



Funded by  
the European Union



## D5.1 | CyclOps Platform Release (IT-1)

<b>Project ref. no.</b>	<b>HORIZON-CL4-2023-DATA-01 / GA No 101135513</b>
<b>Project title</b>	Automated end-to-end data life cycle management for FAIR data integration, processing and re-use
<b>Duration of the project</b>	01-01-2024 – 31-12-2026 (36 months)
<b>WP/Task:</b>	WP5
<b>Deliverable Type:</b>	DEM
<b>Dissemination level:</b>	PUBLIC
<b>Document due Date:</b>	30/06/2025 (M18)
<b>Actual date of delivery</b>	01/07/2025
<b>Leader of this deliverable</b>	FDI
<b>Other Contributing Partners</b>	FIWARE, ATOS, UPC, TUBS, CERTH, BSC, SUITE5, DC, ONTOPIIC, NTTDES, NUIDUCD-CeADAR
<b>Version</b>	V1.0

## Document History

Version	Date	Document history/approvals
<b>0.1</b>	21/11/2024	1st version of the Table of Contents
<b>0.2</b>	11/06/25	Contributions in Sections 2, 3, 4 and Annex from authors
<b>0.3</b>	18/06/2025	Contributions in Sections 4 and 5 from authors
<b>0.4</b>	22/06/2025	Updates on Sections 2 and Annex
<b>0.5</b>	23/06/2025	Document finalization for Quality Control and Peer-review
<b>0.6</b>	26/06/2025	Peer-review and Quality Control completed. Fine-tuning of the document.
<b>1.0</b>	01/07/2025	Final version to be submitted

## List of authors, contributors and reviewer(s)

Name	Partner (Acronym)	Contributions/comments
<b>AUTHORS AND CONTRIBUTORS</b>		
<b>Alexandros S. Kalafatelis, Alina Pitsiakou, Panagiotis Trakadas</b>	FDI	Main editors. Contributors in Sections 1, 2, 3, 4, a 5,6,7 and 8.
Alberto Abella, Tim Smyth	FIWARE	Contributors Sections 2.1, 2.2, 2.3, 3, 6, 7 and related Annex
Alejandro Garcia, Ross Little	ATOS	Contributors in Sections 2.2, 3, 6, 7 and Annex
Aniol Bisquert, Marc Maynou, Gerard Pons, Anna Queralt, Jordi Garcia, Xavi Masip-Bruin, Josep Llosa	UPC	Contributors in Sections 3, 4, 6, 7, and Annex
Admela Jukan, Ítalo Brasileiro, Urslla Izuazu	TUBS	Contributors in Sections 6, 7 and Annex
Alexandros Oikonomidis, Ilias Gialampoukidis, Anastasia Moutzidou	CERTH	Contributors in Sections 3, 6, 7 and Annex
Javier Conejero, Alex Barceló	BSC	Contributors in Section 7
Spiros Kousouris	SUITE5	Contributor in Section 6 and Annex
Saeed Talebzadeh	DC	Contributor in Sections 6 and 7
Benjamin Cogrel	ONTOPIC	Contributor in Sections 6 and 7
Maria Font, Rocio Peral	NTTDES	Contributor in Sections 3,6,7, and Annex
Saman Cooray, Vikas Ojha	NUIDUCD-CeADAR	Contributor in Sections 6,7 and Annex
<b>REVIEWERS</b>		
Spiros Kousouris	SUITE5	Internal Review
Anna Queralt	UPC	STC Review and Approval
Monica Caballero	NTTDES	PC Review and Approval

**DISCLAIMER**

Funded by  
the European Union

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission as the granting authority. Neither the European Union nor the granting authority can be held responsible for them.

The authors of this document have taken any available measure for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any sort of responsibility that might occur because of using its content.

Copyright ©CyclOps Consortium, 2025

This work is licensed under the Creative Commons License “BY-NC-SA”.



## Table of contents

DOCUMENT HISTORY.....	2
LIST OF AUTHORS, CONTRIBUTORS AND REVIEWER(S).....	2
TABLE OF CONTENTS .....	4
LIST OF FIGURES .....	7
LIST OF TABLES .....	7
LIST OF ACRONYMS .....	8
EXECUTIVE SUMMARY.....	10
<b>1. INTRODUCTION .....</b>	<b>11</b>
1.1. PURPOSE OF THE DOCUMENT AND SCOPE .....	11
1.2. CONTEXT .....	11
1.3. STRUCTURE OF THE DOCUMENT.....	12
<b>2. BRIDGING CYCLOPS AND DATA SPACES.....</b>	<b>13</b>
2.1. SEMANTIC INTEROPERABILITY .....	14
2.1.1. <i>Introduction</i> .....	14
2.1.2. <i>Implementation</i> .....	14
2.1.3. <i>Progress Report</i> .....	14
2.2. DATA EXCHANGE.....	15
2.2.1. <i>Introduction</i> .....	15
2.2.2. <i>Implementation</i> .....	15
2.2.3. <i>Progress Report</i> .....	16
2.3. PUBLICATION.....	17
2.3.1. <i>Introduction</i> .....	17
2.3.2. <i>Implementation</i> .....	18
2.3.3. <i>Progress Report</i> .....	18
2.4. DECENTRALISED IDENTITY, TRUST AND ACCESS CONTROL .....	19
2.4.1. <i>Introduction</i> .....	19
2.4.2. <i>Implementation</i> .....	22
2.4.3. <i>Progress Report</i> .....	27
<b>3. LIST OF COMPONENTS INCLUDED IN CYCLOPS RELEASE 1 .....</b>	<b>28</b>
<b>4. CYCLOPS TESTING APPROACHES .....</b>	<b>31</b>
4.1. TESTING PHASES .....	31
4.2. INTEGRATION TESTING APPROACHES.....	32
4.3. CYCLOPS TESTING STRATEGY .....	34
4.3.1. <i>Testing Requirements</i> .....	34
4.3.2. <i>Unit Testing Framework</i> .....	34
4.3.3. <i>Performance Benchmarking</i> .....	35

4.3.4.	<i>Docker and Deployment Testing</i> .....	35
4.3.5.	<i>API Documentation and Validation</i> .....	36
4.3.6.	<i>Integration Testing Requirements</i> .....	36
4.3.7.	<i>Best Practices Implementation</i> .....	36
<b>5.</b>	<b>CYCLOPS INTEGRATION AND PLATFORM DEPLOYMENT</b> .....	<b>36</b>
5.1.	CYCLOPS ASSETS .....	37
5.1.1.	<i>Integration Plan</i> .....	37
5.1.2.	<i>Infrastructure Assets</i> .....	37
5.1.3.	<i>CI/CD Best Practices</i> .....	38
5.1.4.	<i>Code Review and Quality Assurance</i> .....	38
5.2.	NETWORK ARCHITECTURE AND SERVICE ISOLATION .....	38
5.3.	CI/CD UTILIZING GITLAB RUNNERS .....	38
5.3.1.	<i>Logging and Observability</i> .....	40
5.3.2.	<i>Version Tagging Policy</i> .....	41
5.3.3.	<i>Requirements For Integration</i> .....	42
5.3.4.	<i>Integration Workflow</i> .....	42
<b>6.</b>	<b>INTEGRATION AND TESTING RESULTS FOR CYCLOPS IT-1</b> .....	<b>44</b>
6.1.	UNIT TEST MATRIX .....	44
6.2.	MODULE INTEGRATION MATRIX .....	45
6.2.1.	<i>User Intent Layer</i> .....	46
6.2.2.	<i>Runtime Layer</i> .....	47
6.2.3.	<i>Knowledge Layer</i> .....	50
6.2.4.	<i>Interoperability Layer</i> .....	51
<b>7.</b>	<b>REQUIREMENTS EVALUATION</b> .....	<b>53</b>
7.1.	PLATFORM .....	53
7.2.	USER INTENT LAYER .....	53
7.3.	RUNTIME LAYER .....	54
7.3.1.	<i>DataOps</i> .....	54
7.3.2.	<i>AIOps</i> .....	56
7.3.3.	<i>Data and Execution Abstraction</i> .....	58
7.4.	KNOWLEDGE LAYER .....	59
7.5.	INTEROPERABILITY LAYER .....	60
<b>8.</b>	<b>CONCLUSIONS</b> .....	<b>61</b>
	<b>ANNEX: LIST OF MODULES INCLUDED IN CYCLOPS RELEASE 1</b> .....	<b>62</b>
A.1.	USER INTENT LAYER .....	62
A.1.1.	<i>User-assisted Intent Interface</i> .....	62
A.1.2.	<i>Natural Language Intent Interface</i> .....	62
A.1.3.	<i>Intent Compiler</i> .....	63

A.1.4. <i>Exploratory Analysis Engine</i> .....	63
A.1.5. <i>Orchestrator</i> .....	63
A.2. RUNTIME LAYER .....	64
A.2.1. <i>DataOps</i> .....	64
A.2.2. <i>AIOps</i> .....	67
A.2.3. <i>DEA</i> .....	69
A.3. KNOWLEDGE LAYER .....	69
A.3.1. <i>IKB Metadata Manager</i> .....	69
A.3.2. <i>Concepts Extraction and Categorization Engine</i> .....	70
A.3.3. <i>IKB Exploitation tool. Natural Language Understanding (NLU)</i> .....	70
A.4. INTEROPERABILITY LAYER .....	72
A.4.1. <i>SSI Wallet</i> .....	72
A.4.2. <i>SSI Agent</i> .....	72
A.4.3. <i>VC Verifier</i> .....	72
A.4.4. <i>Data Access</i> .....	73
A.4.5. <i>Semantic Interoperability</i> .....	73
A.4.6. <i>Data Exchange Broker</i> .....	74

## List of Figures

Figure 1. Overview of the current and expected future modules and components used in the Interoperability Layer .....	13
Figure 2. Interoperability layer SSI Components .....	21
Figure 3. Issue P2M SSI Credential .....	21
Figure 4. P2M SSI Verification .....	22
Figure 5. M2M SSI issuer & Verifier .....	22
Figure 6. Keycloak Issuer flow .....	24
Figure 7. Verification flow .....	25
Figure 8. OID4VP M2M Flow Sequence Diagram: The access of a dataset in the dataspace including its persistence. The Data Provider FIWARE DSC contains the VCVerifier component as marked in D2.2. ....	27
Figure 9. CyclOps CI/CD Workflow .....	31
Figure 10. Testing Phases: Unit, Integration, and Acceptance .....	31
Figure 11. Big-Bang Approach .....	33
Figure 12. Top-Down Approach .....	33
Figure 13. Bottom-Up Approach .....	34
Figure 14. Example UT output of the Preprocessing and Cleaning component .....	35
Figure 15. CyclOps CI/CD with GitLab .....	39
Figure 16. Integration workflow for the CyclOps Project .....	40
Figure 17. Example of GitLab Runner deploying docker environment, retrieves source code, runs test scripts, and verifies build success for CI, based on the preprocessing and cleaning component. ...	40
Figure 18. Centralized container logs. Example of working output of the Preprocessing and Cleaning component using the iris dataset without error logs .....	41
Figure 19. Example Dockerfile for the Preprocessing and Cleaning component with explicit versioning and environment consistency .....	42
Figure 20. CyclOps Integration workflow .....	43
Figure 21. High-level architecture of User Intent Layer (from D4.1) .....	46
Figure 22. High-level architecture of Runtime Layer (from D3.1) .....	47
Figure 23. High-level architecture of Knowledge Layer (from D4.1) .....	50

## List of Tables

Table 1. Components Included in CyclOps Release 1. ....	30
Table 2. Unit Testing vs. Integration Testing. ....	32
Table 3. Unit Test Matrix for CyclOps IT-1 .....	45
Table 4. Integration endpoints for User Intent Layer components. ....	47
Table 5. Integration endpoints for DataOps components. ....	48
Table 6. Integration endpoints for AIOps components .....	49
Table 7. Integration endpoints for DEA components. ....	50
Table 8. Integration endpoints for Knowledge Layer components. ....	51
Table 9. Integration points for the Interoperability Layer components .....	52
Table 10. Requirements for the Platform .....	53
Table 11. Requirements for the User Intent Layer .....	54
Table 12. Requirements for the DataOps .....	55
Table 13. Requirements for the AIOps .....	57
Table 14. Requirements for the DEA. ....	58
Table 15. Requirements for the Knowledge Layer. ....	60
Table 16. Requirements for Interoperability Layer .....	60

## List of acronyms

Abbreviation	Meaning
<b>AI</b>	Artificial Intelligence
<b>API</b>	Application Programming Interface
<b>AT</b>	Acceptance Test
<b>BERT</b>	Bidirectional Encoder Representations from Transformers
<b>CI/CD</b>	Continuous Integration and Continuous Delivery
<b>CSV</b>	Comma-separated values
<b>DCP</b>	Decentralized Claims Protocol
<b>DEA</b>	Data and Execution Abstraction
<b>DEE</b>	Distributed Execution Engine
<b>DID</b>	Decentralized Identifier
<b>DOME</b>	Distributed Open Marketplace for Europe
<b>DSBA</b>	Data Spaces Business Alliance
<b>DSP</b>	Data Space Protocol
<b>EBSI</b>	European Blockchain Services Infrastructure
<b>EDC</b>	Electronic Data Capture
<b>eIDAS</b>	Electronic Identification, Authentication and Trust Services
<b>ESSIF</b>	European Self Sovereign Identity Framework
<b>EUDIW</b>	EU Digital Identity Wallet
<b>FAIR</b>	Findable, Accessible, Interoperable, and Reusable
<b>GDPR</b>	General Data Protection Regulation
<b>HPC</b>	High Performance Computing
<b>IAM</b>	Identity and Access Management
<b>IKB</b>	Integrated Knowledge Base
<b>IKB-CECE</b>	IKB Concept Extraction and Categorization Engine
<b>IoT</b>	Internet of Things
<b>IT</b>	Integration Test
<b>JSON</b>	JavaScript Object Notation
<b>JWT</b>	JSON Web Token
<b>LSDM</b>	Large-Scale Data Management
<b>M2M</b>	Machine-to-Machine
<b>ML</b>	Machine Learning
<b>NER</b>	Named Entity Recognition
<b>NGSI</b>	Next Generation Service Interface



Abbreviation	Meaning
<b>NGSI-LD</b>	Next Generation Service Interface-Linked Data
<b>NLP</b>	Natural Language Processing
<b>OID4VCI</b>	OpenID for Verifiable Credential Issuance
<b>OID4VP</b>	OpenID for Verifiable Presentations
<b>OPA</b>	Open Policy Agent
<b>OWL2QL</b>	QL profile of the Web Ontology Language version 2
<b>P2M</b>	Person-to-Machine
<b>PAP</b>	Policy Administration Point
<b>PDP</b>	Policy Decision Point
<b>PEP</b>	Policy Enforcement Point
<b>R2RML</b>	RDB to RDF Mapping Language
<b>RDF</b>	Resource Description Framework
<b>SAGE</b>	Semantic-Aware Global Explanations
<b>SDM</b>	Smart Data Models
<b>SHACL</b>	Shapes Constraint Language
<b>SIOP-2</b>	Self-Issued OpenID Provider version 2
<b>SME</b>	Small and Medium-sized Enterprises
<b>SPARQL</b>	SPARQL Protocol and RDF Query Language
<b>SQL</b>	Structured Query Language
<b>SSI</b>	Self-Sovereign Identity
<b>TTL</b>	Terse RDF Triple Language
<b>UC</b>	Use Case
<b>UT</b>	Unit Test
<b>VC</b>	Verifiable Credentials
<b>VM</b>	Virtual Machine

## Executive summary

---

This deliverable provides a clear overview of the Interoperability Layer as well as of the first integrated release of the CyclOps platform (IT-1), a framework designed to enable intelligent, automated management of the entire data life cycle. CyclOps places particular emphasis on supporting FAIR data principles and ensuring smooth interoperability within and across new and existing data spaces.

Specifically, this deliverable provides details on the development and implementation of the Interoperability Layer, which are key outcomes of WP5 (T5.1, T5.2, T5.3). Additionally, this deliverable also constitutes the first release of CyclOps components (IT-1) and serves as a practical reference for users. The deliverable outlines the approach adopted for the design, implementation, integration, and deployment of the CyclOps platform components, which have been developed and documented in previous deliverables (i.e., D2.2, D3.1, D4.1). It explains the methodologies utilized to ensure coherent and robust integration across distributed and heterogeneous data environments.

In this context, detailed mappings for each technical layer of the platform have been defined to showcase the level of integration achieved for this iteration. Additionally, all project requirements specified in D2.2 have been mapped to each platform layer to demonstrate how the platform addresses them in this release. Dedicated mechanisms for Continuous Integration and Continuous Deployment (CI/CD) have been established to enhance continuous development and quality assurance. The CI/CD mechanisms will be continuously refined and extended in future iterations to accommodate new features and to maintain high standards of code quality, security, and operational stability. Lastly, Annex 1 provides additional details on the components included in IT-1, along with links to their README files and to the CyclOps Repository.

## 1. Introduction

---

### 1.1. Purpose of the document and scope

The primary aim of the D5.1 deliverable is to provide a supporting document that will guide the readers through the components composing the first release for the CyclOps platform. It is designed to guide CyclOps users and the UCs to navigate the detailed architecture, functionalities, and deployment of the CyclOps modules.

The document presents the resulting efforts of partners in transitioning from conceptual design to practical implementation of the platform, demonstrating how initially scarce, siloed components have been integrated into a coherent, robust framework for intelligent and automated management of the entire data life cycle. In addition, this deliverable performs as a benchmark for monitoring project progress, showcasing the consortium's achievements regarding the integration of the developed components.

The document provides comprehensive insight into the integration approach, including design decisions, interface mappings, and practical implementation details to ensure that independently developed modules are smoothly integrated end-to-end. The integration of components across the four fundamental layers of the CyclOps architecture, User Intent, Runtime, Knowledge, and Interoperability is facilitated through well-defined interfaces, shared data models, and standardized APIs. This structure enables seamless communication and data flow between components.

This document outlines the first release, while also setting a foundation for ongoing and future releases.

### 1.2. Context

The CyclOps initiative tackles the urgent and critical need to govern the end-to-end data life cycle to enable advanced, trustworthy, secure and interoperable AI-based data-driven applications for various domains and stakeholders. To achieve this, CyclOps delivers an integrated platform that automates the end-to-end data life cycle, including ingestion, curation, sharing, and AI-driven exploitation, while adhering to FAIR data principles and aligning with European Data Space standards.

