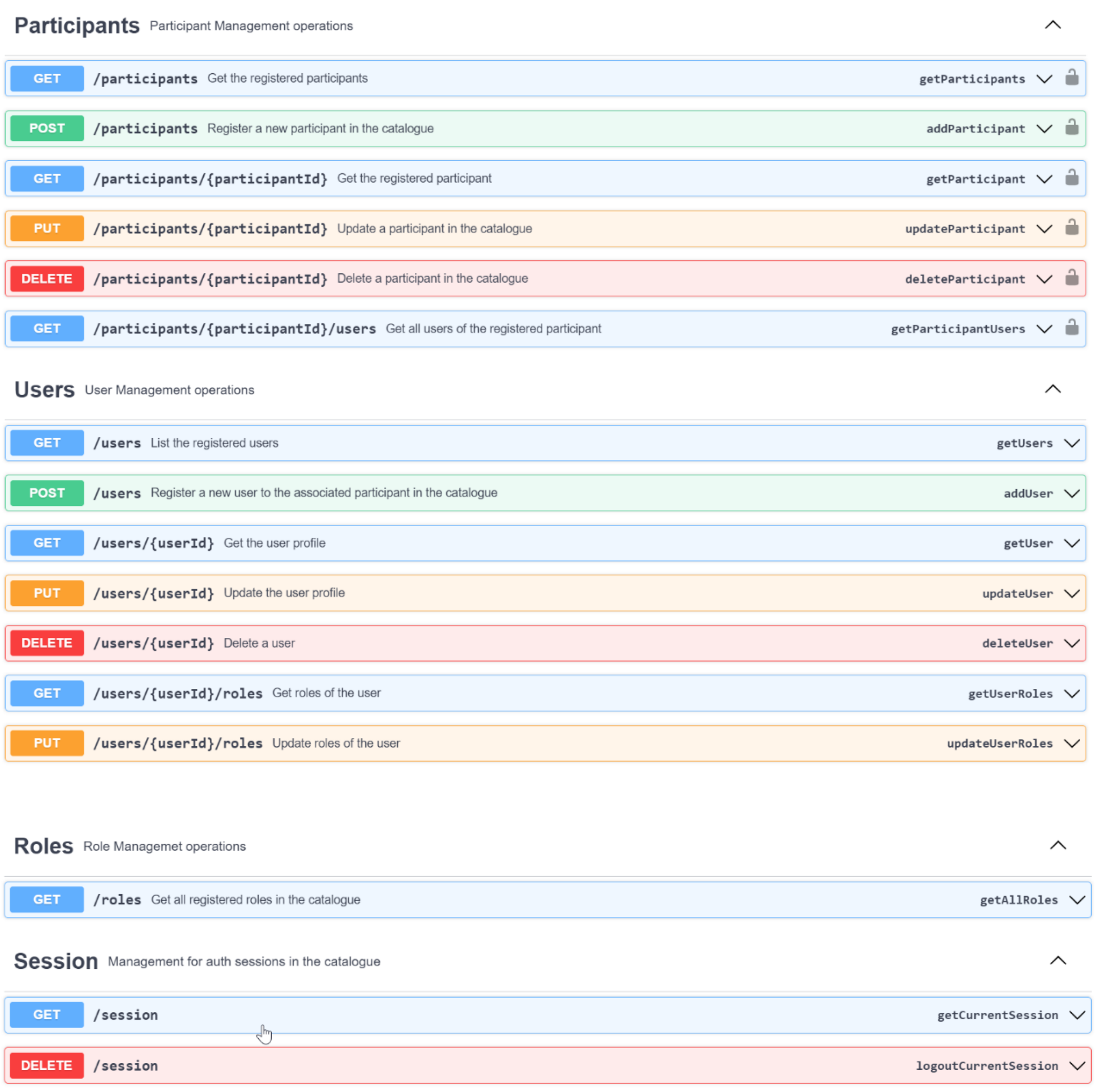
GET	/query	Retrieve an HTML website to send openCypher queries to the Catalogue	querywebsite
POST	/query	Send a query to the Catalogue	query
POST	/query/search	Run distributed search query in the Catalogue	search

Schemas The format of the self-descriptions are defined by schemas in the catalogue. Here you get information about the latest schema.

GET	/schemas	Get the full list of ontologies, shapes and vocabularies.	getSchemas
POST	/schemas	Add a new Schema to the catalogue.	addSchema
GET	/schemas/{schemaId}	Get a specific schema.	getSchema
PUT	/schemas/{schemaId}	Replace a schema. This is only allowed for ontologies with the same IRI.	updateSchema
DELETE	/schemas/{schemaId}	Delete a Schema	deleteSchema
GET	/schemas/latest	Get the latest schema for a given type. If no term is specified, then the composite schema is returned.	getLatestSchema

Verification The Catalogue provides a verification service for e.g. checking the syntax

GET	/verification	Show a HTML page to verify (portions of) a signed Self-Description	verifyPage
POST	/verification	Send a JSON-LD document to verify with the information from the Catalogue	verify



Participants Participant Management operations

GET /participants Get the registered participants getParticipants

POST /participants Register a new participant in the catalogue addParticipant

GET /participants/{participantId} Get the registered participant getParticipant

PUT /participants/{participantId} Update a participant in the catalogue updateParticipant

DELETE /participants/{participantId} Delete a participant in the catalogue deleteParticipant

GET /participants/{participantId}/users Get all users of the registered participant getParticipantUsers

Users User Management operations

GET /users List the registered users getUsers

POST /users Register a new user to the associated participant in the catalogue addUser

GET /users/{userId} Get the user profile getUser

PUT /users/{userId} Update the user profile updateUser

DELETE /users/{userId} Delete a user deleteUser

GET /users/{userId}/roles Get roles of the user getUserRoles

PUT /users/{userId}/roles Update roles of the user updateUserRoles

Roles Role Managemet operations

GET /roles Get all registered roles in the catalogue getAllRoles

Session Management for auth sessions in the catalogue

GET /session getCurrentSession

DELETE /session logoutCurrentSession

Figure 96: API Endpoints – Eclipse XFSC Federated Catalogue

2.26.5 Input/Output

In the context of Gaia-X, dataspace self-descriptions refer to the metadata and information associated with a dataspace that provide essential details about its characteristics, data assets, and usage policies. These self-descriptions play a crucial role in facilitating data sharing and collaboration within the Gaia-X ecosystem, as they enable organizations to discover, evaluate, and interact with dataspaces in a standardized and trustworthy manner.