This deliverable, D5.1, reports on the first integrated release of the CyclOps platform. It assembles the requirements and architecture developed in WP2, the essential AI and data-related modules produced in WP3, the governance and human-centric tools offered in WP4, and the integration components developed in WP5. It builds on the early requirements and architectural designs outlined in D2.1 (Requirements and State-of-the-Art Analysis), D2.2 (CyclOps Architectural Design), D3.1(AIOps, DataOps, and DEA Components Development (IT-1)) and D4.1(Knowledge Layer and Use Intent Layer). Together, these coordinated efforts demonstrate the transition from conceptual design to a working prototype, which will be validated through four use cases and supports continuous evolution towards the project's final objectives.

Within WP5, this deliverable describes the work performed to ensure interoperability with data spaces and standardized data exchange under Task T5.1. It also covers the design and implementation of decentralized identity, trust, and access control mechanisms in Task T5.2, the integration with the Distributed Open Marketplace for Europe Cloud and Edge Services (DOME) addressed in Task T5.3, and the in-lab testing and preliminary validation activities carried out in Task T5.4. Finally, the deliverable reports on the overall platform integration and release management conducted in Task T5.5. These collective efforts ensure that the platform's modular components operate cohesively, deliver reliable performance, and could be validated under realistic scenarios, paving the way for IT-2 and continuous enhancement of the CyclOps platform.

### 1.3. Structure of the document

This document is structured in the following way:

- **Section 2** describes how CyclOps connects and integrates with data spaces, covering key aspects such as data interoperability, decentralized identity, trust and access control, and integration with the EU Marketplace (DOME). It provides information regarding the Interoperability Layer of the CyclOps platform, presents the current status of IT-1, discusses the main challenges addressed, and outlines the next steps planned to enhance seamless data space integration.
- **Section 3** offers a comprehensive overview of all components included in CyclOps IT-1 platform release, detailing the responsible partners, a brief description of each component, and its associated work package.
- **Section 4** introduces the testing and integration approaches that were considered and explains the strategy adopted for CyclOps, including also a practical example of Unit Testing.
- **Section 5** introduces the integration and deployment workflows, including a practical example of the CI/CD pipeline supporting continuous integration and deployment.
- **Sections 6** provides a thorough, layer-driven mapping of the components and their interactions, showcasing the current testing and integration status.
- **Section 7** reviews the project requirements for each platform layer, assessing readiness and defining next steps for IT-2.
- **Section 8** concludes the document with final remarks and outlines the next steps for further development, integration, and validation of CyclOps.

## 2. Bridging CyclOps and Data Spaces

The Interoperability Layer acts as the central enabler for seamless integration and interaction between the CyclOps platform and Data Spaces. It ensures that identity, data access, semantics and data exchange processes are aligned with open standards and are ready for scalable and trustworthy operation. The layer is split into the following modules:

- **Identity:** This component ensures secure and verifiable identification of entities (organizations, users, or systems) using technologies like Verifiable Credentials and Decentralized Identifiers (DIDs). It forms the foundation for trust in digital interactions.
- **Data Access:** Access control mechanisms govern who can view or modify data, supporting fine-grained authorization via modern protocols such as OAuth2 and OID4VP, tailored for both person-to-machine (P2M) and machine-to-machine (M2M) flows.
- **Semantic Interoperability:** Common data models (definition of a physical or logical object, refers to those available at the Smart Data Models initiative<sup>1</sup>) and shared vocabularies (e.g., NGSI-LD and DCAT-AP) ensure that data from different sources can be understood and processed consistently, fostering alignment across domains and services.
- **Data Exchange:** The layer facilitates the secure and standardized exchange of data between participants – both in the CyclOps platform and in external data spaces - using APIs, connectors and established protocols, ensuring traceability and integrity of transactions.
- **Publication:** Data can be made discoverable and accessible through registries or marketplaces, enabling reuse and visibility across organizational boundaries. This includes support for publishing datasets to platforms like the DOME marketplace under defined policies.

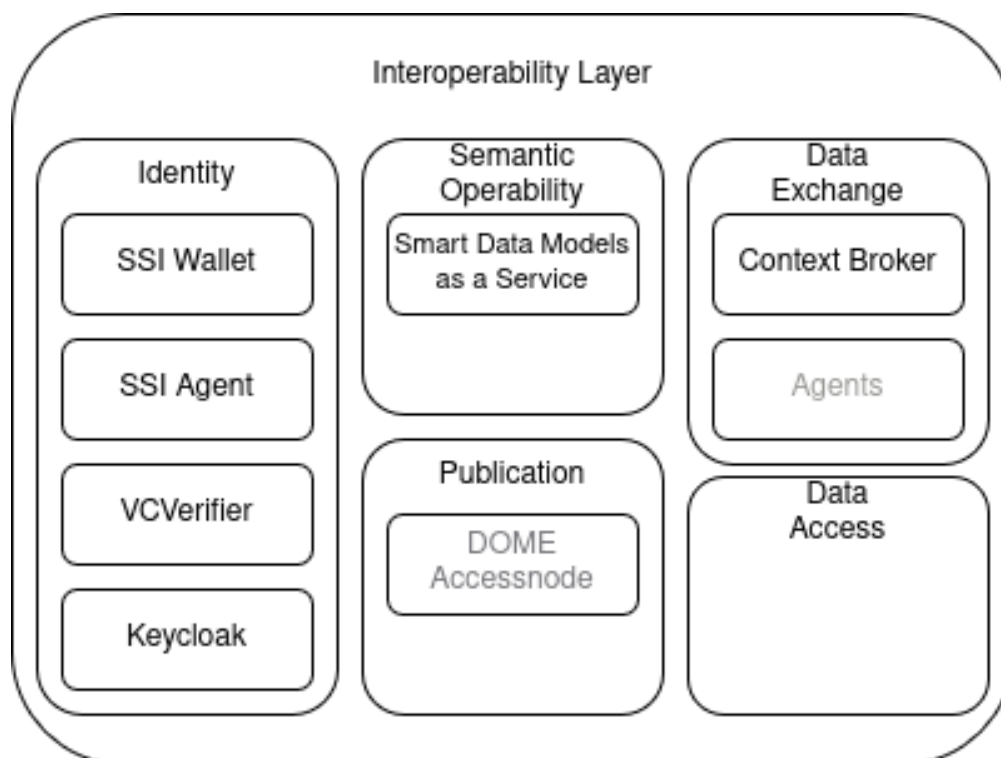


Figure 1. Overview of the current and expected future modules and components used in the Interoperability Layer

<sup>1</sup> <https://smartdatamodels.org> An initiative for the curation of open-licensed data models based on actual examples and open and adopted standards

## 2.1. Semantic Interoperability

### 2.1.1. Introduction

To achieve **semantic interoperability**, datasets retrieved by the data exchange module are harmonized using **Smart Data Models**, ensuring consistent interpretation of domain-specific information across participants.

### 2.1.2. Implementation

#### 2.1.2.1. Smart Data Models as a Service

##### Purpose and Role

The aim of the Smart Data Models as a Service component within the Semantic Interoperability block is to provide a validation of the contents to be shared between the CyclOps platform and external sources like data spaces in iteration 1 and also a service marketplace in iteration 2. The main function is to accept a row of a dataset or json payload, currently retrieved through a URL, in example the one to retrieve the information from a Context Broker. This service looks for the data model type of the entry to find relevant smart data models. And once found, validate the entry's compliance with those data models.

This is the main function, but other semantic functions have already been included in the package, see the list below.

##### Key functions

At present, we have successfully implemented 9 functions. The functions revolve around data models and subjects (group of data models related by a common topic, grouped in folders in the Smart Data Models initiative). These 9 functions are:

- i) /validate-url. It validates a JSON payload from a URL against Smart Data Models
- ii) /subjects. It lists all available subjects for the data type of the payload
- iii) /datamodels/{subject\_name}. It lists the data models for a known subject
- iv) /datamodels/{subject\_name}/{datamodel\_name}/attributes. It prints attributes of a data model (in example those for the corresponding type of the payload)
- v) /datamodels/{subject\_name}/{datamodel\_name}/example it creates an example of a payload matching a data model in a subject
- vi) /search/datamodels/{name\_pattern}/{likelihood}. It searches for data models by approximate name based on the likelihood paramter (between 0 and 100) where 100 is exact match
- vii) /datamodels/exact-match/{datamodel\_name} It looks for a data model by exact name
- viii) /subjects/exact-match/{subject\_name} the same function to look for an exact match in the subjects
- ix) /datamodels/{datamodel\_name}/contexts it returns the @context(s) for a data model name, with all the semantic references for the terms used across the data model

##### Integration in the Architecture

The service offers its functions to all components in the CyclOps platform and can be integrated for importing and validating data where needed. Besides this, for iteration 2, other search functions will be also implemented. Some of these functions could include: (i) looking the description of an attribute; (ii) look for the units of an attribute; and (iii) retrieve the metadata associated to a data model (schema, examples, specifications, etc).

### 2.1.3. Progress Report

##### Current Status

The foundation for semantic interoperability has been established with the deployment of the **Smart Data Models as a Service** application. This service provides access to shared and reusable data models aligned with NGSI-LD, enabling consistent interpretation of data across systems. It supports

the validation and mapping of entities to recognized semantic structures and ensures that data can be integrated into broader interoperable ecosystems.

### Next Steps

In the upcoming period, the service will be **extended with additional functionality** based on evolving project requirements. This may include enhanced model management capabilities, support for domain-specific extensions, and improved interfaces for developers and data providers.

Moreover, the **integration of the service into the data ingestion flows** is a key priority. By embedding semantic validation and transformation into the ingestion pipeline, data consistency and interoperability will be enforced at the point of entry, ensuring alignment with platform-wide standards from the outset.

## 2.2. Data Exchange

### 2.2.1. Introduction

The **Data Exchange Module** ensures that data exchanged across diverse systems and data spaces can be meaningfully understood, integrated and utilized within the CyclOps platform. At its core, an **NGSI-LD Context Broker** serves as the central component for storing and managing structured data in a standardized and queryable format. To fill the NGSI-LD Context broker, **Agents** are used as go-betweens to external data sources. Their main job is to **retrieve, transform and ingest** data into the **NGSI-LD Context Broker**. They ensure this data is semantically consistent and interoperable, regardless of its original format.

### 2.2.2. Implementation

#### 2.2.2.1. Context Broker

In the CyclOps project, the **NGSI-LD Context Broker**, namely **Scorpio**<sup>2</sup>, serves as the **intermediate storage and processing layer** for integrating data from multiple sources, including data spaces based on FIWARE data space connectors and others.

### Purpose and Role

The Scorpio Context Broker acts as a temporary but structured data hub within the data interoperability layer. It enables real-time and historical access to contextual data by ingesting, normalizing, and storing information retrieved from various trusted sources. This includes datasets pulled via secure Machine2Machine (M2M) flows (e.g., through OID4VP used in the Data Space Retrieval Tool, as mentioned in chapter 2.4.2.2), as well as additional data feeds. Additionally, data that is to be published to data spaces shall be stored here.

### Key Functions

- i) **Data Aggregation:** Consolidates contextual data from heterogeneous sources into a unified NGSI-LD model.
- ii) **Semantic Interoperability:** Maintains semantic consistency using standardized data structures and ontologies, making the data ready for further use across services and systems.
- iii) **Temporary Processing Store:** Acts as a staging layer where data can be validated, enriched, or transformed before being forwarded to downstream systems (e.g., Data Ingestion Functions or Preprocessing functions of the Runtime Layer).
- iv) **Subscription and Query Interface:** Supports NGSI-LD subscriptions and queries, enabling reactive or pull-based data access by other components of the CyclOps platform.

---

<sup>2</sup> <https://scorpio.rtfd.io/> Scorpio is an NGSI-LD Broker that allows managing and requesting context information. Context Producers can manage their context – creating, updating, appending and deleting context information.



### Integration in the Architecture

- Serves as the receiving endpoint for data ingested via the Data Space Retrieval Script from the dataspace. Enables data consumers within the platform to access a harmonized, semantically rich view of external and internal data.
- Enables the Distributed Open Marketplace for Europe (DOME) stack to integrate data and act as data provider

### Advantages of Using Scorpio

- i) Fully compliant with the NGSI-LD specification (ETSI ISG CIM).
- ii) Scalable and modular architecture suited for distributed systems.
- iii) Well-supported within the FIWARE ecosystem, ensuring compatibility with other dataspace components.

By leveraging Scorpio as the context broker, the project ensures a robust, flexible and standards-based approach to intermediate data storage and processing, forming a key pillar in the overall interoperability strategy.

#### 2.2.2.2. Agents

##### Purpose

**Agents** are crucial components within the CyclOps platform, acting as go-betweens for external data sources and the **Interoperability Layer's NGSI-LD Context Broker**. Their main job is to **retrieve, transform and ingest** data into the **NGSI-LD Context Broker** and they are dependent on the external data source. They ensure this data is semantically consistent and interoperable, regardless of its original format.

##### Integration

External data often comes in various formats like JSON, XML, CSV, or through proprietary APIs. To achieve semantic interoperability within CyclOps, agents are responsible for converting this diverse data into the standardized **NGSI-LD format**. This conversion involves:

- **Mapping raw attributes to Smart Data Models.**
- **Adding semantic context** using @context definitions.
- **Structuring the data into NGSI-LD Entity representations** for efficient storage and querying.

Beyond data transformation, agents would be useful to facilitate the connection with various **data space connectors**, such as the FIWARE Data Space Connector (FDSC) or others that follow the **Data Space Protocol (DSP)**. These connectors handle secure and policy-compliant data exchange between different organizations. Agents can support functionalities like:

- **Pulling data** from endpoints exposed via data space connectors.
- **Initializing Authentication and Authorization** according to connector policies.
- **Handling contract agreements and metadata**, if the underlying data space infrastructure requires it.

##### Advantages

This approach guarantees that incoming data conforms to **common standards**, which significantly eases its **integration, analysis and reuse** across all components of the CyclOps platform. By facilitating secure and standardized data exchange, agents enhance the platform's ability to work seamlessly with diverse external data sources.

#### 2.2.3. Progress Report

##### Current Status

At the current stage of the project, the **NGSI-LD Context Broker** is fully operational, forming the backbone of the initial data ingestion workflow and temporary storage. The broker is already capable



of receiving and storing NGSI-LD compliant data, and its API is accessible for local querying and data manipulation. For IT-2 the access will be controlled using the CyclOps platform's authentication framework to ensure data security.

Data population is currently enabled through the **Data Space Retrieval Script**, as described in Section Decentralized identity, trust and access control - M2M. This script retrieves datasets from the dataspace, based on valid Verifiable Credentials, and populates the Context Broker with the received entities. The ingestion process adheres to the NGSI-LD specification, ensuring compatibility with downstream applications and services.

Although external data publication is not yet enabled, the current setup provides a solid foundation for integrating additional processing logic, internal data sources or future outbound interfaces. The system is ready to support end-to-end data flows from dataspace ingestion to local contextual data availability.

### Challenges and Mitigation Steps

**Challenge:** Data from external sources often comes in a wide variety of formats and structures, ranging from flat CSV files and proprietary APIs to complex XML or JSON-based schemas. This heterogeneity makes it difficult to directly consume and process data across components of the CyclOps platform.

**Mitigation:** To address this, all incoming data is mapped and transformed into the **NGSI-LD** standard format. NGSI-LD provides a flexible and semantically rich data model that supports linked data concepts, enabling consistent representation and interpretation across domains. Agents are deployed to automate the transformation process, using **Smart Data Models** as a common vocabulary to ensure semantic alignment.

### Next Steps

In the next phase of the project (IT-2), efforts will focus on enabling **external data publication** from the NGSI-LD Context Broker via the **FIWARE Data Space Connector (FDSC)**. This integration will allow the broker to serve as a trusted data provider within the dataspace ecosystem, enabling secure and controlled sharing of curated datasets with external consumers.

The FIWARE Data Space Connector will be configured to expose the **Data Space Protocol (DSP)** specification, supporting machine-readable contract negotiation through its API. This will allow external organizations to discover available datasets, initiate data access requests, and negotiate usage agreements in a fully automated and standardized manner.

To further streamline and enhance the data exchange process, a **marketplace component** is planned to be deployed and integrated—such as the **BAE Marketplace** provided by FICODES. This marketplace will offer a user-friendly interface for dataset discovery, browsing, and transaction management. It will simplify the experience for data consumers by enabling manual and semi-automated purchasing workflows, while ensuring alignment with the dataspace's governance and compliance frameworks.

Together, these future developments will extend the current architecture into a fully functional, federated data exchange environment—supporting both technical interoperability and business-level interactions across organizational boundaries.

## 2.3. Publication

### 2.3.1. Introduction

As part of its long-term strategy, the CyclOps project aims to ensure that its data products and services are not only technically robust and semantically interoperable, but also visible and accessible to a broader European ecosystem. For this, a connection to the DOME Marketplace (Distributed Open Marketplace for Europe) is desired. This integration offers a pathway to increased visibility, discoverability and potential business opportunities for data providers and consumers alike.

The DOME Marketplace is a strategic initiative of the European Commission designed to support the digital transformation of public and private sector organizations across Europe. Built upon Gaia-X principles and open standards, DOME will serve as the central marketplace for trusted cloud and edge services, including datasets, AI tools, and software components developed through EU-funded programs such as Horizon Europe and the Digital Europe Programme.

CyclOps integration with DOME aims to enable project outcomes, especially curated, NGSI-LD-compliant datasets, to be registered and published via a shared digital catalogue, making them accessible to other marketplace participants. This will be particularly valuable for domains such as smart cities, industry, tourism and energy, where semantic data interoperability is critical and demand for high-quality datasets is growing.

Furthermore, by leveraging the federated marketplace model of DOME, CyclOps ensures that its offerings can reach sector-specific platforms and vertical integration hubs, aligning with the European vision for trusted, sovereign, and interoperable data ecosystems. This connection positions CyclOps as a contributing and benefiting actor in the broader European data economy.

### 2.3.2. Implementation

For IT-2, the integration of CyclOps services with DOME-compatible marketplaces under task 5.3 will focus on providing a robust, standards-based approach to ensure seamless integration within the cyclops platform. The primary integration method will rely on standardized APIs, notably the TMForum APIs<sup>3</sup>, which are explicitly aligned with the Data Spaces Business Alliance (DSBA)<sup>4</sup> convergence model, also aligned with the components of activity 5.1 in the previous sections. This will be complemented using the FIWARE NGSI-LD standard and Context Broker implementation for efficient data exchange.

A critical component will be the implementation of decentralized identity solutions, adhering to the Self-Sovereign Identity (SSI) approach, utilizing W3C Decentralised Identifier (DID) and Verifiable Credential (VC) standards. It will be analysed if these components can be shared between the integration with data spaces (from Task 5.1) and the integration with DOME marketplaces. These are essential for establishing trust and access control, aligning with broader European initiatives such as EBSI/ESSIF, eIDAS, and the Gaia-X Trust Framework. The DOME infrastructure itself incorporates a Trust and Identity and Access Management (IAM) framework and a Decentralised Persistence Layer to support these federated interactions.

### 2.3.3. Progress Report

#### Current Status

The integration with the DOME-based EU marketplace has been postponed to IT-2 to align with the evolving maturity of the project's service candidates (data and services that shall be provided) and technical components. This decision ensures that the functionalities intended for publication and marketplace onboarding are sufficiently stable and well-defined before proceeding with integration activities.

During the current reporting period focus was laid on **collecting and analysing concepts, specifications and integration requirements** related to the DOME ecosystem. This includes understanding the federated architecture, onboarding workflows, identity and access mechanisms and technical components such as the DOME access node, data space connectors and policy enforcement points.

By deferring the technical integration to a later stage, the project creates the necessary space to **refine its offerings and capture precise requirements**, ultimately supporting a more robust and standards-aligned implementation in the next iteration.

---

<sup>3</sup> <https://github.com/FIWARE/tmforum-api>

<sup>4</sup> <https://data-spaces-business-alliance.eu/dsba-releases-technical-convergence-discussion-document/>

## Challenges and Mitigation Steps

**Challenge:** The broader ecosystem of European data spaces, including initiatives like DOME, Gaia-X, and others, is still under active development. As a result, technical standards, onboarding procedures and governance frameworks are subject to change, creating uncertainty for integration planning.

**Mitigation:** CyclOps addresses this by adopting a future-proof design philosophy, aligning closely with EU strategies and open standards. The architecture includes abstraction layers between data ingestion, transformation and publication, making it easier to adjust to future protocol changes or to support new data space connectors as they mature.

## Next Steps

As part of the broader objective to enable seamless participation in federated European data markets, the next steps will focus on preparing for integration with the **DOME** infrastructure. A key aspect of this integration will be to evaluate and ensure compliance with the specific **Verifiable Credential (VC) requirements** mandated for participants within DOME. This includes verifying the format, attributes, and trust framework expected by DOME for secure and standardized identity and contract handling.

Following this assessment, the project will proceed with the **deployment of the DOME Access Node**, which consists of several interconnected components:

- i) **Blockchain-based Connector<sup>5</sup>:** This component handles the **sharing of policies and purchase agreements** via blockchain infrastructure. It acts as a trusted ledger for transaction and rights traceability within the DOME ecosystem. Policies and Product Orders are persisted via the TM Forum API Service
- ii) **Blockchain Adapter (DLT Interface)<sup>6</sup>:** Serves as an **abstraction layer for the underlying Distributed Ledger Technology**, enabling compatibility with different blockchain platforms and simplifying integration efforts.
- iii) **TM Forum API Implementation with NGSI-LD Backend<sup>7</sup>:** This service provides a **standardized API interface for managing product specifications and orders**, with data persistence handled by an NGSI-LD Context Broker. It ensures interoperability with business support systems following the TM Forum standards.
- iv) **APISIX with Open Policy Agent (OPA):** Used as a **Policy Enforcement Point (PEP) and Policy Decision Point (PDP)**. APISIX will intercept and route API requests while enforcing access control decisions provided by OPA, based on the associated usage policies.
- v) **ODRL Policy Administration Point (ODRL-PAP)<sup>8</sup>:** This module acts as the **ODRL Policy Management interface**, where product-related ODRL policies (e.g., usage rights, constraints) are stored and managed. It also **translates ODRL policies into OPA-compatible Rego rules** to be enforced at runtime.

Together, these components will enable secure, standardized and policy-driven data product exchanges within the DOME marketplace. The integration will make the project's data offerings discoverable and consumable by external parties across Europe, while ensuring full alignment with emerging European dataspace standards and trust frameworks.

## 2.4. Decentralised identity, trust and access control

### 2.4.1. Introduction

The CyclOps platform implements a modern, decentralized identity management system based on the principles of Self-Sovereign Identity (SSI), in alignment with the Technical Convergence guidelines established by the Data Spaces Business Alliance (DSBA). This approach ensures compatibility with

<sup>5</sup> <https://github.com/in2workspace/in2-dome-desmos-service>

<sup>6</sup> [https://github.com/alastria/DOME-blockchain\\_connector-dlt\\_interface](https://github.com/alastria/DOME-blockchain_connector-dlt_interface)

<sup>7</sup> <https://github.com/FIWARE/tmforum-api>

<sup>8</sup> <https://github.com/wistefan/odrl-pap>

key European and international initiatives, including EBSI/ESSIF, IDSA, eIDAS, and GAIA-X, fostering interoperability across emerging data spaces.

The solution leverages a robust Trust Framework underpinned by W3C Verifiable Credentials (VCs)<sup>9</sup> and W3C Decentralized Identifiers (DIDs)<sup>10</sup>, OID4VP<sup>11</sup>, OID4VCI<sup>12</sup> standards, while adhering to the EBSI Conformance Guidelines and aligning with the EU Digital identity Wallet<sup>13</sup> as defined in the Architecture and Reference Framework (ARF)<sup>14</sup>. The solution enables secure, privacy-preserving, and verifiable identity interactions between participants, ensuring trust and compliance within the ecosystem.

The identity architecture is composed of modular services that are deployed across the domains of participating organizations within a dataspace, tailored to their specific roles within the data space. Following the GAIA-X Conceptual Model, CyclOps supports the following participant roles:

- **Consumers:** Entities that request and utilize data or services.
- **Providers:** Entities that offer data or services to others.
- **Operators:** Entities responsible for maintaining the trust infrastructure, including the Trust Anchor Framework, which governs the onboarding and verification of trusted participants and credential issuers<sup>15</sup>.

By supporting sovereign identity services across these roles, CyclOps ensures scalability, sovereignty, and alignment with the federated governance models envisioned for EU data spaces.

The CyclOps SSI identity management system supports both: (1) authenticated Person-to-Machine (P2M) access to the CyclOps platform for Data Scientist users within an organization to make sure they are authorized to access CyclOps services (2) authenticated Machine-to-Machine (M2M) access to a data space when the CyclOps platform needs to query a data space.

Currently, there are no data space connectors that support the approach outlined in the DSBA Technical Convergence guidelines with all the above-mentioned open standards, apart from the FIWARE Dataspace Connector (DSC) and hence the interoperability layer identity system is only able to integrate with this DSC. To be able to integrate with other data spaces at the identity level it would be needed for them to either support open-standard SSI or to add the FIWARE VCVerifier as an additional identity mechanism which they could allow access to their data space. In this way the FIWARE VCVerifier identity components are also included within the CyclOps interoperability layer.

At its core, CyclOps delivers a state-of-the-art identity solution that supports:

- Verifiable Credentials Issuing (with Keycloak integrated with SSI)
- Verifiable Credentials Verification
- Verifiable Credentials Wallet

The actual deployment of the SSI components of the interoperability layer depends upon the deployment of CyclOps in a consumer or provider participant in a dataspace. The figure below demonstrates the needed components depending on the deployment, noting that the Credential Config and Trusted issuer list services can be considered to be part of the VCVerifier capabilities.

---

<sup>9</sup> <https://www.w3.org/TR/vc-data-model/>

<sup>10</sup> <https://www.w3.org/TR/did-1.0/>

<sup>11</sup> [https://openid.net/specs/openid-4-verifiable-presentations-1\\_0-14.html](https://openid.net/specs/openid-4-verifiable-presentations-1_0-14.html)

<sup>12</sup> [https://openid.net/specs/openid-4-verifiable-credential-issuance-1\\_0-11.html](https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0-11.html)

<sup>13</sup> <https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/releases/tag/v1.4.0>

<sup>14</sup> <https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/releases/tag/v1.4.0>

<sup>15</sup> Although Operators are not dependent on CyclOps deployments, since CyclOps needs to get access to data in a dataspace or provide data to it, the organization deploying CyclOps is considered a participant

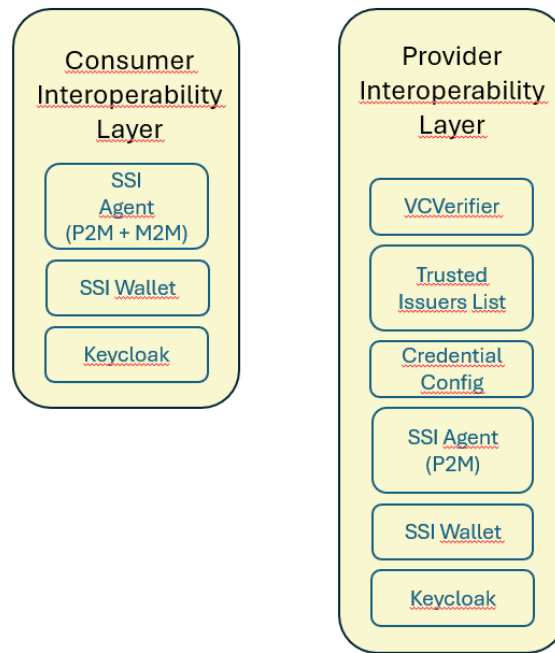


Figure 2. Interoperability layer SSI Components

### Issue P2M SSI Credential

The figure below shows the high-level interaction of the SSI Wallet, SSI Agent and Keycloak for issuing a Verifiable Credential to an employee of Organization X.

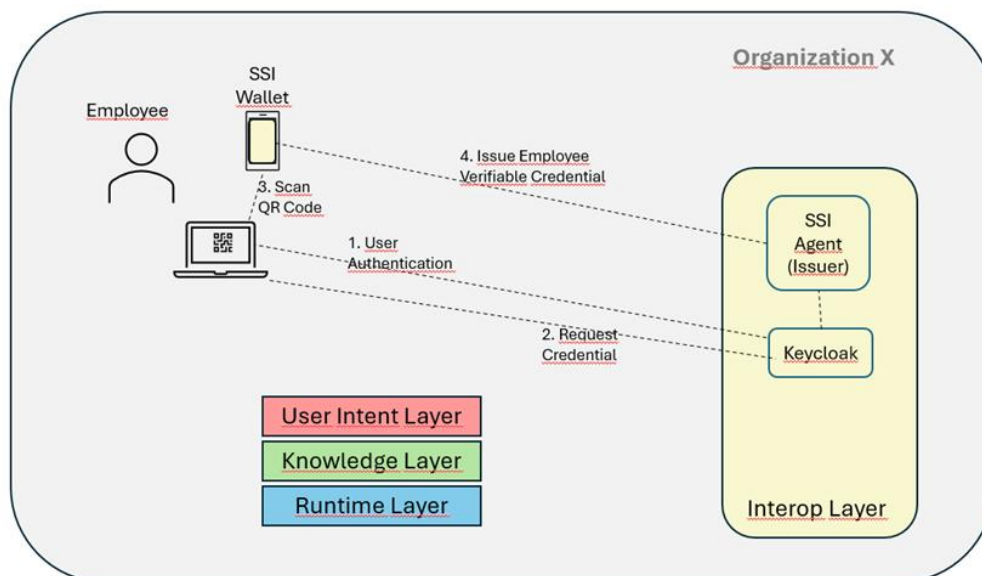


Figure 3. Issue P2M SSI Credential

### P2M Verify SSI Credential

The figure below shows the high-level interaction of the SSI Wallet and SSI Agent for employees of Organization X. The employees present their Verifiable Credentials from their SSI Wallet to the SSI Agent for verification and upon successful verification receive a self-contained Access Token which they can use to access the CyclOps platform and also potentially to marketplaces in a dataspace.



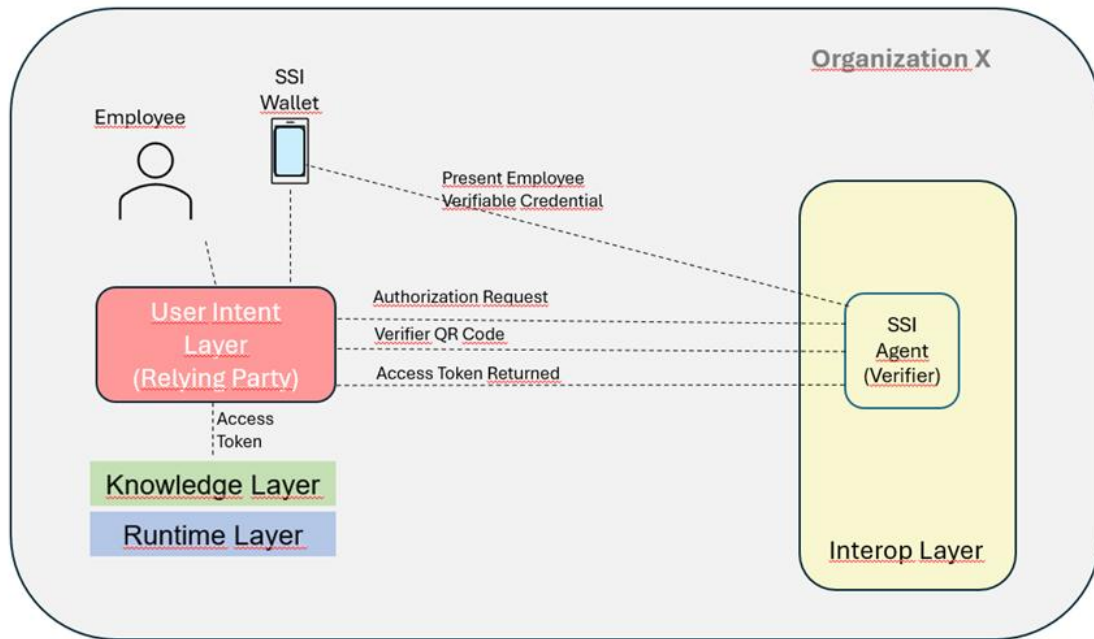


Figure 4. P2M SSI Verification

### M2M Issue & Verify Credential scenarios

The figure below shows a demonstration SSI Script tool to demonstrate one of the candidate approaches. The M2M flow is used for presenting of its credentials to a dataspace where the data provider service is hosted by Organization Y.

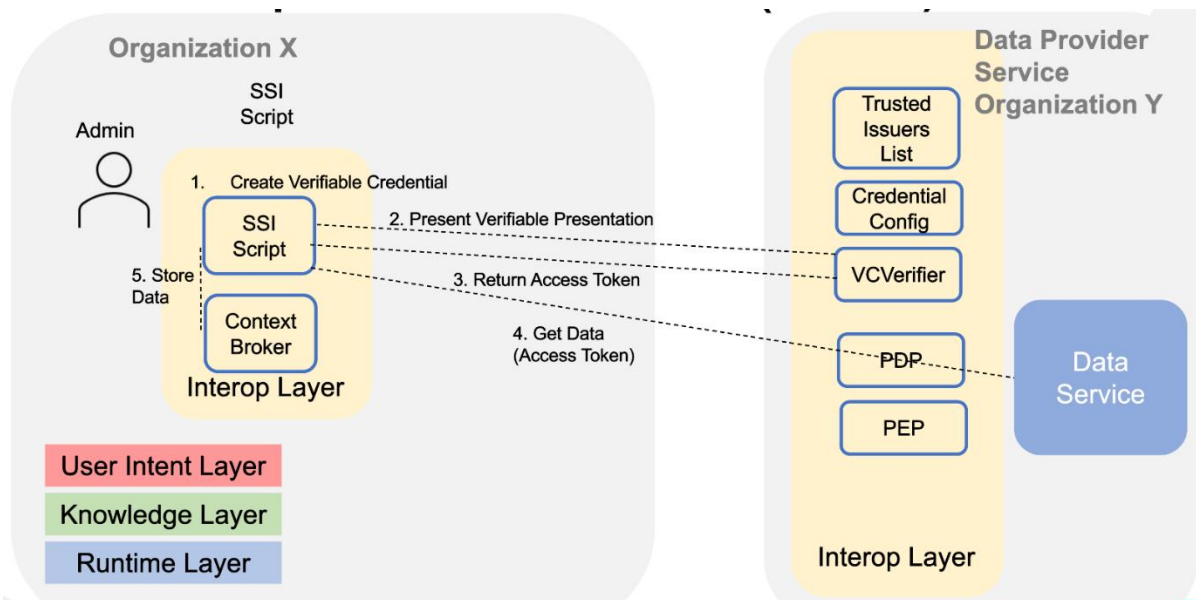


Figure 5. M2M SSI issuer &amp; Verifier

## 2.4.2. Implementation

### 2.4.2.1. P2M

#### Issuer

**SSI Agent** provides the issuing of Verifiable Credentials to SSI Wallets. It is developed further in CyclOps so to support the integration with an SSI adapted Keycloak component. The issuer follows the EBSI Guidelines.

**Keycloak**<sup>16</sup> provides Identity and Access Management and has been modified for the CyclOps project so that an organization's end user's (e.g. Data Scientist employees) can register in Keycloak and request to be issued with their identity attributes registered in keycloak, through integration with the SSI Agent.

**SSI Wallet** provides the end users management of their Verifiable Credentials in a mobile application and is used to receive and present Verifiable Credentials. The base asset is further improved in CyclOps to support the standard verification flow for presenting Verifiable Presentations for verification flow as well as the existing support for the issuer flow.

The issuing of an employee credential is captured in the sequence diagram shown in Figure 6, detailing the interwork between the Keycloak Issuer, SSI Agent, and end user SSI Wallet.