The following JSON snippets represent different examples of Self-Descriptions used in the interaction (via their APIs) with the GAIA-X components that conform DIDI Data Marketplace.

For detailed examples on how to use the DIDI Data Marketplace components (as well as their input and outputs), please refer to the OpenAPI links provided for each of them in the API section.

Participant Self-Description example:

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1"
  ],
  "@id": "http://example.edu/verifiablePresentation/self-
description1",
  "type": [
    "VerifiablePresentation"
  ],
  "verifiableCredential": [
    {
      "@context": [
        "https://www.w3.org/2018/credentials/v1"
      ],
      "@id": "https://www.example.org/legalPerson.json",
      "@type": [
        "VerifiableCredential"
      ],
      "issuer": "http://gaiax.de",
      "issuanceDate": "2022-10-19T18:48:09Z",
      "credentialSubject": {
        "@context": {
          "gax-core": "https://w3id.org/gaia-x/core#",
          "gax-trust-framework": "https://w3id.org/gaia-x/gax-trust-
framework#",
          "xsd": "http://www.w3.org/2001/XMLSchema#",
          "vcard": "http://www.w3.org/2006/vcard/ns#"
        },
        "@id": "gax-core:Participant1",
        "@type": "gax-trust-framework:LegalPerson",
        "gax-trust-framework:registrationNumber": "1234",
        "gax-trust-framework:legalAddress": {
          "@type": "vcard:Address",
          "vcard:country-name": "Country",
          "vcard:locality": "Town Name",
          "vcard:postal-code": "1234",
          "vcard:street-address": "Street Name"
        },
        "gax-trust-framework:headquarterAddress": {
          "@type": "vcard:Address",
          "vcard:country-name": "Country",
          "vcard:locality": "Town Name",
          "vcard:postal-code": "1234",
          "vcard:street-address": "Street Name"
        },
        "gax-trust-framework:termsAndConditions": {
          "@type": "gax-trust-framework:TermsAndConditions",
          "gax-trust-framework:content": {
            "@type": "xsd:anyURI",
            "@value": "http://example.org/tac"
          },
          "gax-trust-framework:hash": "1234"
        },
          "gax-trust-framework:subOrganisation": [
            {
              "@id": "http://example.org/Provider1_1"
            },
            {
              "@id": "http://example.org/Provider1_2"
            }
          ],
          "gax-trust-framework:legalName": "Provider Name"
        },
        "proof": {
          "type": "JsonWebSignature2020",
          "created": "2022-12-02T16:05:37Z",
          "proofPurpose": "assertionMethod",
          "verificationMethod": "did:web:compliance.lab.gaia-x.eu",
          "jws":
"eyJiNjQiOmZhbHNlLCJjcml0IjpbImI2NCJdLCJhbGciOiJQUzI1NiJ9..efXJBVbLU
ieloVXmK11FEmM8ke_QYhf3ObjxKHzzdujrwgToHcpVKNUYwlnUXk7-
V6cuaUFwbWY1zl89u0U2Nu2UKA1kP6iBmUyg1JnWXsCtF1dpFyCZMhrYdbJxF-
USa9f4RbTDcymhbRx8ZI9R4qdhDGuxDrez_Nzl2DlJfSL7hE9JBG7R8cgAq1LIfWCTN0
xjVr8QvVw3R_HIilDSLv-
Clf1WCSAl_7CWXDEGInBW6l7lrJ7efrjZ5GnEwbHZi0b2V0v9hjidqkMc1xl5pl9fIPk
5wHsoVLdKgfJ8hUD-
EtyuFJGPwk77Mqf7BVcuKR4zqmNX3s0CPCGmAmxXfGyY5ARITjgmI4gXcp1I1ax1YTCj
ENR8nqgS8rouO6l3xAAFgBqLdqfR1KdDiCYPFd0fTfX1T8A8yiT8o6Pg36Vewy-
Jb79aWk9byxSS-xKaQGkSMbpZeErz4FKTXLYwcD8bLXcndPMtF5UlBC_A_t-
_BnG5cLeZddSow-
4sMs7g2qvnURIX_KRrPtuP5GRVG_cNPJagY5bzUc8lAlSNsRsivf5wJDwrRvYjn3U6FS
V8sOBBGVv6UBlopl1JWjUL-9-
QLmZUD1jPv9mYUWWChm2YB8dvCjkrvwRCbHFAuab2rYNag61EcYI3lcGwS3Qez-
P4AKIpRfTXEpnCNIzJr2E"
        }
      }
    ],
    "proof": {
      "type": "JsonWebSignature2020",
      "created": "2022-12-02T16:05:37Z",
      "proofPurpose": "assertionMethod",
      "verificationMethod": "did:web:compliance.lab.gaia-x.eu",
      "jws":
"eyJiNjQiOmZhbHNlLCJjcml0IjpbImI2NCJdLCJhbGciOiJQUzI1NiJ9..IYM1hcL55
GNc115qwjdwAiHoxnx7DD4MwWcYNUDgRO2Tj6vcEtKl5Ao1f_uwpTEJBImYrd4tZL9oj
DNBOTmOnxFWorsUB-iq5PMvM11xS19tl-
hEhRVRY0mnFkT9er2xArWShcO6cNTnDAJuWGCtHxsU-bH3HMHCvT9u2WWKIIIJi9Axp-
CGwnNaF7vddEatiXRfuZCj8RCYxKa5goCxE4vueI-OMqIF-
AWU86FjTNjDXS9DJI2yYt91SFdgxQqfuG0pJF7oJv9LI-
9bJjMRKBSfjPO1hqbxuPzxb4a7nywQedl1_2k2WttUQ3ZDsyju7ktkuGDDPL9p3xq6zy
BsFN9shJqdS-9-tw3Nptu_EsNj1vJdNgrOZt1VdfOEGoUsDtfNg2O5XtBt4-
TbE617_SgkWQhbzWAPWj47QJjcYJklzkJku7DwBicysvCiWGPOgCzFalTZPm6SL55Bz4
a0UtEW8WzjtlxD_j3Mh7WW_sA3vAO9oBLRaWJhQkt0f6EG7sElETizVDB9fq0Ur5QYuH
hPmCqZVr5C0VSSBz4OSE_N7DFusNqLhox13f37myfs0RiTpmwEh24EP67kyZSuUtOx00
4fqAWifIYiyBZj2OTdxnXy7NFiOZl80RM1cJ7WTL02-
159_7E88y1LiL39uWUatFYJBCKnaMRO5uQ8BGU"
  }
}
```

Provider's Offering Self-Description