The message flow is described below:

- 1) User opens the organization's Keycloak frontend web page and start to register to be issued with an employee credential.
- 2) This initiates the following request for an Employee Verifiable Credential to the SSI Agent, as follows: Note: The client\_id is that of the issuer organization and may be a DID or a URI.
- 3) This returns a credential offer uri to the Issuer frontend displays to the user to scan from their wallet.
- 4) The keycloak issuer frontend generates a QR Code with the credential offer uri.
- 5) The user opens mobile wallet to scan the QR Code.
- 6) The user scans the QR Code.
- 7) The offer is presented to the wallet.
- 8) The user accepts the offer.
- 9) The wallet will query the credential offer uri it received in step 3 to fetch the offer from the agent.
- 10) The employee credential offer is received at the wallet:
- 11) Get openid-credential-issuer metadata
- 12) Return issuer metadata
- 13) Get issuer's authorization server metadata
- 14) Return authorization server metadata
- 15) Authorization request
- 16) 302 Redirect to wallet with a ID Token Request URI
- 17) Redirect 302
- 18) Lookup the issuer's signature
- 19) Return the jwks
- 20) ID Token Response (Direct Post) from Wallet to Authorization Server of the Issuer
- 21) Authorization Response reaches the SSI Wallet on the mobile
- 22) Token Request
- 23) Token Response
- 24) Credential Request
- 25) Query Keycloak for claims against the type of credential being requested
- 26) Return the user claims for the subjectID
- 27) Receive an employee credential.
- 28) Credential is stored in the wallet

---

<sup>16</sup> <https://www.keycloak.org/>

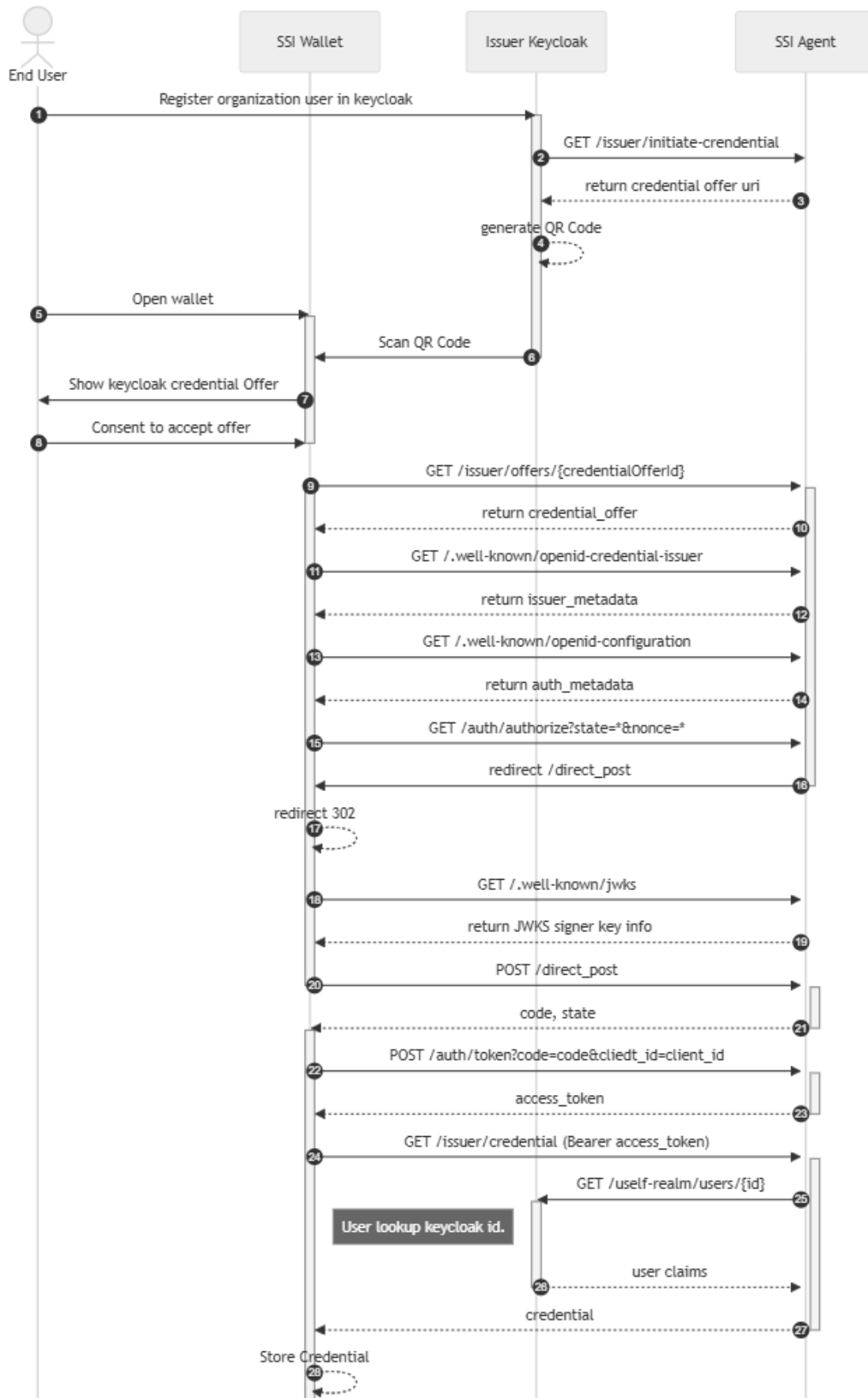


Figure 6. Keycloak Issuer flow



## Verification

The following interwork diagram and description conforms to the OID4VP<sup>17</sup> draft 15 for a relying party acting as a verifier in accordance with the reference specification of the European Wallet Ecosystem defined in the EU DIW ARF.

In this verification flow, an end user authenticates to the CyclOps Platform's UI Layer with their Employee Credential (presented from mobile SSI Wallet). The credential has been issued from their organization and as such when presented for verification, they are seen to be representing their organization. Similarly, the same user credential may be used to access a marketplace that offers data services in the data space.

This scenario shows an employee accessing a consumer service and being authenticated with his/her mobile SSI Wallet by the DSC, so to obtain an Access Token to be used by the consumer service to retrieve data over the DSC. Cross device verification code flow:

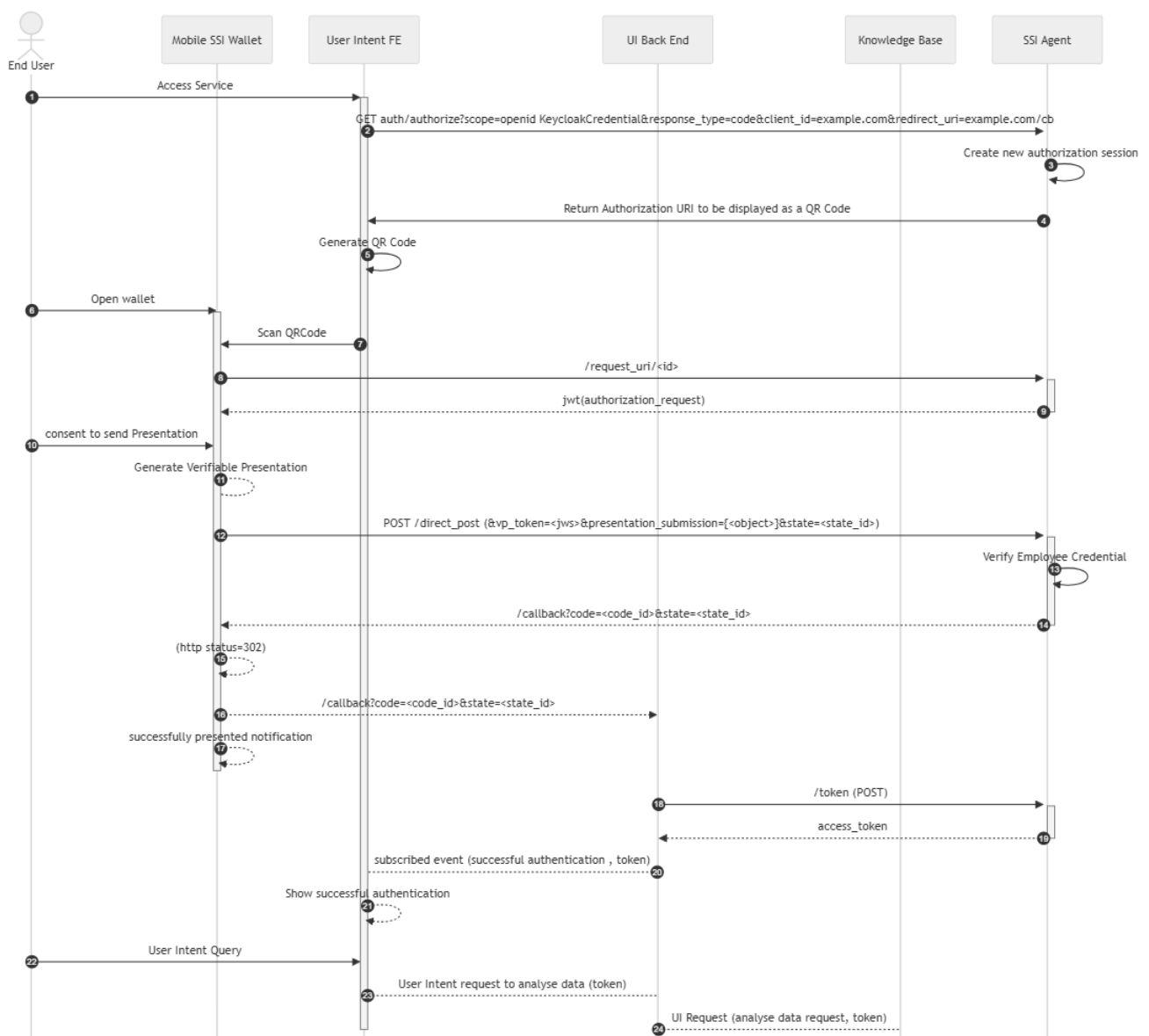


Figure 7. Verification flow

<sup>17</sup> [https://openid.net/specs/openid-4-verifiable-presentations-1\\_0-15.html](https://openid.net/specs/openid-4-verifiable-presentations-1_0-15.html)

- 1) The user opens the User Intent web page.
- 2) User clicks on SSI login and an authorization call is made to the CyclOps platform's identity management with the needed parameters including the clientID of the CyclOps service being requested.
- 3) The SSI Agent initiates a new authorization session
- 4) The SSI Agent returns 200 OK
- 5) QR Code is generated and presented on user interface
- 6) User opens their mobile SSI Wallet
- 7) User scans the QR Code on their mobile SSI Wallet
- 8) Fetch the authorization object in the Request URI
- 9) Authorization Request received in the wallet
- 10) User confirms the Verifiable Credential to be authenticated with
- 11) Mobile SSI Wallet generates the Verifiable Presentation
- 12) The Authorization Request redirect is sent to the previously obtained /direct\_post URI and includes the Verifiable Presentation token with the Verifiable credentials that meet the input descriptor constraints. The presentation submission contains a descriptor map indicating which verifiable credentials fulfill the respective input descriptor constraints. These parameters, both vp\_token and presentation\_submission, are sent within HTTP POST to the direct post endpoint.
- 13) The SSI Agent verifies the Employee VC is correctly presented checking the signature proof against the wallet, that the issuer signature proof is correct and that the issuer is trusted and the credential is not revoked.
- 14) A successful authentication will result in a redirect back to the callback address of the User Intent endpoint. Alternatively, in case of failure scenario an error code is returned. Standard HTTP response codes are supported.
- 15) The UA - SSI Wallet handles the 302 redirect
- 16) In response to the HTTP 302 redirect response by the Client above, the wallet send the callback request to the callback address of the UI backend.
- 17) Mobile SSI Wallet gives successfully presented credential notification
- 18) The UI Backend now exchanges the code to get the access token. Note: The client\_id is that of the organization and the same as was given in the authorization request.
- 19) A successful response to the above token request results in the access token being returned.
- 20) The UI Backend informs UI Frontend of the token it must use for the session.
- 21) The user is informed of successful authentication on the UI Frontend.
- 22) The user inputs a request to the UI Frontend.
- 23) The UI Frontend uses the token for the session when making requests to the backend. The UI Backend will check the token and makes sure the user has the correct role, and the token is correctly signed.
- 24) The UI backend uses the token for the session when making requests to other CyclOps Services.

#### 2.4.2.2. M2M

The Data Space Retrieval Tool is a scripted component that enables machine-to-machine (M2M) data ingestion through an OID4VP (OpenID for Verifiable Presentations) flow within a OID4VP/FIWARE-based dataspace ecosystem. The component operates as part of the Interoperability Layer, facilitating secure and standards-compliant data access and integration without immediate user interaction (required for updated datasets).

Note: For IT1 this is a demonstration tool to show one of the approaches on how M2M flow is able to be supported. However, research is currently ongoing to analyse open-standard approaches for IT2 implementation considering Organization Wallets issued with legal identities through the GAIA-X or EBSI trust model.

The tool initiates an OID4VP authentication process using a pre-issued Verifiable Credential (VC), which was provisioned by a Keycloak instance and includes all necessary cryptographic key material.

The organization has been pre-registered in the dataspace, and the dataset in question has already been acquired through a prior purchase agreement.

Upon successful completion of the OID4VP flow, the script uses the credential for an authorized access to a remote NGSI-LD interface exposed by the data provider. It retrieves the relevant dataset and ingests it into a NGSI-LD Context Broker instance in the Interoperability Layer for further processing and consumption within the CyclOps platform.

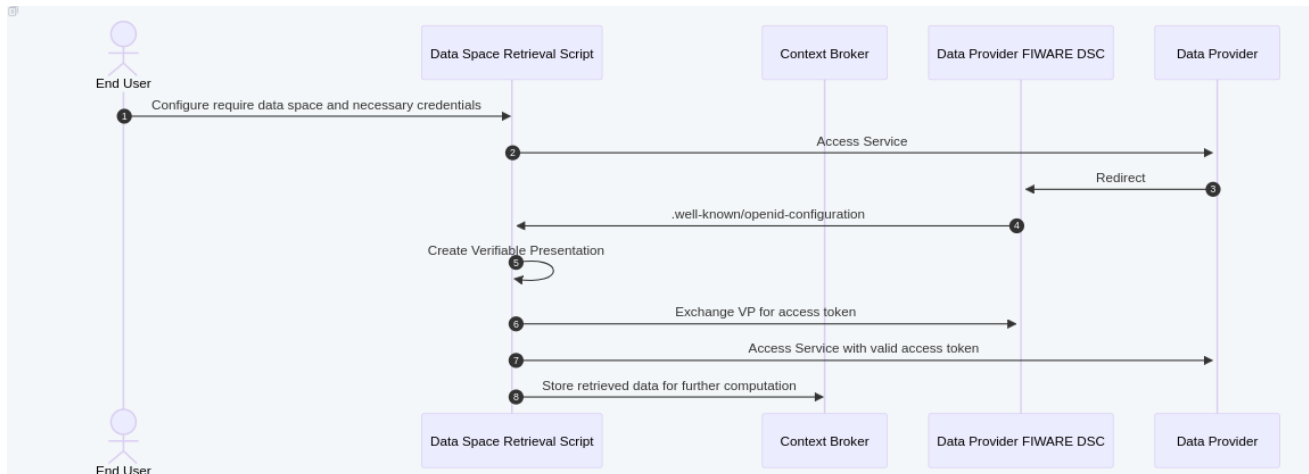


Figure 8. OID4VP M2M Flow Sequence Diagram: The access of a dataset in the dataspace including its persistence. The Data Provider FIWARE DSC contains the VCVerifier component as marked in D2.2.

### 2.4.3. Progress Report

#### Current Status

The current status of Task 5.2 is as follows:

- The SSI Agent docker is deployed with Keycloak service to provide Issuer and Verification functions for organization employees authenticating to the CyclOps Platform.
- The SSI Wallet APK is uploaded to the CyclOps Repo for supporting P2M authentication to the CyclOps Platform for organization employees
- The M2M Data Space Retrieval Tool is deployed in a dockerized form as part of the Interoperability Layer. Currently the script is meant to be run as a job for every dataset that is required

#### Challenges and Mitigation Steps

**Challenge 1:** In considering the data spaces that could be integrated with to meet the identity requirements of the project with open standards, it was found that only FIWARE DSC complies with the DSBA Technical Convergence, whereas EDC has implemented its own non-standard Decentralized Claims Protocol (DCP)<sup>18</sup> for M2M authentication loosely based on SSI specifications. FIWARE have written a report on this<sup>19</sup>.

**Mitigation 1:** The CyclOps Interoperability layer cannot support proprietary implementations and therefore the only mitigating solution at this time is to recommend that the EDC data space connector also includes the FIWARE DSC identity components to allow an SSI standard based M2M authentication.

**Challenge 2:** The original requirement from WP2 was to use P2M (Person-to-Machine) authentication to access the Data Space Connector (DSC).

<sup>18</sup> <https://github.com/eclipse-dataspace-dcp>

<sup>19</sup> [Identity and Authentication Management based on Verifiable Credentials and Decentralized Identities - Google Docs](#)

**Mitigation 2:** Later, after further analysis, it was decided that authentication would also be needed for the CyclOps platform. To address this in an integrated approach, a P2M token exchange method was considered to enable P2M access to the data space. However, it was also additionally considered that access might be performed by AI in batch mode, as well as with real-time human assistance, so it was decided to limit P2M to the CyclOps platform and marketplace only. For accessing the data space, M2M (Machine-to-Machine) authentication was decided to be the best fit. Both P2M and M2M capabilities will rely on the same trusted credential issuer.

### Next Steps

- i) Extend the Data Space retrieval script to be configured through an API. When the Data Scientist buys access to new data sources and dataspace, this information will be used to update the functionality.
- ii) Extend the Data Space retrieval script to trigger Ingestion Jobs at the Runtime Layer.
- iii) Extend the Data Space retrieval script to respect presentation definitions provided by data spaces to adjust the credentials being sent.
- iv) Integrate other data sources (as used in UC1 and UC2) to provide a uniform interface to the CyclOps platform.
- v) Deploy the FIWARE Data Space Connector components to enable data publishing while providing a robust access control mechanism using ODRL and trusted standard components
- vi) Analyse, select and implement trust model for the trusted onboarding of an organization to support the same trusted legal issuing of Verifiable Credentials for both P2M & M2M implementations considering EU Digital Identity Wallet, EBSI and GAIA-X for the SSI Agent
- vii) Analyse support of P2M Verifiable Presentations to the DOME Marketplace from the SSI Wallet based on the selected trust model, discussed above.
- viii) Support standards-based M2M Organization Credential Verifiable Presentations on SSI Agent to the VCVerifier considering EU Digital Identity Wallet, EBSI & GAIA-X.
- ix) Support Organization wallet capability for the M2M flow

## 3. List of Components Included in CyclOps Release 1

The following table presents the main components included in the CyclOps IT-1 release. These components are described in detail in deliverables D3.1, D4.1 and Section 2 of this document (i.e., regarding the interoperability layer), where their architecture, functionalities, and internal workflows are explained. Components described are also available in the open CyclOps GitLab repository (<https://gitlab.com/cyclops>) along with their respective README files, providing guidance on installation, usage, and integration<sup>20</sup>.

For each component listed below, in an alphabetical order, an extensive description including functionality, and the list of technical dependencies required to run it is provided in the [Annex](#) of this deliverable.

CyclOps Component Name (Resp. Partner)	Short Description	WP
AI Marketplace (CeADAR)	Allowing the user to select the most suitable algorithm from the decentralized algorithms repository (DAR)	WP3
AI Model Protection (ATOS)	This module protects AI models by continuously monitoring inference data for drift, detecting deviations from the training data in schema, feature distribution, or statistical patterns to ensure prediction reliability and accuracy.	WP4

<sup>20</sup> Components provided externally have also a repository in the projects GitLab, containing only the necessary README files. Note that components that are proprietary (i.e., and not licensed as open-source), based on specific partner recommendations, are not accessible through the open CyclOps repository.