```
{
    "@id": "http://example.edu/verifiablePresentation/self-
description1",
    "proof": {
        "created": "2023-03-01T11:05:08Z",
        "jws":
"eyJiNjQiOmZhbHNlLCJjcml0IjpbImI2NCJdLCJhbGciOiJQUzI1NiJ9..RMGGGRlvi
GNdoBBTygv9JohKwB88mUQirUOZnOhxvNma-
kBvP1kXvnnMFVitJ26wsctJjk5hM4EDiagGaxNplBPtFgrrilp6UgGlQiQIx8arjAnwk
tQIx3HF9E7uZ5iucwlfrsUjro5svej4aqS5n-
tCNDW6wwkLgfd1frjIkWb01GA17qa3U1E762-
1pnu5Z3EOSatufcnXgY3GdCOezP57y8-CLrpE9s8iTtRfNuJbs-4ka09akuDq_Z-
6K_buQ5IXo5_B9C69DAkH_uzj9CGEgZxRwQg-
ZykVXdDenfm9TkT2mPlqolt30cUyhDatMwHkOZpRk1vDdQMTmJwdPt_vpVpyXUGcN59N
D5stuscXkU43e0lWoivtEs9eHmcuFz-mRsx6VnI-
84qkH_TPdB2g5gfsH0HJCB9poK8wAPpNq3O7hf9rnctwt6doc_ZOKkZrVtJT9zaSR78v
cER1FXInxx9Qe6idcIYR105YS9Q_VNJSE0vRj6fddi3B05FjweLW3XIqpDmRsFkuAhZI
6aI6auNRtLlpNg-
xgr_64Ek9EcaLc7PLw6iCvmdIYkl9NzUZgF0EDBPfOHIyjVprVO3bZ16WezbqjGnw6uR
2MukDpretLN4ZbhqOiJM80BMz6Kdyk99bWy1xtCseu2Z6eNjwo3QcPhaZfw0pFeWb0V8
",
        "proofPurpose": "assertionMethod",
        "type": "JsonWebSignature2020",
        "verificationMethod": "did:web:compliance.lab.gaia-x.eu"
    },
    "type": ["VerifiablePresentation"],
    "@context": ["https://www.w3.org/2018/credentials/v1"],
    "verifiableCredential": {
        "issuanceDate": "2022-10-19T18:48:09Z",
        "credentialSubject": {
            "gax-trust-framework:policy": "list of policy",
            "gax-trust-framework:serviceTitle": "Software Title",
            "dcat:keyword": [
                "Keyword1_1",
                "Keyword1_2",
                "Keyword1_3",
            ],
            "@type": "gax-trust-framework:ServiceOffering",
            "gax-core:offeredBy": {"@id":
"https://www.example.org/Provider1.json"},
            "@id": "https://www.example.org/mySoftwareOffering",
            "@context": {
                "gax-trust-framework": "https://w3id.org/gaia-x/gax-
trust-framework#",
                "xsd": "http://www.w3.org/2001/XMLSchema#",
                "dcat": "http://www.w3.org/ns/dcat#",
                "gax-core": "https://w3id.org/gaia-x/core#"
            },
            "gax-trust-framework:dataAccountExport": {
                "gax-trust-framework:formatType":
"application/gzip",
                "@type": "gax-trust-framework:DataAccountExport",
                "gax-trust-framework:accessType": "digital",
                "gax-trust-framework:requestType": "API"
            },
            "gax-trust-framework:termsAndConditions": {
                "gax-trust-framework:content": {
                    "@value": "http://example.org/tac",
                    "@type": "xsd:anyURI"
                },
                "@type": "gax-trust-framework:TermsAndConditions",
                "gax-trust-framework:hash": "1234"
                }
        },
        "@type": ["VerifiableCredential"],
        "@id": "https://www.example.org/SoftwareOffering.json",
        "proof": {
            "created": "2023-03-01T11:05:07Z",
            "jws":
"eyJiNjQiOmZhbHNlLCJjcml0IjpbImI2NCJdLCJhbGciOiJQUzI1NiJ9..Y9vkgfzsc
LnY82AUTJsevtve5pmLv28VP39SvjaeKvUk9hUyzFIVrIUxfhIcMowRExLxzMVFKm3lz
_1km6-6TU3HkKHgMIhUny6VEv_ozPQYYAWv-TWahefjnpb1Ta6Sl_K7mA4gpuKOFg7C-
WQDz8JL-
L6OHCSeiPmBUxkhmKdcMFbBX4dMoj90Oqj05hCQqmSs_bYaipMXsa8lDlaZhNQ9mk10w
qe3SEf-Cp9YS0d-
dvJvdw23HqB_DqH6UFoQ_NcANUKUKOxHDjIafCXhL8j59KezM0fV8rmVzRDxk5wsD0vj
AwAOz4tCiiOxVPjQMdQub4ISNEw0ZcLGOYYKsRkIHQHYQo1cbgWg_f8Z6hebI5Jx8rDd
wz0vnJky1mw5I09CprPq03Kh6E9PXWPiIJFU4jZxZdDyginm3bZfVaKZ07KRF3HEq00v
OgjHBpNaNXkxloUi6hmVEGYTp04OQ-
jopQM87zvnoIOqB78jQpRSQK2CYckAph_R0Sna_qI5unb5t-
29DDNKuRxYOBJoK5O5i0OgbfecY82FyTvPOoYr2q7tN1WONlZqqBdlyqoo_NBsJ3xVsJ
QMXeTpF8XCOb9iCn-R3qpCZFdyg-
6C0MWMTft8dEZIqVOfyuDSKyiz24qiHufVt94v87LlYIkDnNIFitkLqh4hQ7R4uYIMtE
o",
            "proofPurpose": "assertionMethod",
            "type": "JsonWebSignature2020",
            "verificationMethod": "did:web:compliance.lab.gaia-x.eu"
        },
        "@context": ["https://www.w3.org/2018/credentials/v1"],
        "issuer": "http://gaiax.de"
    }
}
```

Contract Negotiation Request example:

```
{
  "@context": {
    "edc": "https://w3id.org/edc/v0.0.1/ns/",
    "odrl": "http://www.w3.org/ns/odrl/2/"
  },
  "@type": "NegotiationInitiateRequestDto",
  "connectorId": "provider",
  "connectorAddress": "http://localhost:19194/protocol",
  "consumerId": "consumer",
  "providerId": "provider",
  "protocol": "dataspace-protocol-http",
  "offer": {
    "offerId":
"MQ==:YXNzZXRJZA==:YTc4OGEwYjMtODRlZi00NWYwLTgwOWQtMGZjZTMwMGM3Y2Ey"
,
    "assetId": "assetId",
    "policy": {
      "@id":
"MQ==:YXNzZXRJZA==:YTc4OGEwYjMtODRlZi00NWYwLTgwOWQtMGZjZTMwMGM3Y2Ey"
,
      "@type": "Set",
      "odrl:permission": [],
      "odrl:prohibition": [],
      "odrl:obligation": [],
      "odrl:target": "assetId"
    }
    }
}
```

2.26.6 Implementation Technology

Given the large diversity of modules and services that comprise the DIDI Data Marketplace a summary follows of the main technologies used in the implementation of the components depicted in the architecture diagram.

<u>Data Space Horizontal Modules:</u>

- *Notarization Service*: Node, Docker
- *Compliance Service*: Node, Docker
- *Registry Service*: Node, MongoDB, Docker
- *Federated Catalogue:* Java 11, Spring Security and Spring Boot, PostgreSQL, Neo4j with neosemantics, Keycloak, Docker

<u>Data and Services Marketplace:</u>

- *Data Contract Service:* Node, Redis, Docker
- *Data Exchange Logging Service:* Node, Docker
- *Portal*: React, Docker

<u>Data Exchange Services:</u>

- *Data Connector*:
    - Eclipse Data Connector (EDC)[12]: Java

2.26.7 Comments

The aim of this document is not to provide a comprehensive and fully detailed description of DIDI Data Marketplace and its components given the fact that its implementation leverages many of the GAIA-X open-source components.

Please, refer to GAIA-X documentation for a complete and very detailed description of all concepts mentioned here as well as for the technical details of the software components:

- GAIA-X Architecture Document: https://docs.gaia-x.eu/technical-committee/architecture-document/22.10/
- GAIA-X Services (source repository): https://gitlab.com/gaia-x

12 https://github.com/eclipse-edc/Connector

3 RE4DY Data Space and Toolware Catalogue: Experiment Plan

This section provides a list of experimentation plans used to assess the applicability and functionality of a tool or integrated set of tools in the TEFs and pilot sites included in the RE4DY project. Each plan describes the subject and aim of the experiment and introduces a number of scenarios which enumerate the steps to be followed in order to test each tool leveraged in the respective scenario. Furthermore, every scenario outlines the components which participate in the scenario, the datasets or AI models employed for its implementation and a list of proposed added values. In addition, it declares a number of KPIs used to examine the functionality of the related components and finally, the partners who are involved in the scenario.

3.1 Enabling digital resources sharing, findability and monetization.

In the manufacturing sector, supply chain collaboration is crucial for the efficient and reliable production of goods. The aim of this experiment is to test and demonstrate the feasibility of enabling collaboration among multiple manufacturing organizations within a supply chain to share digital resources seamlessly and securely by using a digital data marketplace based on the dataspace concepts and technologies.

To that end, this experiment will consider the following scenarios:

- Resource Publication in the data marketplace
- Discoverability and Access (including FAIR Access Verification)
- Contract Signing and Payment

3.1.1 Scenario 01: Resource Publication in the data marketplace

In this scenario a set of digital resources will be published in the data marketplace using self-descriptions, which will include detailed metadata, specifying the data type, source, price, and usage restrictions among others. The type of digital resources considered for publication in the marketplace are:

- Datasets generated by a manufacturing organization and offered using the "data as a product" concept as defined by RE4DY
- AI and machine learning models published as digital assets, complete with documentation on their functionality, input data requirements, and outcomes.

Components participation	• Atos' 'Dataspace for Industrial Data Intelligence (DIDI)' based on several GAIA-X software components • eIDAS • Data as Product Containers (DC)
Datasets/AI Models	To be confirmed which ones
Proposed Added Value	• Interoperability: Standardized formats improve interoperability among manufacturing organizations, streamlining operations. • FAIR Compliance: The implementation of FAIR principles ensures that digital resources are well-structured and readily discoverable. • Data Governance and Compliance: DIDI data marketplace helps enforce data governance policies and regulatory compliance by setting access controls, contracts, and usage terms for resources. This mitigates legal and compliance risks.
KPIs	• Number of datasets published: at least 2 datasets will be published free of cost and 2 others with an associated cost. • Number of AI models published: at least 2 AI models will be published.
Partners Participation	ATOS, INTRA, UPV

3.1.2 Scenario 02: Discoverability and Access (including FAIR Access Verification)

In this scenario it will be tested whether a manufacturing organization user can easily search for digital resources within the DIDI data marketplace using specific criteria, such as product type, production stage, supplier or cost.

The FAIR (Findable, Accessible, Interoperable, and Reusable) principles are implemented to ensure that digital resources are easily discoverable and accessible. The self-descriptions associated with each resource enable automated searches and seamless access.

Users can access resources based on permissions and contracts defined within the DIDI data marketplace for each of the resources. Access can be free for open resources, while premium resources may require contractual agreements and potentially involve payment transactions (see scenario 03).

In some cases, access to the data will be done via specific connectors (e.g., Eclipse Data Connector (EDC) in compliance with IDSA/GAIA-X) whereas in others a link or API will be available for direct retrieval once the access has been authorized.

Components participation	• Atos' 'Dataspace for Industrial Data Intelligence (DIDI)' based on several GAIA-X software components • eIDAS • F-UJI FAIRness assessment tool
Datasets/AI Models	To be confirmed which ones
Proposed Added Value	• Improved Accessibility: DIDI data marketplace enables resources to be made accessible to a wider audience through standardized interfaces and authentication mechanisms. By ensuring that the right users have access to the right data, it promotes data accessibility, while also enforcing data security and privacy, which aligns with the FAIR principle of Accessibility. • Efficiency: The DIDI data marketplace facilitates the swift exchange of production data and models, enabling manufacturing organizations to respond quickly to supply chain demands. • Interoperability: Standardized formats improve interoperability among manufacturing organizations, streamlining operations. • Quality and Consistency: Standardized data formats and clear self-descriptions ensure data quality and consistency, reducing errors in data interpretation. • Cost Reduction: By improving resource discovery and access, Dataspaces can reduce costs associated with redundant data acquisition, processing, and management. This is particularly valuable in sectors where data efficiency is essential, such as manufacturing. • Automation and Scalability: DIDI data marketplace supports automated processes for resource discovery, retrieval, and management. Automation helps streamline operations and enables scalability, making it easier to handle an ever-growing volume of digital resources.