CyclOps Component Name (Resp. Partner)	Short Description	WP
AI Models Repository (S5)	Object storage solution utilized for storing, managing and versioning trained AI models	WP3
Concept Extraction and Categorization Engine (EXAI) [Externally deployed]	The Concept Extraction and Categorization Engine is a domain-agnostic tool that extracts structured information from documents based on ontologies, facilitating data interoperability and traceability.	WP4
CyclOps Lab (CERTH)	Modular containerized lab for developing, executing, and monitoring machine learning workflows	WP3
CyclOps Long-Term Storage (CERTH)	The Long-Term Storage (LTS) component is a centralized storage solution.	WP3
Data Access (FIWARE)	Tool for retrieving datasets from data spaces and storing them in the interoperability layer's context broker	WP5
Data Augmentation (ATOS)	Generate new synthetic data based on a reference dataset.	WP4
Data Discovery (UPC)	Discovering and ranking potential joinable attributes across datasets by analysing data similarity patterns	WP3
Data Exchange Agent (FIWARE)	Agents for transforming data from different sources into a common format	WP5
Data Exchange Broker (FIWARE) [Externally deployed]	NGSI-LD Context broker for storing datasets before ingesting them into the Runtime Layer and for publishing them to external partners	WP5
Data Ingestion Functions (S5)	Ingesting structured data from diverse sources through multiple modalities (File Upload, Data Provider's available API, Component's API)	WP3
Data Preprocessing and Cleaning Functions (FDI)	Normalizing and transforming raw input data for ML training	WP3
Data Quality Rules Discovery (UPC)	Discovering data quality rules to ensure coherence and avoid redundancy on a given dataset	WP3
Distributed Execution Engine (BSC)	Application execution solution for distributed computing environments	WP3
Exploratory Analysis Engine (TUBS)	Transforming user intent to SPARQL query sent to the IKB for data extraction	WP4
Feature Engineering (UPC)	Helps CyclOps users select the most relevant features for their machine learning tasks by providing statistical analysis, correlation insights, and feature importance metrics to improve model accuracy and efficiency.	WP3
IKB Exploitation tool- Natural Language Understanding (NTTDES)	Enriching texts from the IKB with extra information provided by NLP algorithms	WP4
IKB Metadata Manager (ONTO)	Stores and manages metadata (mainly annotations), serves these annotations via GET requests, and exposes them through a PostgreSQL database mapped as Virtual Knowledge by the Ontopic Server.	WP4
Info Retrieval (EXAI) [Externally deployed]	Processes and interprets natural language conversations to extract structured intents for automated data analytics pipeline generation within the CyclOps platform.	WP4

CyclOps Component Name (Resp. Partner)	Short Description	WP
Intent Compiler (TUBS)	Parsing and storing the users' high-level inputs, verifying and matching the intents with adequate policies and compliance checking.	WP4
Large-Scale Data Management (BSC)	Distributed data management solution for the distributed execution engine	WP3
Model Deployment (CERTH)	Deployment of models in production	WP3
MPBoot (UNIBZ)	Command-line tool that allows the user to bootstrap an ontology and mapping file starting from an initial relational database instance.	WP4
Multimodal Explainability Module (CeADAR)	Set of explainability techniques for AI models offering valuable insights into model behaviour	WP4
NextiaMG (UPC)	Automated bottom-up schema bootstrapper and R2RML mapping generator from structured and semi-structured data (without a reference ontology)	WP4
NLP Chat (DC)	Managing Interaction with the user to specify their intents	WP4
Ontopic Suite (ONTO)	Environment for building Knowledge Graphs from relational data. The intuitive no code environment enables easy onboarding and collaboration.	WP4
Optimization Functions Repository (FDI)	Collection of optimization algorithms for ML model tuning	WP3
Orchestrator (UPC)	Generating pipeline for data analytical function	WP4
Publication (FIWARE)	Components covering publication of dataset via different means (e.g. DOME).	WP5
Semantic Interoperability (FIWARE)	Provides a validation of a payload if it is compliant with any of the existing Smart Data Models.	WP5
Semantic-based Data Curation (NTTDES)	Ensuring the quality of the data in semantic format	WP3
Semantic-based Data Discovery (NTTDES)	Finding similarities amongst datasets in semantic format	WP3
SSI Agent (ATOS)	A core identity service component responsible for supporting issuing of W3C-VCs to members of both consumer and provider organizations.	WP5
SSI Wallet (ATOS)	A user-centric identity android mobile application that enables individuals to securely store, manage, and present W3C Verifiable Credentials (VCs).	WP5
User-assisted Intent Interface (TUBS)	Managing user login and intent expression	WP4
VC Verifier (FIWARE) [Externally deployed]	Provides the necessary endpoints to offer SIOP-2/OIDC4VP compliant authentication flows, then used by downstream components.	WP5

Table 1.Components Included in CyclOps Release 1.



## 4. CyclOps Testing Approaches

### 4.1. Testing Phases

CyclOps follows an iterative, data-driven, continuous, and user-centric methodology based on an incremental approach and continuous development strategy (CI/CD), as showcased in Fig. 9. This strategy enables CyclOps to automate the software development process (i.e., from coding through deployment), enabling partners to release new features and fixes on their components faster, enhancing the platform's responsiveness to use cases and user needs.

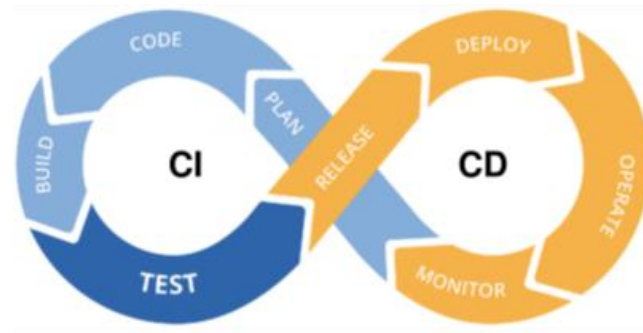


Figure 9. CyclOps CI/CD Workflow

Under the CyclOps CI/CD strategy, continuous testing is performed using various types of test phases (Fig. 10) including:

- **Unit Testing (UT)**, which evaluates individual units of code working as expected
- **Integration Testing (IT)**, which evaluates how different components work together
- **Acceptance testing (AT)**, performed by the use cases after the previous system tests have been completed.

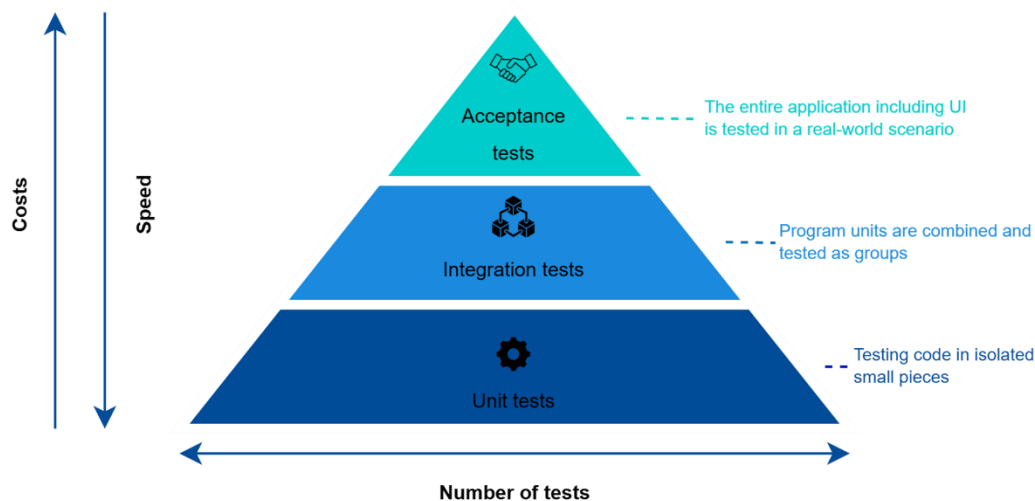


Figure 10. Testing Phases: Unit, Integration, and Acceptance

In detail, UT is a software testing approach that focuses on evaluating individual components or units of a program. The main goal is to confirm that each unit of code behaves as expected. This practice is a fundamental part of the software development process, as it helps identify and resolve bugs early on, making the code more reliable and robust. When changes are made to the software, previously written unit tests can reassure developers that existing functionality still works as intended.

IT, on the other hand, is a software testing approach where individual modules are combined and tested as a group. Its primary purpose is to uncover flaws in the interactions between these integrated units. This type of testing emphasizes the interfaces and interactions between components, aiming to detect issues such as data inconsistencies, communication problems, or function mismatches. Integration testing is typically conducted after unit testing and before system testing in the overall software testing process.

AT is the concluding stage of the software testing process, wherein actual users from the predefined use cases evaluate the platform, to verify its compliance with their requirements and its performance in real-world settings. AT is conducted after the software has passed all other testing phases (i.e., unit and integration testing), to validate that the platform is ready for deployment, aligning with the business/technical requirements. During AT, end users perform test cases to identify any flaws or bugs that may have been overlooked previously.

Understanding the differences between UT and IT is crucial for implementing the appropriate testing strategy at the right time. UT and IT tests serve distinct purposes and are performed at different stages of the development lifecycle. Table 1 compares the differences between these tests:

Criteria	Unit Testing	Integration Testing
<b>Granularity</b>	Fine-grained	Coarse-grained
<b>Focus</b>	Focuses on individual units or specific components.	Evaluates the interaction of integrated modules.
<b>Complexity</b>	Less complex	More complex
<b>Testing Technique</b>	White-box testing	Black-box testing
<b>Dependency</b>	It tests parts of the project without waiting for others to be completed.	It tests only after the completion of all parts.
<b>Running Speed</b>	Fast execution as compared to integration testing.	Its speed is slow because of the integration of modules.
<b>Testing Workflow</b>	Developers typically perform unit tests before integration tests.	Usually conducted during the software integration phase.

*Table 2. Unit Testing vs. Integration Testing.*

## 4.2. Integration Testing Approaches

ITs play a crucial role in software development, as they identify issues that UT may not detect, particularly regarding databases, connections to other modules or systems, and performance validations. With the increasing prevalence of microservices, the significance of IT has grown, given that they ensure the smooth interaction of various services, particularly when these services are managed by different teams. In essence, while UT is essential for verifying fundamental logic, IT is required, as they demonstrate how different components of an application interact and perform collectively. ITs necessitate a broader range of scenarios for comprehensive coverage, which may overlook certain corner cases that UT can effectively address. The main IT approaches include:

**Big-Bang Approach** (non-incremental, see Figure 11): The Big-Bang approach entails the simultaneous integration of all modules and testing the entire system. This method provides thorough testing of the whole system, quicker integration, and immediate detection of interaction issues. It simplifies management by removing the need for stubs and drivers. However, debugging can be difficult due to the concurrent integration of numerous components, making it complex to identify the



source of defects. This approach requires that all modules are prepared for integration at the same time.

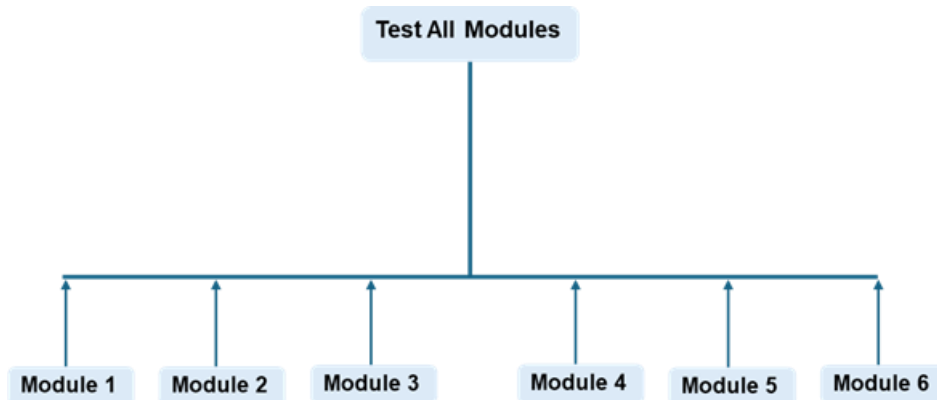


Figure 11. Big-Bang Approach

**Top-Down Approach** (incremental, see Figure 12): The top-down approach initiates integration from the top-level modules and progresses downwards. Stubs are utilized to mimic lower-level modules during initial testing. While this strategy allows for early validation of high-level functionality and can uncover significant design flaws, it postpones the testing of lower-level modules and requires the development of stubs, which can be time-intensive. As a result, integration problems in the lower-level modules may only become apparent later in the process.

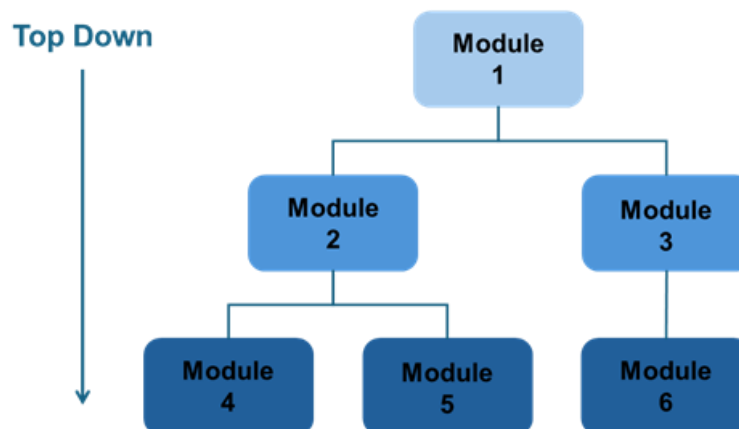


Figure 12. Top-Down Approach

**Bottom-Up Approach** (incremental, see Figure 13): In contrast, the bottom-up approach begins with the lowest-level modules and moves upward, employing drivers to simulate higher-level modules. This method facilitates early validation of lower level functions and makes it easier to discover and rectify defects in those modules. However, testing high-level functionality is postponed, and creating drivers can also be time-consuming, potentially leading to delays in identifying integration problems in the higher-level modules.

The CyclOps platform **utilizes the Big-Bang technique** to provide rapid and reliable integration. This approach concurrently integrates all components and subsequently evaluates them as a unified system. This methodology is preferred over the bottom-up or top-down methodologies because of its unique advantages:

- **Holistic Evaluation:** The CyclOps project entails the integration of various intricate components, which must function cohesively. The Big-Bang approach guarantees the comprehensive validation of the complete platform, verifying that all components interact appropriately from the beginning. Furthermore, this approach is more effective when discrete components are comparatively advanced and thoroughly validated in isolation. The

components of the CyclOps project, have undergone extensive UT, rendering them appropriate for this approach.

- **Rapid Integration:** Considering the project's magnitude and the necessity for fast development cycles, the methodology facilitates more rapid integration than phased approaches such as top-down and bottom-up. This is essential for adhering to stringent project deadlines and executing incremental releases effectively.
- **Streamlined Management and Timely Feedback:** The approach simplifies the integration process by consolidating it into a one, complete phase. This reduces the complicated task of managing various integration phases and guarantees an efficient workflow. Additionally, by concurrently integrating all components, offers rapid feedback regarding the overall functionality of the system, facilitating the rapid discovery and resolution of integration issues that may impact several components.

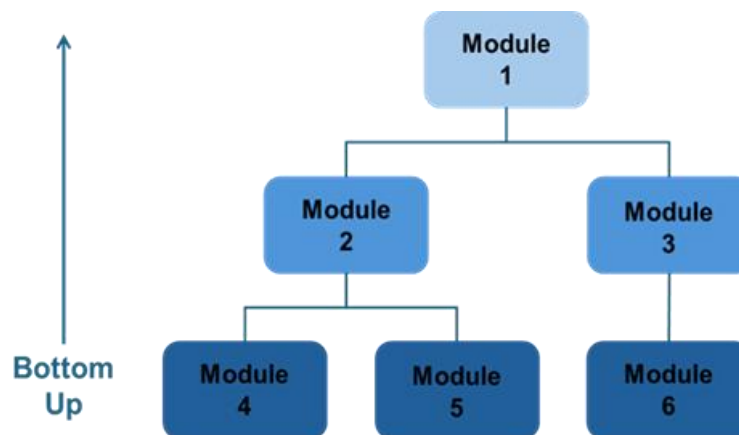


Figure 13. Bottom-Up Approach

## 4.3. CyclOps Testing Strategy

### 4.3.1. Testing Requirements

The CyclOps platform implements a comprehensive testing strategy that encompasses multiple testing methodologies and validation approaches to ensure robust, scalable, and reliable platform operation. This testing framework is designed to validate functionality across all components and architectural layers while maintaining high performance and reliability standards.

### 4.3.2. Unit Testing Framework

Component level validation forms the foundation of the testing strategy, with individual module testing implemented across the User Intent, Runtime, Knowledge, and Interoperability layers. The platform components will maintain a good test coverage, adhering to best practices thresholds (e.g., 80 % test coverage for critical components), with test-driven development practices encouraging unit test creation prior to implementation for core functionalities. In detail, the UT framework is designed to catch issues at the earliest possible stage, reducing the cost and complexity of bug fixes. Comprehensive unit tests protect existing functionality, enabling stable updates and supporting modular, independent development across partners.

To ensure robust and reliable behaviour, CyclOps components are individually validated through UTs. For example, the Preprocessing and Cleaning component (provided by FDI), includes a suite of UTs designed to verify core data transformation functionalities such as handling missing values, encoding categorical variables, scaling numerical features, and generating metadata. The figure below showcases a successful test run using pytest, confirming that all preprocessing unit tests passed, including:

- `test_drop_missing_rows`: validates correct removal of rows containing missing values. Ensures resulting DataFrame has no NaNs and the row count matches expectations.

- `test_is_one_hot_encoded_true`: verifies that properly one-hot encoded DataFrames (binary columns only) are accurately detected.
- `test_is_one_hot_encoded_false`: Ensures non-binary or partially encoded DataFrames are not misclassified as one-hot encoded.
- `test_apply_one_hot_encoding`: Tests transformation of categorical variables into one-hot vectors. Confirms new column naming, proper count of generated features, and preservation of numeric columns.
- `test_auto_scaling_standardization`: Applies standardization (Z-score scaling) on normally distributed data. Confirms correct method selection and expected output distribution (mean  $\approx$  0).
- `test_auto_scaling_normalization`: Applies Min-Max normalization on uniformly distributed data. Verifies output values fall within the range [0, 1].
- `test_generate_metadata`: Confirms correct generation of dataset-level metadata. Validates fields such as dataset name, dimensions, and per-feature summaries.

This test suite exemplifies how CyclOps enforces correctness and reproducibility at the component level, complementing broader integration and system-level validations. The results of the unit tests of CyclOps components in IT-1 are presented in section [Unit Test Matrix](#).

```
===== test session starts =====
platform linux -- Python 3.10.6, pytest-7.4.0
collected 7 items

test_preprocessing.py::test_drop_missing_rows PASSED
Dropped 2 row(s) due to missing values.
test_preprocessing.py::test_is_one_hot_encoded_true PASSED
test_preprocessing.py::test_is_one_hot_encoded_false PASSED
test_preprocessing.py::test_apply_one_hot_encoding PASSED
test_preprocessing.py::test_auto_scaling_standardization PASSED
Data appears to be normally distributed. Using Standardization (Z-score).
test_preprocessing.py::test_auto_scaling_normalization PASSED
Data does not appear normally distributed. Using Normalization (Min-Max).
test_preprocessing.py::test_generate_metadata PASSED

===== 7 passed in 0.94s =====
```

Figure 14. Example UT output of the Preprocessing and Cleaning component

### 4.3.3. Performance Benchmarking

Performance validation will be also conducted through comprehensive load testing that evaluates data ingestion and processing capacity on the CyclOps Platform Integration hub in IT-2. Scalability assessment should involve testing distributed executions performance across the infrastructure to ensure the platform can handle increasing workloads effectively. Response time validation focuses on API endpoint performance testing to guarantee real time user interactions meet acceptable performance thresholds. Additionally, resource utilization is another important aspect, which enables monitoring and provides detailed insights into CPU, memory, and storage performance profiling, enabling optimization of system resources and identification of potential bottlenecks. This comprehensive performance testing approach will ensure that the CyclOps platform can scale effectively to meet varying demand levels while maintaining consistent user experience and system reliability. Results concerning performance benchmarking are expected to be delivered in IT-2, when the full integration of all components will allow for a comprehensive evaluation of the platform's efficiency and scalability.

### 4.3.4. Docker and Deployment Testing

Container orchestration testing validates Docker Compose multi-service deployment scenarios to ensure seamless integration between different platform components. Cross-platform compatibility testing verifies deployment consistency across Linux version and distributions, ensuring reliable

operation regardless of the underlying infrastructure variations. Service integration testing focuses on inter-container communication and dependency management validation to guarantee proper component interaction.

#### 4.3.5. API Documentation and Validation

API testing involves automated endpoint validation for inter components communication, supported by comprehensive test suites that verify proper functionality and error handling. Query testing specifically targets the Knowledge layer, validating query performance and accuracy to ensure reliable data retrieval and manipulation. Interoperability testing focuses on FIWARE dataspace integration and federated data exchange validation, ensuring seamless operation across different data sources and platforms.

Documentation standards are maintained through documentation generation and consistency checks. This approach ensures that API documentation remains current and accurate, facilitating integration efforts and reducing the learning curve for developers working with platform components.

#### 4.3.6. Integration Testing Requirements

End-to-end pipeline validation encompasses complete user intent to execution workflow testing across all platform layers, ensuring that complex user scenarios function correctly from initial request through final result delivery. Cross layer communication testing validates data flow between architectural layers with comprehensive error handling to ensure robust operation even when individual components experience issues.

Dataspace interoperability testing covers federated data scenarios and semantic interoperability validation, ensuring that the platform can effectively work with diverse data sources and formats. Regression testing through automated testing suites prevents feature degradation during updates, maintaining platform stability as new features are added and existing components are modified.

#### 4.3.7. Best Practices Implementation

Continuous testing should be integrated into all development workflows, ensuring that quality validation occurs throughout the development process rather than as a final step. Documentation maintenance should be achieved through automated test documentation updates and validation coverage reports, ensuring that testing procedures and results remain current and accessible to all development teams. This comprehensive approach to best practices implementation ensures that the testing framework remains effective and efficient as the platform continues to evolve.

## 5. CyclOps Integration and Platform Deployment

---

Within the framework of the CyclOps project, a virtual machine (VM) has been allocated as the primary integration hub for all software components created by collaborating partners. A unified docker-compose file has been created to facilitate smooth orchestration and minimize deployment complexity by automating the creation of all containers necessary for the complete procedure. This method provides a single-entry deployment technique, allowing contributors to establish a fully integrated development and testing environment with little configuration. The project seeks to improve consistency in deployments, expedite integration testing, and foster reproducibility across various workstations and infrastructure configurations by encapsulating each component into a container and coordinating them via a unified script. This technique substantially reduces the entry barrier for new partners, enabling them to concentrate on complying with the standard interface specifications instead of concerning themselves with infrastructure specifics.

Nonetheless, this centralized deployment strategy presents some architectural and operational challenges that must be handled to maintain the integrity and scalability of the integration environment. A major challenge is the presumption that all partner components can cohabit seamlessly inside the same network and runtime environment. Port conflicts, redundant environment variable naming practices, and discrepancies in service discovery procedures might result in erratic behaviour or complete failure. Furthermore, certain services may need certain starting sequences or

extended initialization durations, which, if not included in the orchestration logic, might lead to race situations or cascade problems. Dependency management has an intrinsic risk, especially when components are closely linked to particular versions of common services, resulting in integration vulnerability as those dependencies progress. Inconsistent container runtime environments across various operating systems or host kernel settings may lead to significant complications in debugging and interoperability.

To tackle these issues and guarantee a solid integration process, several mitigation methods have been used. The implementation of stringent interface specifications, encompassing API contracts, data format agreements, and well-defined environment variable namespaces, facilitate the decoupling of partner components and diminish the probability of integration errors. Secondly, the utilization of dependency declarations, health checks, and startup wait-for scripts inside the docker-compose setup helps ensure appropriate sequencing and mitigate temporary issues during container initialization. It is prudent to modularize the orchestration stack, either by subdividing the docker-compose file into logically categorized override files or by employing service labels to facilitate selective deployment during debugging. Ultimately, the implementation of automated integration tests and continuous validation pipelines provides the early detection of breaking changes, so ensuring that updates from individual partners may be securely integrated into the common ecosystem without compromising overall functioning. Establishing frequent review cycles and integration milestones is essential to guarantee that all services progress in accordance with the project's long-term technical strategy.

## 5.1. CyclOps Assets

The CyclOps platform leverages a sophisticated array of infrastructure and software assets to support comprehensive testing and integration processes across the distributed development environment. The testing ecosystem is built upon a robust infrastructure architecture that enables seamless collaboration and validation across multiple partner organizations.

### 5.1.1. Integration Plan

The CyclOps integration plan created in T5.5 contains a variety of resources, including a comprehensive guide on best practices for continuous integration and continuous delivery (CI/CD) specifically designed for the CyclOps project. It provides clear instructions and practical help for building GitLab CI/CD workflow .yaml files, ensuring that automated testing and deployment processes are correctly implemented. Additionally, example repositories with pre-configured CI/CD pipelines have been provided to facilitate efficient workflow setup, along with standardized templates for Dockerfiles and Docker-Compose files to maintain consistency across different projects. The plan also includes standards and guidelines to ensure that all code and dependencies meet security and compliance requirements. Automated checks are also integrated into the CI/CD pipeline to uphold these standards. Finally, code review tools and processes are included to maintain high quality code by identifying potential issues early in the development process (i.e., duplicates, etc.). These resources have been designed to provide comprehensive support and optimize collaborative efforts on the CyclOps project.

### 5.1.2. Infrastructure Assets

The platform operates on a distributed GitLab runner infrastructure comprising 11 virtual machines (VMs), with individual component environment for different testing scenarios. The CyclOps Platform Integration VM serves as the primary hub for complex multi-component integration testing, providing enhanced processing capabilities with 4 CPU cores, 32GB RAM, and 200GB storage. Additionally, a dedicated FIWARE dataspace testing environment with 4 CPU cores and 16GB RAM enables comprehensive interoperability and federated data exchange validation.

Container orchestration forms the backbone of the testing infrastructure, with Docker and Docker Compose standardized across all virtual machines to ensure consistent deployment and testing conditions. This containerized approach eliminates environment specific issues and provides reproducible testing scenarios across different development teams. The infrastructure is



complemented by a reverse-proxy configuration that manages load balancing and provides unified access points for distributed service testing and traffic management.

### 5.1.3. CI/CD Best Practices

The platform implements advanced CI/CD practices through automated pipeline integration using GitLab runners that execute parallel testing across partner virtual machines while maintaining centralized integration validation. This approach ensures that individual components are thoroughly tested in isolation before being integrated into the broader platform ecosystem. The containerized testing strategy guarantees consistent testing conditions across all development teams, eliminating the common problems.

A multi-stage deployment approach has been implemented, progressing from individual component testing through integration testing to final dataspace interoperability validation. Branch protection rules enforce mandatory code review and automated testing before merge approvals, while centralized artifact management ensures proper storage and versioning of Docker images and deployment artifacts. This systematic approach maintains code quality and deployment reliability throughout the development lifecycle.

### 5.1.4. Code Review and Quality Assurance

The platform maintains high code quality through standardized GitLab merge request workflows that require peer review. Repository standardization ensures consistent documentation through README.md files, proper dependency management, and adherence to established coding standards. Version control integration with Git provides comprehensive tracking capabilities, with automated testing triggers activated on code commits to catch issues early in the development process.

Quality gates have been implemented through automated code quality checks and security scanning before deployment, ensuring that only validated, secure code reaches production environments. This comprehensive approach to quality assurance helps maintain the platform's reliability and security standards across all components and layers.

## 5.2. Network Architecture and Service Isolation

CyclOps utilizes Docker's networking functionalities to isolate services and avert inadvertent exposure. By specifying custom bridge networks in the docker-compose.yml, we divide subsets of containers (e.g., AI/ML components, data ingestion services, monitoring agents) into coherent groupings. This method prevents port conflicts and limits intra-service traffic to essential routes alone. Host networking can be selectively utilized for high-performance or low-latency needs, while the default bridge suffices for the majority of microservices. Explicitly designating static container aliases and network subnets augments predictability and streamlines DNS-based service discovery. Moreover, segmenting services across several networks facilitates the implementation of more precise firewall regulations and aids in the enforcement of security protocols, guaranteeing that only authorized services may access sensitive data streams or configuration files.

### 5.3. CI/CD Utilizing GitLab Runners

To maintain the integrity of our integration pipeline, each merge request initiates a GitLab CI task that deploys the complete docker-compose stack within a runner environment. The pipeline uses docker-compose up --abort-on-container-exit to initiate services, succeeded by a series of integration tests (e.g., API smoke tests, end-to-end workflow simulations). If any container terminates with a non-zero exit code, the task fails instantaneously, therefore preventing the introduction of regressions. Artifacts, including test logs and container status reports, are preserved for post-mortem study. This automatic validation phase delivers prompt feedback to developers and sustains the integrity of the communal environment. Furthermore, performance data from test executions are continuously monitored to identify slowdowns or resource constraints, while a nightly comprehensive build guarantees that even seldom modified components remain compatible with the developing codebase.

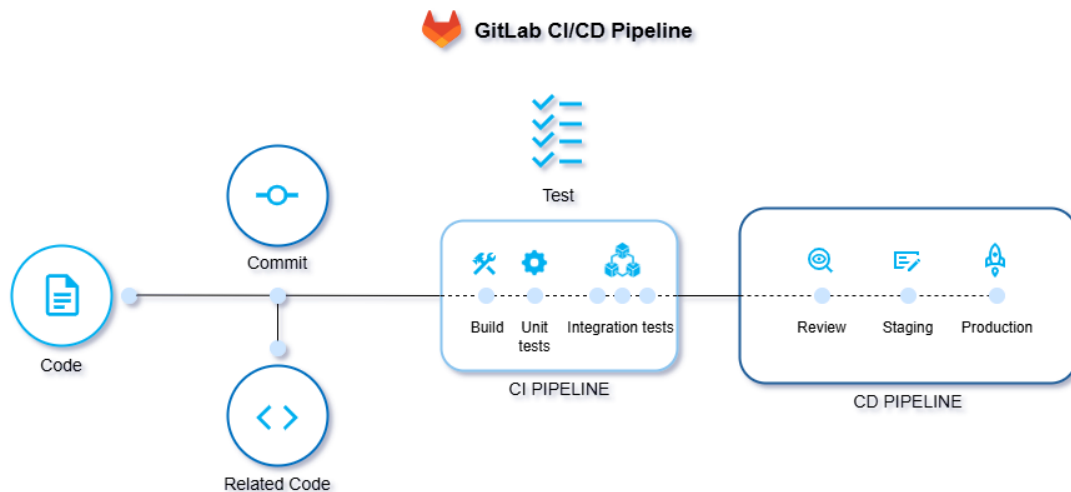


Figure 15. CyclOps CI/CD with GitLab

GitLab Runners are an essential application that integrates with GitLab CI/CD to execute pipeline jobs on designated computing environments. It acts as the agent that is responsible for running CI/CD jobs as defined in `.gitlab-ci.yml` file. To maintain feature compatibility, it's important that the GitLab Runner's major and minor version align with the GitLab instance's major and minor version, since older or newer versions may still work but could lack support for newer features.

Registering a runner sets up the connection between GitLab instance and the machine where GitLab Runner is installed, allowing the runner to pick up and execute jobs. Jobs typically run on the same machine where the runner is installed, but they can also run in isolated environments like Docker containers, or cloud auto-scaled instances. The environment in which jobs run is determined by the executor you choose during registration (e.g., the Docker executor to run jobs in isolated containers, including configurations like Docker-in-Docker). Before registering a runner, a decision must be made about who will have access to it, as runners can be scoped at three levels: instance runners are available to all projects within the entire GitLab instance, group runners serve all projects and subgroups in a specific group, and project runners are limited to individual projects. For monitoring, CyclOps uses the GitLab Runner provided Prometheus integration, enabling the tracking of metrics such as CPU usage and the number of running jobs, with referee workers passing metrics and job data back to GitLab. Key features of GitLab Runner include the ability to run multiple jobs concurrently, use multiple tokens for different servers or projects, limit the number of concurrent jobs per token, and execute jobs locally, over SSH, or inside Docker containers with or without autoscaling across clouds and hypervisors, and connecting to remote SSH servers. GitLab Runner works on GNU/Linux, macOS, and Windows, which means anywhere Docker can run, and supports environment customization, automatic configuration reloads without service restarts, easy installation as a service, caching of Docker containers for faster builds, and an embedded Prometheus metrics server.

Figure 16 depicts a sequential diagram of the process of registering runners and the procedure for requesting and managing jobs. It also indicates which actions utilize registration, authentication, and job tokens. The architecture includes also a runner manager, which reads the `config.toml` file containing one or more runner configurations that appear as individual runners in the GitLab UI. Each runner configuration can run jobs concurrently, and each job runs in a separate process called a runner, which executes tasks on a designated machine, either local to the runner manager (e.g., using the Docker executor) or remote (e.g., via autoscaling with Docker). Once configured and registered, runners seamlessly pick up and execute CI/CD jobs playing a critical role in automating builds, tests, and deployments in a secure and scalable manner for the CyclOps platform. Fig. 17 depicts an example Runner deploying docker environment, retrieving source code, running test scripts, and verifying build success for CI for the Preprocessing and Cleaning component.

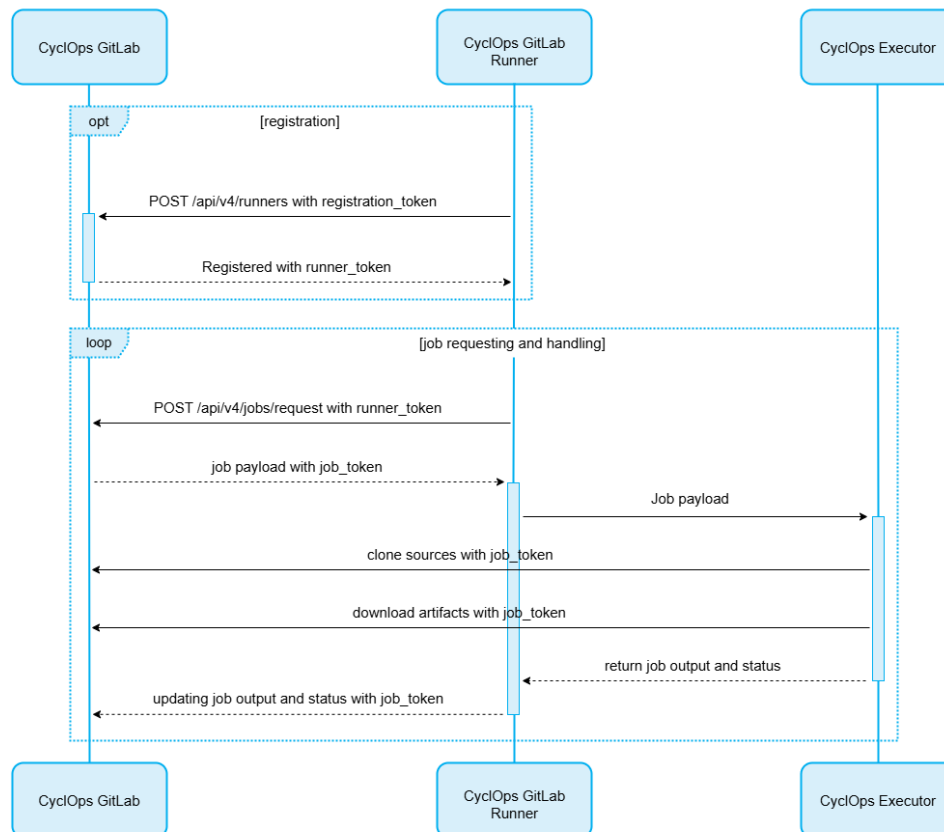


Figure 16. Integration workflow for the CyclOps Project

```

1 Running with gitlab-runner 17.11.1 (96856197)
2 on preprocessing-p02byj8C, system ID: s_572a6339c259
3 Preparing the "docker" executor
4 Using Docker executor with image docker:latest ...
5 Pulling docker image docker:latest ...
6 Using docker image sha256:2da0bd7ecf78eecd7de485edf3565b12c1f71facbeceb4f9b8bfc60805d7b4e9 for docker:latest with digest docker:sha256:eceba5b0fc2fcf83a74c298391c2e09e1adbdaf94ee173611bd6282ec973e7ba ...
7 Preparing environment
8 Running on runner--pdrgy38c-project-66794343-concurrent-0 via cyclops...
9 Getting source from Git repository
10 Fetching changes with git depth set to 20...
11 Initialized empty Git repository in /builds/cyclops4100086/CYCLOPS/.git/
12 Created fresh repository.
13 Checking out c9114a21 as detached HEAD (ref is main)...
14 Skipping Git submodules setup
15 Executing "atop_script" stage of the job script
224 Cleaning up project directory and file based variables
225 Job succeeded
  
```

Figure 17. Example of GitLab Runner deploying docker environment, retrieves source code, runs test scripts, and verifies build success for CI, based on the preprocessing and cleaning component.

### 5.3.1. Logging and Observability

Centralizing logs and data is essential for diagnosing multi-component problems. Currently, we monitor the running status (stdout/stderr) of each component based on: i) log level (debug, info, warning, error), and ii) timestamp. For IT-2, we will implement a streamlined observability stack, where Promtail agents will gather stdout/stderr from each container and transmit them to a centralized Loki instance. Grafana dashboards will retrieve data from Loki and Prometheus to provide insights into application health, resource use, and error rates. Structured logging in JSON format is mandated across services to enable processed queries and alarms. This configuration will facilitate swift root-cause investigation when an integration problem occurs, eliminating the need to SSH into separate containers. Furthermore, Grafana's alerting system is designed to initiate email alerts upon the detection of particular fault thresholds or anomalous behaviours, hence minimizing downtime and human monitoring efforts during production simulations.