KPIs	• FAIRNESS of the datasets discovered: assess the degree of fairness of each of the datasets discovered in the data marketplace. • Data access methods: at least one dataset will be accessed using the Data as Product Container API; at

	least one dataset will be accessed using the Eclipse Data Connector (EDC); at least one dataset/model will be accessed using a common/proprietary API (as defined by the data owner).
Partners Participation	ATOS, INTRA

3.1.3 Scenario 03: Contract Signing and Payment

This scenario will test and demonstrate the specific use case when a digital resource has a cost and requires signing a contract and adhering to the terms of use before accessing sensitive data or manufacturing AI models found through DIDI data marketplace.

The scenario will validate the effectiveness of Keycloak and EIDAS in granting authorized access to sensitive manufacturing data.

In addition, two monetization schemas will be implemented and tested: pay-per-use (that is, pay once and use it) and a subscription model.

Components participation	• Atos' 'Dataspace for Industrial Data Intelligence (DIDI)' based on several GAIA-X software components. • eIDAS
Datasets/AI Models	To be confirmed which ones
Proposed Added Value	• Security: Strong authentication, authorization, and encryption mechanisms protect sensitive data and manufacturing models. • Monetization: The system allows for the monetization of premium resources, which can generate revenue for the resource owners. • Contract Management: Digital supply chain contracts are executed within the platform, guaranteeing that agreements are honoured and that goods and services are delivered as specified.
KPIs	• Number of monetization schemas: at least 2 datasets with two different payment schemas will be available, requiring to sign a contract.
Partners Participation	ATOS, INTRA

3.2 Dataset lifecycle traceability

Data provenance and traceability involves tracking and tracing the lifecycle of a piece of data. It constitutes a valuable tool in multiple sectors as it ensures transparency, accountability and aids in demonstrating regulatory compliance. This experiment aims to exhibit how blockchain-based asset traceability can prove beneficial in a data marketplace where transparency and accountability are of essential importance, by using a blockchain to securely record a dataset's lifecycle events such as creation, purchase and consumption.

With this objective in mind, this experiment encompasses the following scenarios:

- Dataset Lifecycle Record
- Dataset Provenance and Traceability Report

3.2.1 Scenario 01: Dataset Lifecycle Record

This scenario demonstrates the ability to record asset traceability events on-chain by leveraging blockchain-enabled smart contracts which allow secure and trusted algorithm execution.

Through the use of smart contracts, it is possible to develop functions which store a reference to an asset and events related to its lifecycle in an append-only decentralized ledger. The type of asset considered for this experiment is a dataset published in the AGORA data marketplace and Hyperledger Fabric constitutes the blockchain of choice. An event will be sent to the Data Provenance and Traceability application whenever a dataset is published or purchased through the AGORA data marketplace. Furthermore, a consumption event will be sent to the Data provenance and traceability application whenever the dataset is accessed through the related Data Container. The aforementioned events will be stored on-chain using a smart contract deployed in a Hyperledger Fabric channel. Keycloak and eIDAS will be utilized for access control and in particular to manage rights to smart contract functions which implement traceability.

Components participation	• eIDAS • Keycloak • Data Provenance and Traceability app • AGORA • Data Container
Datasets/AI Models	To be confirmed which ones
Proposed Added Value	• Trust: Due to the immutable and transparent nature of the blockchain, it is possible to ensure that stored records are tamper-proof and that members are able to access chain data from the first to the latest block and independently verify it.

	• Accountability: Transactions are signed with the private key of the transaction sender and transparently recorded within an append-only and immutable ledger, which ensures accountability of actions within the chain. • Security: The immutability and secure execution of smart contracts eliminates the need for centralized, trusted intermediaries and removes the possibility of tampering and manipulation of results.
KPIs	Number of datasets recorded: At least 2 dataset references will be recorded on-chain. Number of traceability events recorded: At least 4 traceability events will be recorded on-chain.
Partners Participation	INTRA, ATOS, UPV

3.2.2 Scenario 02: Dataset Provenance and Traceability Report

This scenario will serve as a test ground for the access and transparency of on-chain dataset references and their traceability events. A dashboard will be developed to present the data available on-chain in a user-friendly format. The data will be retrieved through interfaces implemented for Hyperledger Fabric and presented in a report which will enumerate in order the events related to the dataset, such as creation, purchase and access, from the point of its creation up to the present. Through the report, it will also be possible to trace the exchange of ownership of the related dataset. Keycloak and eIDAS will be integrated with Hyperledger Fabric for access control and in particular for the purpose of managing access rights to the Fabric channel where the related smart contract is deployed.

Components participation	• eIDAS • Keycloak • Data Provenance and Traceability application
Datasets/AI Models	To be confirmed which ones
Proposed Added Value	Transparency: The code defining a smart contract is publicly available and accessible for any party to verify. After its deployment it becomes immutable and secure execution

	within the blockchain ensures that the rules and conditions defined in it are carried out as intended. Availability: Through its decentralized architecture, blockchain may achieve high availability. Nodes are continuously operational and due to redundancy, the network remains accessible regardless of occasional node failure.
KPIs	Number of reports generated: At least 2 reports, one report per dataset, will be generated using the data available through the smart contract.
Partners Participation	INTRA

3.3 Dataset FAIRness assessment

Data FAIRness refers to the four principles of data being Findable, Accessible, Interoperable, and Reusable. FAIRness constitutes an essential pillar of data quality. Therefore, the aim of this experiment is to demonstrate how an automated FAIRness assessment tool may contribute to higher efficiency when evaluating a dataset by detecting deficiencies and evaluating the compliance with the aforementioned principles.

In pursuit of this objective, this experiment encompasses the following scenario:

- Automated dataset FAIRness assessment

3.3.1 Scenario 01: Automated dataset FAIRness assessment

This scenario will conduct the testing and evaluation of F-UJI, an automated FAIRness assessment tool. A dataset will be used as input to F-UJI and the results will be analyzed in order to identify which FAIRness principles are met and determine the optimal way of complying with them. First, the dataset and its metadata will be prepared and formatted appropriately for assessment by F-UJI. Next, the dataset will be given as input to F-UJI where it will be processed and assessed. The output consists of a score for each principle, details on how and whether it is satisfied, and a total FAIRness score. The results will be recorded and analyzed with the goal of implementing any modifications that may ameliorate the quality of the dataset and its metadata. In addition, the results will be manually verified to ensure their validity and accuracy.

Components participation	• F-UJI
Datasets/AI Models	To be confirmed which ones
Proposed Added Value	• Efficiency: While FAIRness assessment is a process that requires a degree of human effort, automated tools such as F-UJI may quickly and efficiently evaluate the FAIR aspects of an input dataset, leading to reduced processing times and increased throughput in cases where the volume of input data is high. • Consistency: The use of an automated FAIRness assessment tool guarantees a consistent evaluation across different datasets, facilitating comparing and contrasting of FAIRness compliance when there are multiple datasets. • Integration: The F-UJI tool implements a REST API which may be used to perform a FAIRness evaluation using as input a URL pointing to the dataset to be assessed. The results are returned in JSON format. This offers flexibility regarding the integration of the FAIRness assessment service in other tools and applications and may be used to process and present the results in a customized manner.
KPIs	• Number of datasets evaluated: at least one dataset will be evaluated for FAIRness.
Partners Participation	INTRA, GF and FRAISA

3.4 Data as a Product

Data as a Product methodology not only ensures data availability but also emphasizes its quality and usability, making it a valuable asset for informed decision-making and driving operational excellence. By treating data as a product, this approach fosters a culture of data-driven insights and continuous improvement, transforming historical data into a strategic resource that empowers the organization to adapt, optimize, and innovate in an ever-evolving landscape. This experiment aims to establish an open and transparent data ecosystem, where data and services can be provided, aggregated, and exchanged within a context of confidence and reliability.

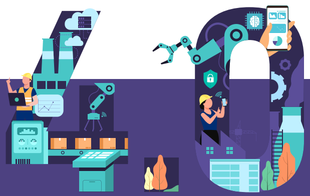
With that objective in mind, this experiment will examine the subsequent scenarios:

- Historical Data as a Product
- DaaP and Data Spaces
- DaaP and decentralized Machine Learning

3.4.1 Scenario 01: Historical Data as a Product

This scenario is in charge of providing access to the pilot's historical data following the Data as a Product approach. It provides an easy-to-use environment for sharing, collecting, and pre-processing data simplifies the entire workflow. This simplicity not only saves time and therefore reduces costs, but also enhances data quality and consistency. When historical data is well-prepared and standardized, it becomes a valuable resource for analysis and benchmarking, allowing the pilot to identify trends, patterns, and opportunities for improvement more effectively.

Components participation	• Data as Product Container (DC) • Data Connection Profile (DCP) • Data Ingestion • Data Transformation services • Keycloack • eIDAS • Data Container Monitoring
Datasets/AI Models	To be confirmed which ones
Proposed Added Value	• Trustworthy and truthful: The data should accurately portray the organization's actual state to be of value. To achieve this, various methods will be employed, including data purification and evaluating data quality. Additionally, incorporating data source and data history details within the metadata will further enhance trustworthiness. • Interoperability: Ensuring the existence and application of standards and harmonization rules is essential to facilitate the sharing and correlation of data across diverse domains. Standardization empowers data consumers to meaningfully correlate and merge data from disparate domains. • Security: For the secure sharing of data, it is imperative to guarantee the preservation of data confidentiality, availability, and integrity. This involves the essential implementation of data access control, wherein access control policies are thoughtfully established to match



	the precise requirements of each individual data product. • Data quality: Accurate and consistent data builds trust and confidence, ensuring that collaborating entities are working with a common, reliable foundation.
KPIs	• Number of data sources shared: at least 2 data sources will be shared via Data as a Product Containers. • Number of DCPs delivered: at least 2 DCPs will be delivered.
Partners Participation	UPV, CERTH, INTRA, S21SEC

3.4.2 Scenario 02: DaaP and Data Spaces

This scenario will demonstrate the use of the Data Containers in a Data Space to enable trusted and secure data sharing among different stakeholders in order to create data-driven solutions. The use of IDS-certified technologies will allow the different parties to remain in control of the data that they decide to share, while use of DC will guarantee that the data being shared meets the quality criteria of the consumers.

Components participation	• Data as Product Container (DC) • Data Connection Profile (DCP) • IDS connector • Hyperledger Fabric • Keycloak • eIDAS
Datasets/AI Models	To be confirmed which ones
Proposed Added Value	• Discovery: Enables the capability to access and utilize information independently, without requiring direct involvement from the data supplier. Additionally, metadata assumes a significant role in rendering data comprehensible, while also contributing to the formulation of data quality standards and policies for data usage. • Data Sovereignty: Maintaining control over data helps organizations to mitigate risks of data breaches and unauthorized access. It also allows them to uphold the integrity and quality of their data, protecting them

	against potential data degradation when data is shared or manipulated among multiple parties. • Trusted data sharing: Encourages collaboration between organizations. When entities have confidence in the integrity and security of shared data, they are more willing to work together, leading to synergistic efforts and enhanced outcomes. • Interoperability: Transforms isolated data repositories into interconnected ecosystems, where information flows freely, supporting collaborative research, innovation, and informed decision-making.