Figure 18. Centralized container logs. Example of working output of the Preprocessing and Cleaning component using the iris dataset without error logs.

Cyclops' CI/CD environment ensures deterministic deployments by eschewing changeable latest tags. Each container image is semantically versioned (e.g., component-name: v1.2.3), and these tags are explicitly referenced in the docker-compose.yml. Upon the release of a new patch or feature, the version is incremented in accordance with semantic versioning principles, and the CI pipeline assesses compatibility with the existing stack. This strategy prohibits quiet upgrades and discrepancies between environments, guaranteeing that the team is consistently aware of the precise code in operation at any moment. Additionally, a unified version registry is upheld, recording the compatibility matrix across components, therefore assisting developers in discerning secure upgrade pathways and rollback procedures in the event of complications. Cyclops's CI/CD environment guarantees auditability and complete traceability of system state at any moment by anchoring each deployment to immutable versions.

```
# ---- build stage ----
FROM python:3.11-slim AS base

LABEL maintainer="vnik@fourdotinfinity.com"
LABEL version="1.0.0"
LABEL description="Data Preprocessing And Cleaning"

ENV PYTHONUNBUFFERED=1 \
    PIP_NO_CACHE_DIR=1

WORKDIR /app

# Copy dependency list first (leverages Docker layer caching)
COPY requirements.txt .

# Install dependencies (includes scientific stack)
RUN pip install --upgrade pip && \
    pip install -r requirements.txt

# Copy source
COPY . .

EXPOSE 8765
CMD ["python", "-m", "uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8765"]
```

Figure 19. Example Dockerfile for the Preprocessing and Cleaning component with explicit versioning and environment consistency.

### 5.3.3. Requirements For Integration

To guarantee a smooth and efficient integration process across the CyclOps platform, the following aspects must be ensured for each individual repository and project component:

1. **Unit Testing:** Each repository/project must include robust unit tests to validate that its core functionalities work as expected when tested in isolation.
2. **Load Testing and Performance Evaluation:** Dedicated scripts and guidelines should be developed to perform load tests and assess performance metrics. This is essential to detect potential bottlenecks and verify that each component can handle anticipated workloads.
3. **Component Dependencies and Interaction Mapping:** A precise inventory of each component's dependencies must be created. This should cover all required libraries, frameworks, external tools, and any inter-component relationships.
4. **Integration Tests:** Integration tests should be designed to validate the proper interaction between components. This should encompass dependency calls, API communications, and clear mapping of component interaction flows.
5. **Dockerfiles:** Every component should include a Dockerfile that defines how to build and package it into a Docker container, ensuring uniform deployment and reproducibility across environments.
6. **Docker-Compose Configuration:** A Docker Compose file is prepared to orchestrate how multiple Docker containers interact, specifying service dependencies, network configurations, and volumes.
7. **Environment Setup:** Detailed instructions and configuration files must be provided to define all necessary environment variables and setup scripts for different environments (development, testing, staging, production).
8. **API Reference Documentation:** Complete API documentation must be supplied for each service, including details of all endpoints, request and response formats, authentication requirements, and usage examples.

### 5.3.4. Integration Workflow

In CyclOps, we utilize GitLabs CI/CD pipelines to automate and streamline the continuous integration process. Figure 20 showcases within the CyclOps Project how an integration workflow operates and demonstrates how components have been integrated using the above-mentioned Big-Bang method, where all modules are combined simultaneously and tested as an entire system. While the example

in the diagram highlights two modules (Component A and Component B), the approach can be scaled to encompass more modules as required.

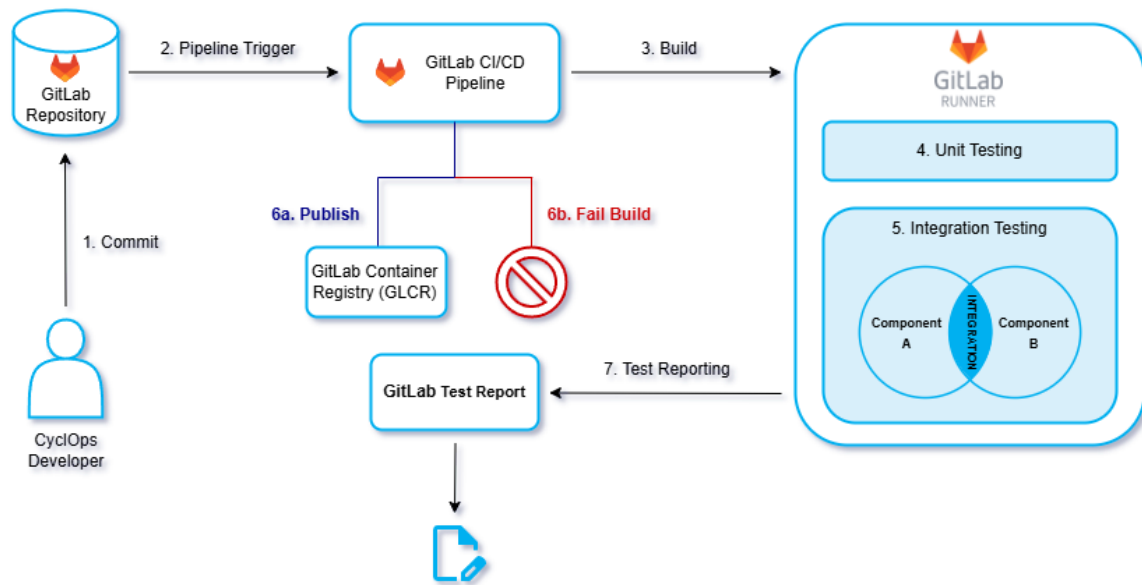


Figure 20. CyclOps Integration workflow

**Step 1 - Commit:** The procedure begins when a component developer makes a commit to the GitLab repository, acting as the trigger point of the CI/CD pipeline.

**Step 2 - Pipeline Trigger:** Based on this commit, a GitLab CI/CD pipeline is activated, enabling the automation of tasks like building, testing, and deploying code directly in the GitLab environment.

**Step 3 - Build:** The GitLab CI/CD pipeline then starts the build process on a GitLab Runner, offering a consistent environment where tasks unfold.

**Step 4 - Unit Testing:** The build step includes conducting unit testing, where each component (such as Component A and Component B) is carefully validated, verifying that individual modules work correctly in isolation.

**Step 5 - Integration Testing:** When unit tests are successfully completed, integration testing is executed. In order to ensure that component A and component B function as a cohesive unit, this step tests their interactions.

**Step 6a & 6b - Publish or Fail:** If all unit and integration tests are passed, the workflow proceeds to publish the build, which entails pushing the Docker image to the GitLab Container Registry (GLCR) (6a). If any tests fail, the workflow stops, and the build process is marked as failed. Consequently, the Docker image is not published, suggesting that issues need to be resolved before attempting integration again (6b). Upon the successful completion of the build process and subsequent publication of the Docker image, it is stored in the GLCR, making it accessible for deployment and further testing.

**Step 7 - Test Reporting:** The workflow incorporates a step that utilizes the Test Reporter to create comprehensive reports of the test results. These reports assist developers be aware of the outcomes of their build and tests, offering insights into any issues that require attention.

## 6. Integration and Testing Results for CyclOps IT-1

### 6.1. Unit Test Matrix

In this section, we provide the Unit Testing Matrix, detailing the verification status performed at the component level to validate the core functionalities of each CyclOps module in line with the initial targets. The status used in the Unit Testing Matrix follows the classification: Successful, Partially Successful, or Planned for IT-2. A status of Successful indicates that the component has been fully implemented and passed its unit tests. Partially Successful means that some testing has been performed and part of the functionality is validated, but full test coverage or final results are still pending. Planned for IT-2 indicates that testing activities for the component are scheduled to take place in the second iteration of the CyclOps platform.

Unit ID	Component	Responsible Partner	Status
UD.1	NLP Chat	DC	Successful
UD.2	User-assisted Intent Interface	TUBS	Successful
UD.3	Exploratory Analysis Engine	TUBS	Successful
UD.4	Pipeline Orchestrator	UPC	Successful
UD.5	Data Ingestion Functions	S5	Successful
UD.6	Data Preprocessing and Cleaning	FDI	Successful
UD.7	Data Discovery	UPC	Successful
UD.8	Data Quality Rules Discovery	UPC	Successful
UD.9	Semantic-based Data Curation	NTTDES	Successful
UD.10	CyclOps Long-Term Storage	CERTH	Successful
UD.11	Data Augmentation	ATOS	Successful
UD.12	CyclOps Lab	CERTH	Successful
UD.13	AI Models Repository	S5	Successful
UD.14	Optimization Functions Repository	FDI	Successful
UD.15	AI Marketplace	CeADAR	Successful
UD.16	Model Deployment	CERTH	Successful
UD.17	Large-Scale Data Management	BSC	Successful
UD.18	Distributed Execution Engine	BSC	Successful
UD.19	IKB Metadata Manager	ONTO	Successful
UD.20	Multimodal Explainability Module	CeADAR	Successful
UD.21	SSI Wallet	ATOS	Successful

Unit ID	Component	Responsible Partner	Status
UD.22	SSI Agent	ATOS	Successful
UD.23	VC Verifier	FIWARE	Successful
UD.24	Data Access	FIWARE	Successful
UD.25	Semantic Interoperability	FIWARE	Successful
UD.26	Data Exchange Broker	FIWARE	Successful
UD.27	Data Exchange Agent	FIWARE	IT-2
UD.28	Publication	FIWARE	IT-2
UD.29	Info Retrieval	EXAI	Successful
UD.30	AI Model Protection	ATOS	IT-2
UD.31	Ontopic Suite	ONTO	Successful
UD.32	Data-Based Data Discovery	UPC	Successful
UD.33	NextiaMG	UPC	Successful
UD.34	Feature Engineering	UPC	Partially Successful
UD.35	Concept Extraction and Categorization Engine	EXAI	Successful
UD.36	Information Extraction and Categorization Engine	EXAI	Successful
UD.37	IKB Exploitation Tool	EXAI	Successful
UD.38	MPBoot	UNIBZ	Successful
UD.39	NLU Module	NTTDES	Successful
UD.40	Semantic-based Data Discovery	NTTDES	IT-2

Table 3. Unit Test Matrix for CyclOps IT-1

## 6.2. Module Integration Matrix

In this section, we present the integration status for each CyclOps layer for IT-1 developments, as defined and aligned with the architectural guidelines from D2.2, and the component developments from D3.1 and D4.1, ensuring coherent interaction across the platform. It is important to note that full inter-layer communication has not yet been achieved at this stage, as this is the first integration iteration. Several mappings are scheduled for completion in the second iteration, based also on the outcomes of WP2, WP3, WP4, and WP6.

The integration status for each component mapping is categorized as Successful, Partially Successful, In Progress or Planned for IT-2. A status of Successful indicates that communication between components is fully established. Partially Successful means integration has started, with

some functionality in place but pending validation. In Progress refers to ongoing development where initial connections may exist but are not yet operational. Finally, Planned for IT-2 designates mappings that are scheduled for implementation in the second iteration of the CyclOps platform.

### 6.2.1. User Intent Layer

The User Intent Layer is the entry point for users, where their needs and objectives are captured and translated into actionable steps within CyclOps. This layer translates and processes user requirements to configure and generate data processing pipelines. The User Intent Layer is divided into six major components: User-assisted Intent Interface, Natural Language Processing (NLP) Chat, Info Retrieval, Intent Compiler, Orchestrator and Exploratory Analysis Engine.

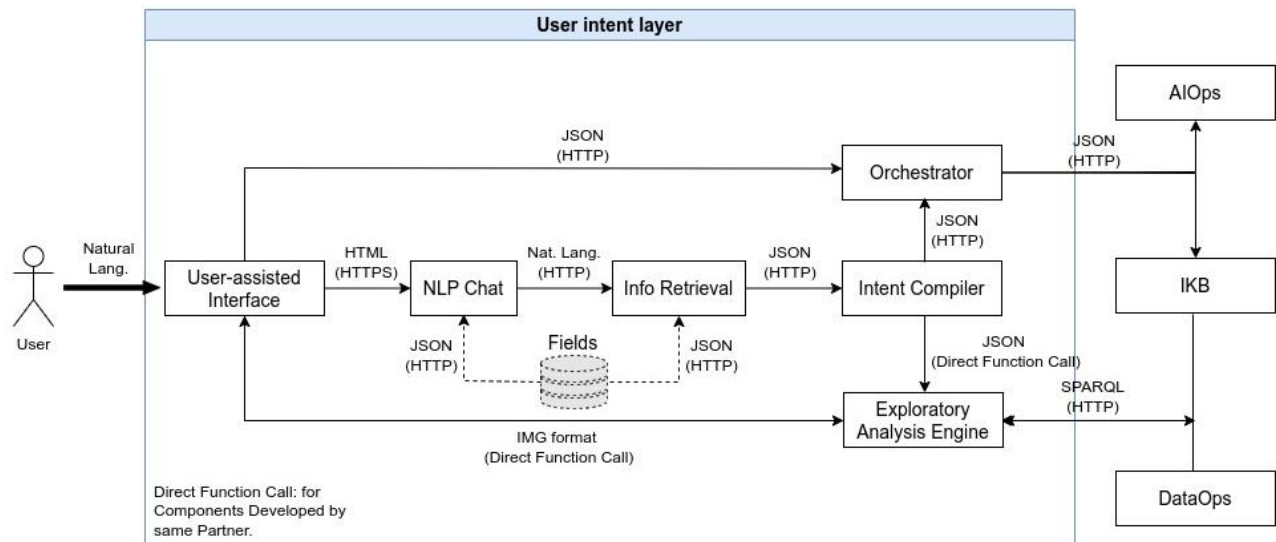


Figure 21. High-level architecture of User Intent Layer (from D4.1)

The following table showcases the various connection mappings of the layer components, along with their integration details, including data types transferred, communication protocols, and integration status.

Integration Endpoints			Responsible For		Data Type & Protocol	Status
ID	CompA	CompB	CompA	CompB		
UI.1	User-assisted Interface	NLP Chat	TUBS	DC	HTML / HTTPS	Successful
UI.2	User-assisted Interface	Orchestrator	TUBS	UPC	JSON / HTTP	IT-2
UI.3	User-assisted Interface	Exploratory Analysis Engine	TUBS	TUBS	Direct Function Call	Successful
UI.4	Exploratory Analysis Engine	User-assisted Interface	TUBS	TUBS	IMG / Direct Function Call	IT-2
UI.5	NLP Chat	Info Retrieval	DC	EXAI	Nat. Lang. / HTTP	Successful
UI.6	Info Retrieval	Intent Compiler	EXAI	TUBS	JSON / HTTP	Successful



Integration Endpoints			Responsible For		Data Type & Protocol	Status
ID	CompA	CompB	CompA	CompB		
UI.7	Fields	NLP Chat	TUBS	DC	JSON / HTTP	Successful
UI.9	Intent Compiler	Orchestrator	TUBS	UPC	JSON / HTTP	IT-2
UI.10	Intent Compiler	Exploratory Analysis Engine	TUBS	TUBS	JSON / Direct Function Call	Successful
UI.11	Orchestrator	IKB Metadata Manager	UPC	ONTO	JSON / HTTP	IT-2
UI.12	Exploratory Analysis Engine	IKB Metadata Manager	TUBS	ONTO	JSON / HTTP	Successful

Table 4. Integration endpoints for User Intent Layer components.

### 6.2.2. Runtime Layer

The Runtime Layer is where data operations are executed and managed. This layer includes DataOps, which provides the services and repositories necessary to support data management and preprocessing tasks within pipelines. AIOps provides methods to complement data pipelines with advanced analysis operations, and the Data and Execution Abstraction (DEA) simplifies the complexity of the underlying infrastructure, making the execution of pipelines transparent to CyclOps users across a distributed environment.

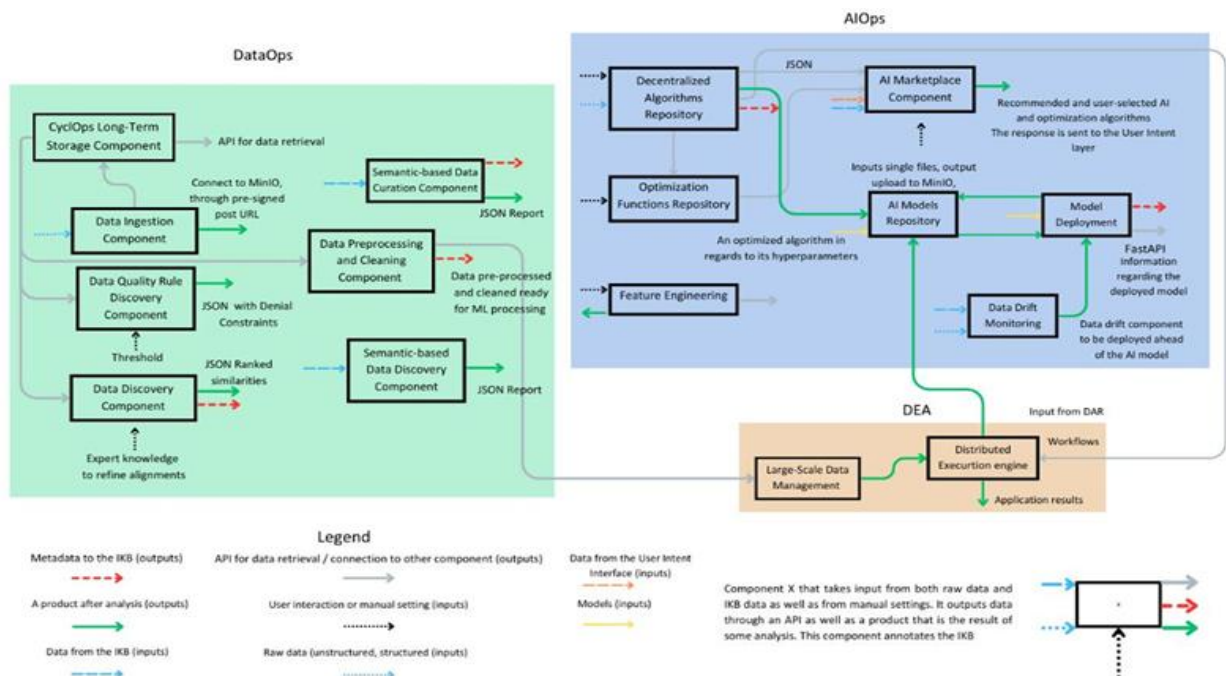


Figure 22. High-level architecture of Runtime Layer (from D3.1).