KPIs	• Number of data assets shared using IDS connectors: at least 2 data assets will be shared using Data Containers and IDS connectors.
Partners Participation	UPV, CERTH, INTRA

3.4.3 Scenario 03: DaaP and decentralized Machine Learning

In this scenario, ML is combined with data management under the DaaP paradigm. Data Containers act as safeguards for data quality, enable access to historical information, and facilitate data preprocessing to enhance the performance of federated machine learning algorithms. By employing Data Containers, a data quality assurance system is established, increasing the accuracy and robustness of the algorithms, thereby contributing to more effective machine learning models.

Components participation	• Data as Product Containers (DC) • Data Connection Profile (DCP) • Decentralised data management & analytics • ALIDA • Data Ingestion • Data Transformation services
Datasets/AI Models	To be confirmed which ones
Proposed Added Value	• Interoperability based on the use of standardized vocabularies and metadata, as well as the use of

	homogenised data formats and APIs to access the data. • Data quality: Accurate and consistent data builds trust and confidence, ensuring that collaborating entities are working with a common, reliable foundation. • Efficacy: Ability to train ML models in a collaborative fashion while preserving data privacy. • Efficiency: Ability to reach similar performance of centralised ML in terms of accuracy, without moving data from where they are generated. • Data Sovereignty: Maintaining control over data helps organizations to mitigate risks of data breaches and unauthorized access. It also allows them to uphold the integrity and quality of their data, protecting them against potential data degradation when data is shared or manipulated among multiple parties. • Reusability: enhanced through an extensible catalog of BDA services for ingestion, preparation and ML/DL. • Scalability: supported and enhanced by a micro-service based cloud-native platform. • Easier sourcing data, building models and pipelines integrated through a platform for BDA applications such as, but not limited to, FML participants and aggregator services.
KPIs	• Number of models trained: at least one model will be trained using the Data Container. • Accuracy: decentralised (federated) machine learning models implemented in the experiment reach at least 90% accuracy of corresponding algorithms trained in centralised (unfeasible) settings • Number of FML participant and aggregator BDA services deployed
Partners Participation	UPV, CNR, CERTH, ENG

3.4.4 Scenario 04: DaaP and AAS / Digital Twin

In this scenario, the digital twin or Asset Administration Shell (AAS) of a 5G network exposes information about the 5G network to external nodes or application. The 5G digital twin is integrated by two main components, the 5G User Equipment (UE) Digital Twin and the 5G Network Digital Twin. The 5G Network Digital Twin digitally models the characteristics of all nodes and functions integrating the 5G radio and core networks, such as, gNBs, User Plane Functions (UPF), Session Management Functions (SMF), etc. and edge computing nodes integrated in the 5G network.

In the considered scenario, an external cloud node will access data of an edge node of the 5G network through the Internet. In this context, Data Containers will be used to facilitate the exchange of data and interoperability between different components and guarantee the data quality.

Components participation	• Data as Product Containers (DC) • Data Connection Profile (DCP) • 5G AAS (UMH)
Datasets/AI Models	To be confirmed which ones
Proposed Added Value	• Interoperability guaranteeing interconnected ecosystems with data exchange among them, as well as the use of homogenised data formats and APIs to access the data. • Discovery: The AAS and digital twins will expose relevant data to other applications and the use of the Data Containers will facilitate the access and the use of the information without knowing the technical characteristics of the corresponding assets.
KPIs	Number of components connected using Data Containers
Partners Participation	UPV, UMH

3.5 Scalable industrial IoT Solution

In this experiment, SIEMENS' Insights Hub platform is connected and used within the TEF of the Swiss Smart Factory. Several demonstrators of the Test – and Demo platform of the Swiss Smart Factory are connected to the Insights Hub platform. From the demonstrators, data regarding the machine status, environmental data and energy related data will be provided. The aim of this experiment is to monitor and analyze this gathered data in the Insights Hub platform.

With that objective in mind, this experiment will examine the subsequent scenarios:

- Asset monitoring dashboard(s)
- Energy Management of the demo factory

3.5.1 Scenario 01: Asset monitoring dashboard(s)

The aim of the first Scenario is mainly to enable the monitoring of the environment (temperature, humidity, vibration and atmospheric pressure) in the TEF Swiss Smart Factory. By using the gathered data of the environment, optimization of the factory and improved business decisions can be made. Based on this, numerous dashboards and alerts will be implemented.

Components participation	• Insights Hub of SIEMENS • TEF Swiss Smart Factory
Datasets/AI Models	• Dataset with environmental data of the SSF TEF demonstrators of the drone factory
Proposed Added Value	• Enabling monitoring of environment in the factory • Creating alerts based on specified threshold values • Creating reports of measured data • Enabling optimization of the factory based on created reports. • Enable improved operational and business decisions with data-driven insights.
KPIs	• Number of dashboards implemented: At least 10 dashboards are created on the platform based on the environmental data. • Number of alerts implemented: At least 20 alerts are implemented on the platform based on the environmental data.
Partners Participation	SIEMENS, SIPBB

3.5.2 Scenario 02: Energy Management of the demo factory

In the second scenario, an energy management of the TEF Swiss Smart Factory will be executed. The connected assets of the Swiss Smart Factory are providing specific energy related data which will be managed and monitored in the energy manager app of the Insight Hub platform. This will enable several added values like the identification of anomalies and the provision of a high transparency about the energy consumption.

Components participation	• Insights Hub of SIEMENS • TEF Swiss Smart Factory
Datasets/AI Models	• Energy related dataset of the SSF TEF demonstrators of the drone factory
Proposed Added Value	• Enabling power monitoring of the factory • Transparency about energy consumption over the factory • Enabling identification of anomalies • Optimization of the energy consumption • Batch and material related energy analyzation
KPIs	• Emissions measured: Energy emission data are measured for each connected asset in kgCO2. • Number of reports generated: At least 1 report per connected asset per month is created based on the energy consumed and emissions generated
Partners Participation	SIEMENS, SIPBB

3.6 Comparative Analysis Between Centralized, Local, and Federated Machine Learning for Predictive Maintenance Tasks: Privacy, Performance, and Efficiency.

This experiment aims to compare centralized, local, and federated computing methods in the context of machine learning and AI for predictive maintenance tasks (e.g., remaining useful life, anomaly detection), and it aims to provide valuable insights into the respective advantages and disadvantages of each approach.

This experiment will consider three scenarios:

- Use local computing method: The machine learning model will be implemented on each edge device using localized data.
- Use centralized computing method: In this scenario, a centralized machine learning model will be developed from data collected from all edge devices.
- Use federated learning method: Machine learning applications will aggregate parameters from all edge devices without centralizing raw data.

3.6.1 Scenario 01: Local computing for machine learning applications implemented for each edge device.

In this scenario a Docker image, containing a machine learning algorithm will be shared with stakeholders responsible for edge devices. The stakeholders will train the model based on instructions and share performance metrics after the implementation.

Components participation	• CORE: Docker Image Creation (Develop and package the machine learning model in python programming language within a Docker image). • CORE: Docker Image Sharing (Share the Docker image with stakeholders, allowing them to train the model with their own data locally).
Datasets/AI Models	AI model (To be confirmed): Remaining Useful File/ Anomaly Detection. Datasets: timeseries data from Milling Tool.
Proposed Added Value	• Dockerized learning allows for portable and consistent model deployment. • Stakeholders can train the model with their own data locally, preserving data privacy.
KPIs	• Model Performance: Evaluate the performance of each local model using appropriate metrics, tailored to the specific machine learning task. • Docker Image Size: To assess efficiency and portability. • User/Organization Satisfaction: Feedback on Dockerized model sharing and local training.
Partners Participation	CORE: Software for data manipulation and machine learning algorithm implementation. Docker Image creation

	and sharing with stakeholders. GF and FRAISA: Collaborating Stakeholders

3.6.2 Scenario 02: Centralized Machine Learning Implementation

In this scenario, a centralized approach is implemented for machine learning model development. The central model is trained on all available data, and performance metrics are collected after implementation. This approach allows leveraging data from all edge devices for predictive maintenance tasks.

Components participation	CORE: Centralized Model Training.
Datasets/AI Models	AI model (To be confirmed): Remaining Useful File/Anomaly Detection. Datasets: timeseries data from Milling Tools
Proposed Added Value	• Centralized computing allows for in-depth data analysis and comprehensive model training, benefiting from a larger and more diverse dataset. • Centralized systems can efficiently scale computational resources as needed, ensuring optimal model training and performance.
KPIs	• Model Performance: Evaluate the performance of the centralized model using appropriate metrics, tailored to the specific machine learning task. • Resource Efficiency: Measure the computational resources and time required for centralized model training and data collection.
Partners Participation	CORE: Software for data manipulation and machine learning algorithm implementation. GF and FRAISA: Collaborating Stakeholders, providing the data.

3.6.3 Scenario 03: Federated learning for machine learning applications by using flower framework.

That scenario includes the implementation of a federated learning system that distributes the machine learning model to the data sources, ensuring compliance with regional privacy regulations. The data distribution and model aggregation process are executed as follows:

- Federated Learning Framework: We deploy the Flower framework, which acts as the orchestrator for federated learning. Flower facilitates communication and coordination among edge devices and the server.
- Model Initialization: A global machine learning model is initially provided to each edge device as a starting point for training.
- Local Model Training: Edge devices independently train the model using their own data.
- Model Updates: After local training rounds, each edge device generates model updates based on the knowledge gained during training.
- Secure Communication: Model updates are securely transmitted to the central aggregator, maintaining data privacy during transmission.
- Aggregation Server: The central aggregator receives model updates from all participating edge devices. Those updates are aggregated to construct a global model that encapsulates knowledge from all edge devices.
- Model Distribution: The global model is then redistributed to all participating edge devices, serving as a new starting point for subsequent training rounds.

Components participation	CORE: Create software for federated learning framework.
Datasets/AI Models	AI model (To be confirmed): Remaining Useful File/ Anomaly Detection. Datasets: timeseries data from Milling Tools
Proposed Added Value	• Encourages collaboration while respecting data protection laws. • It potentially leads to better generalization of the ML model, in comparison with the local approach as it uses information from multiple sources and at the same time preserves data security.
KPIs	• Model Evaluation: Evaluation of the federated model's performance compared to local models' performance. • Training Time: Time required for federated model convergence.

	• User/Organization Satisfaction: Feedback on federated learning application.
Partners Participation	• CORE: Federated Learning Infrastructure Provider. GF and FRAISA: Collaborating Stakeholders

4 Conclusion

The collection of technical descriptions of tools encompassed in the first-generation digital continuum 4.0 open toolkit presents a thorough exploration of their functionality and capabilities. It constitutes a significant source of knowledge which may aid partners in acquiring a more in-depth understanding of the inner workings of the respective tool as well as in facilitating its utilization and integration in other components of the toolkit.

Following the analysis of the aforementioned components, the introduced experimentation plans formulate a space and plan for component testing and validation within the network of TEFs and pilot sites. Through the delineation of the steps described in each scenario it is possible to better infer the purpose and use cases of each tool and to support the first steps of communication between different components and collaboration between different partners.

In conclusion, this deliverable constitutes a foundation for the implementation of the digital 4.0 continuum open toolkit. The comprehensive documentation of tools and experimentation plans outlined in this document will provide a higher degree of clarity and insights into the functionality, performance and interaction of the tools and support the integration processes to be conducted within WP3 and related tasks.