The following tables show the connection mappings of DataOps, AIOps, and DEA components, including their integration details such as transferred data types, communication protocols, and integration status.



## 6.2.2.1. DataOps

Integration Endpoints			Responsible For		Data Type & Protocol	Status
ID	CompA	CompB	CompA	CompB		
DO.1	Data Ingestion	CyclOps Long-term Storage	S5	CERTH	JSON,XML, CSV, Parquet/ HTTP	Successful
DO.2	CyclOps Long-term Storage	Data Quality Rules Discovery	CERTH	UPC	CSV, Parquet / S3 (HTTP)	Successful
DO.3	CyclOps Long-term Storage	Data Discovery	CERTH	UPC	CSV / S3 (HTTP)	Successful
DO.4	CyclOps Long-term Storage	Data Preprocessing and Cleaning	CERTH	FDI	JSON / RestAPI	Partially Successful
DO.5	Data Preprocessing and Cleaning Functions	Large-Scale Data Management	FDI	BSC	GRPC/SDK/TBD	Partially Successful
DO.6	Data Discovery	IKB Metadata Manager	UPC	ONTO	JSON / RestAPI	Partially Successful
DO.7	IKB Metadata Manager	Semantic-based Data Curation	ONTO	NTTD	JSON-LD	IT-2
DO.8	Semantic-based Data Curation	IKB Metadata Manager	NTTD	ONTO	JSON-LD	IT-2
DO.9	Data Preprocessing and Cleaning Functions	IKB Metadata Manager	FDI	ONTO	JSON / RestAPI	Partially Successful
DO.10	IKB Metadata Manager	Semantic-based Data Discovery	ONTO	NTTD	JSON-LD	IT-2
DO.11	Semantic-based Data Discovery	IKB Metadata Manager	NTTD	ONTO	JSON-LD	IT-2
DO.12	Semantic-based Data Curation	CyclOps Long-term Storage	NTTD	CERTH	CSV, JSON / S3 (HTTP)	Successful
DO.13	Semantic-based Data Discovery	CyclOps Long-term Storage	NTTD	CERTH	CSV, JSON / S3 (HTTP)	IT-2
DO.14	Data Quality Rules Discovery	IKB Metadata Manager	UPC	ONTO	JSON / RestAPI	Partially Successful
DO.15	CyclOps Long-term Storage	Data Augmentation	CERTH	ATOS	CSV, JSON / S3 (HTTP)	IT-2

Table 5. Integration endpoints for DataOps components.

## 6.2.2.2. AIOps

Integration Endpoints			Responsible For		Data Type & Protocol	Status
ID	CompA	CompB	CompA	CompB		
AIO.1	CyclOps Lab	AI Models Repository	CERTH	S5	JSON/HTTP	In progress
AIO.2	CyclOps Lab	AI Marketplace	CERTH	CeADAR	JSON/HTTP	Successful
AIO.3	CyclOps Lab	Distributed Execution Engine	CERTH	BSC	SDK/Bridge (JSON/RestAPI)	Successful
AIO.4	CyclOps Lab	Optimization Functions Repository	CERTH	FDI	JSON / RestAPI	Successful
AIO.5	Optimization Functions Repository	AI Marketplace	FDI	CeADAR	JSON / RestAPI	Partially Successful
AIO.6	AI Models Repository	Model Deployment	CERTH	CERTH	JSON/HTTP	In progress
AIO.7	Model Deployment	AI Models Repository	CERTH	CERTH	JSON/HTTP	In progress
AIO.8	AI Model Protection	Model Deployment	ATOS	CERT	JSON/HTTP	IT-2
AIO.9	Feature Engineering	CyclOps Lab	UPC	CERT	CSV/JSON + Python/HTTP	IT-2
AIO.10	CyclOps Lab	IKB Metadata Manager	CERTH	ONTO	JSON/RestAPI	IT-2
AIO.11	IKB Metadata Manager	AI Marketplace	ONTO	CeADAR	JSON/RestAPI	IT-2
AIO.12	Model Deployment	IKB Metadata Manager	CERTH	ONTO	JSON/RestAPI	IT-2
AIO.13	IKB Metadata Manager	AI Model Protection	ONTO	ATOS	JSON/HTTP	IT-2
AIO.14	AI Marketplace	User-assisted Interface	CeADAR	TUBS	JSON/RestAPI	IT-2

Table 6. Integration endpoints for AIOps components.

### 6.2.2.3. DEA

Integration Endpoints			Responsible For		Data Type & Protocol	Status
ID	CompA	CompB	CompA	CompB		
DEA.1	Large-scale Data Management	Distributed Execution Engine	BSC	BSC	Storage Interface	Successful
DEA.2	Distributed Execution Engine	AI Models Repository	BSC	CERTH	JSON/HTTP	IT-2

Table 7. Integration endpoints for DEA components.

### 6.2.3. Knowledge Layer

At the core of CyclOps lies the Knowledge Layer, which provides a backbone for data operations and decision-making. This component enhances data interoperability and traceability across the platform. It provides mechanisms for managing, processing, and analyzing various data sources by utilizing ontologies, semantic mappings, and knowledge graph technologies. The Knowledge Layer consists of seven primary components: IKB Design (composed of MPBoot, NextiaMG, and Ontopic Suite), IKB Metadata manager, and IKB Exploitation (composed of IKB Concept Extraction and Categorization Engine, Natural Language Understanding Module, and Multimodal Explainability Module).

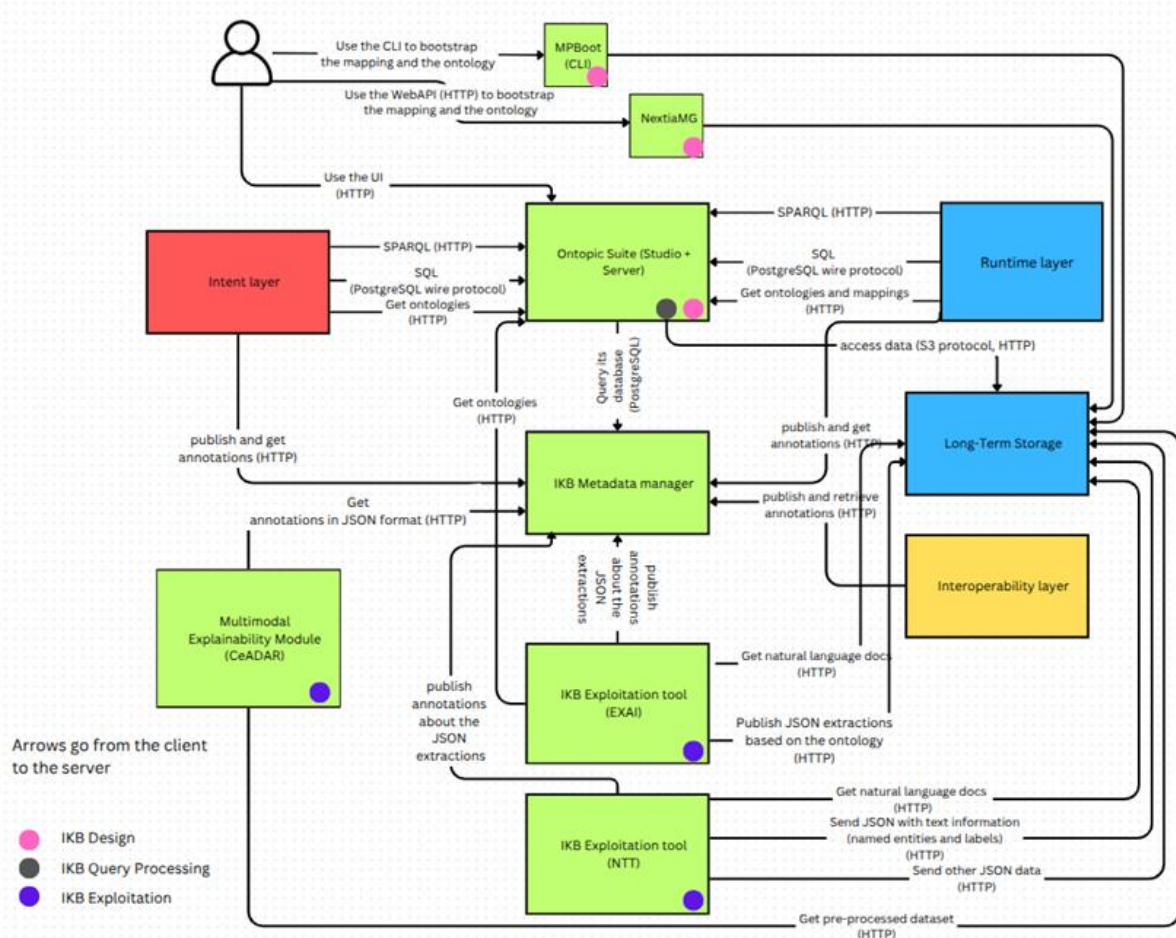


Figure 23. High-level architecture of Knowledge Layer (from D4.1)

The table below presents the connection mappings of layer components along with their integration information (i.e., data types transferred, communication protocols, and integration status).

Integration Endpoints			Responsible For		Data Type & Protocol	Status
ID	CompA	CompB	CompA	Comp B		
KL.1	MPBoot	Long-Term Storage	UNIBZ	CERTH	CSV, JSON, JDBC / S3 (HTTP)	In Progress
KL.2	NextiaMG	Long-Term Storage	UPC	CERTH	CSV, JSON / S3 (HTTP)	Successful
KL.3	Ontopic Suite (Studio+Server))	Long-Term Storage	ONTO	CERTH	CSV, JSON Parquet, S3 (HTTP)	Successful
KL.4	Ontopic Suite (Studio+Server)	IKB Metadata Manager	ONTO	ONTO	PostgreSQL wire protocol	Successful
KL.5	IKB Concept Extraction and Categorization Engine	IKB Metadata Manger	EXAI	ONTO	JSON/RestAPI	Successful
KL.8	Natural Language Understanding	Ontopic Suite (Studio+Server)	NTTD	ONTO	JSON-LD/RestAPI	IT-2
KL.9	Natural Language Understanding	Long-Term Storage	NTTD	CERTH	JSON-LD/RestAPI	IT-2
KL.10	Multimodal Explainability Module	IKB Metadata Manager	CeADAR	ONTO	JSON/RestAPI	IT-2

Table 8. Integration endpoints for Knowledge Layer components.

#### 6.2.4. Interoperability Layer

The Interoperability Layer encompasses the necessary components for integrating the CyclOps platform with external data sources. It ensures that data managed by CyclOps can be accessed, exchanged, and utilized in compliance with FAIR principles and other regulatory standards. The components of the Interoperability Layer are: the SSI Wallet, SSI Agent, DataSpace Connector related subcomponents (e.g., VC Verifier, Data Access, Semantic Interoperability, Data Exchange Broker, Data Exchange Agent, Publication).

For a detailed view of the Interoperability Layer architecture, please refer to [Section 2.4](#), where the corresponding architectural diagrams are provided. The table below outlines the connection mappings of layer components, detailing their integration information, including data types transferred, communication protocols, and integration status.

Integration Endpoints			Responsible For		Data Type & Protocol	Status
ID	CompA	CompB	CompA	CompB		
IL.1	SSI Wallet	SSI Agent	ATOS	ATOS	JSON / RestAPI	Successful
IL.2	SSI Wallet	VC Verifier (Marketplace)	ATOS	FIWARE	JSON / RestAPI / P2M	IT-2
IL.3	SSI Wallet	Keycloak <sup>21</sup>	ATOS	ATOS	QR Code	Successful
IL.4	SSI Agent	Keycloak	ATOS	ATOS	JSON / RestAPI	Successful
IL.5	SSI Agent	User-assisted Intent Interface	ATOS	TUBS	JSON / RestAPI	Successful
IL.6	SSI Agent	Trust Anchor	ATOS	FIWARE	JSON / RestAPI	IT-2
IL.7	SSI Agent	VCVerifier (DSC)	ATOS	FIWARE	JSON / RestAPI / M2M	IT-2
IL.8	Data Retrieval Tool	VCVerifier (DSC)	FIWARE	FIWARE	JSON / RestAPI / M2M	Successful
IL.9	Data Retrieval Tool	Context Broker (DSC)	FIWARE	FIWARE	NGSI-LD	Successful
IL.10	Data Retrieval Tool	Context Broker	FIWARE	FIWARE	NGSI-LD	Successful
IL.11	Data Retrieval Tool	SDMaaS	FIWARE	FIWARE	REST	IT-2

Table 9. Integration points for the Interoperability Layer components

<sup>21</sup> Atos modified version of keycloak to support issuing of Verifiable Credentials

## 7. Requirements Evaluation

In this section, we review all CyclOps requirements defined in WP2 (D2.2) to showcase the platform's readiness for this initial iteration, broken down by each architectural layer. The status used in the requirement evaluation follows a similar classification as in the previous sections: Ready, Partially Ready, or Planned for IT-2. A status of Ready indicates that the requirement has been fully addresses. Partially Ready means that the specific component addresses with some functionality, partially meeting the requirement, full validation and completion. Planned for IT-2 indicates that the requirement is scheduled to take place in the second iteration of the CyclOps platform.

### 7.1. Platform

Requirement	Code	Status
CyclOps should provide user interfaces considering human-in-the-loop.	BZpfF02	Partially Ready
CyclOps should be able to produce data, models and services.	DApfF05	Partially Ready
CyclOps should allow non-technical users to generate data pipelines.	DApfF06	Partially Ready
CyclOps should consider ethical and legal aspects throughout the data lifecycle.	TRpfN18	IT-2
CyclOps should trace information about the execution of a data pipeline.	TRpfN19	Partially Ready
Decrease by a factor of 2 the amount of time in user's manual work to develop data management pipelines.	BZpfN21	Partially Ready
Data privacy: Adhere to GDPR guidelines.	TRdoN80	IT-2
Design the system using a modular architecture. Clear and stable APIs.	DAaiN67	Partially Ready
Provide comprehensive documentation and user guides.	BZdoN43	Partially Ready

Table 10. Requirements for the Platform

### 7.2. User Intent Layer

Requirement	Code	Status
AI model developed by the IHU for detection of data drift across the specific Cyclops's asset consumed by a user to ensure data stability, with the drift communicated to the user through the exploratory analysis engine.	AlihuF01	IT-2
Intent assurance model to ensure accuracy and reliability of the intent compiler.	AlihuF02	Partially ready
Execution of data life cycle management actions automatically according to intents.	DAihuF03	Partially ready
Efficient language for mapping of data sources to intents.	DAihuF04	Ready
Automatic creation of pipelines from a high-level user intention.	DAihuF05	Partially ready
Advanced analytics features for data processing and decision support systems.	DAihuF06	IT-2
Capability to dynamically adapt data visualizations and chats based on user interaction and system feedback.	DAihuF08	IT-2



Requirement	Code	Status
Integration of AI tools for predictive analytics and insights generation based on user activity patterns.	DAihuF09	IT-2
Implementation of robust error handling and data validation mechanisms to enhance stability.	DAihuF11	IT-2
Timely mapping of intents to data sources.	DAihuF12	Ready
Efficient search and pruning of all possible candidate pipelines.	DAihuF13	Partially ready
Pipelines should be executable over the DEA.	DAihuF15	Partially ready
Hybrid systems that combine the features of SQL for structured query efficiency and NoSQL for scalability with unstructured data.	DAihuF18	Partially ready
An interface for both technical and non-technical users.	DAihuF19	Partially ready
Implementation of data encryption and access controls to safeguard user data and ensure compliance with privacy regulations.	SEihuF23	IT-2
Reception of intents from Info Retrieval through an interface.	DAihuN26	Ready
Ability of the module to permit users to query data assets without deep knowledge of query languages through interactive query or search interface.	DAihuN27	Ready
Usability by non-technical users, intents can be expressed in natural language.	DAihuN28	Ready
User-centricity of data visualization and chat interfaces.	DAihuN29	Partially ready
Shall work with uniform data since it will be normalized before reaching the intent compiler.	DAihuN32	Ready
Assurance of real-time responsiveness and adaptive user interface updates based on user interactions.	TRihuN37	IT-2
Resolution of conflicts in the execution of users' intents received by the intent compiler.	TRihuN39	IT-2

Table 11. Requirements for the User Intent Layer.

## 7.3. Runtime Layer

### 7.3.1. DataOps

Requirement	Code	Status
Shall be capable of securing sensitive data (through techniques such as encryption, anonymisation) during the data integration process.	SEdoF76	IT-2
Automatically rank the most suitable datasets/attributes to join/union given a reference one.	DAdoF03	Ready
Shall be able to automatically discover semantic entities from datasets.	DAdoF04	IT-2
Automatically discover implicit constraints in the data sources to help the user detect erroneous elements.	DAdoF05	Ready
The set of constraints discovered must not include redundancies.	DAdoF06	Ready
The set of constraints discovered must not include trivial constraints (i.e., constraints that are satisfied to the same degree in any dataset).	DAdoF07	Ready



Requirement	Code	Status
Shall be able to integrate structured data from various sources offering multiple data ingesting modalities.	DAdoF08	Ready
Support integration with at least three types of unstructured data sources: multimedia, text files, and social media.	DAdoF11	Partially ready
Find possible duplicities amongst different datasets.	DAdoF14	Ready
Define generic quality checks of the data before its integration into the knowledge graph, and specific ones based on the domain specific data model.	DAdoF15	Ready
Shall be able to integrate and pre-process data from multiple diverse sources.	DAdoF28	Ready
Shall offer data transformation and data cleaning processes.	DAdoF29	Ready
Shall be able to discover datasets in heterogeneous formats.	DAdoF31	IT-2
Shall be able to apply quality processes on datasets in heterogeneous formats.	DAdoF32	Ready
Shall be able to integrate structured or semi structured data in heterogeneous formats.	DAdoF33	Ready
Shall discover all the existing constraints within a given precision threshold.	DAdoF49	Ready
Understandable representation of constraints + point out tuples that do not satisfy the identified constraints.	DAdoN46	Partially ready
High precision and recall when discovering related datasets.	DAdoN48	Partially ready
Shall be able to handle large volumes of data from various distributed sources without degradation in performance.	DAdoN61	Partially ready
This component must minimize latency in processing unstructured data.	DAdoN62	IT-2
Shall be able to support standard data exchange formats (e.g., CSV, JSON, XML) and data exchange protocols (e.g., HTTP, MQTT, APIs etc).	DAdoN71	Partially ready
It shall reduce the overall time spent on data discovery and curation and quality processes.	DAdoN73	Partially ready
Shall provide an intuitive UI enabling users to configure the data ingestion process.	BZdoN02	Ready
Data augmentation module will allow an iterative process if it is detected that the training has not been corrected.	DAdgtF09	Partially ready
Data augmentation module shall offer the AI model developer the possibility of avoiding bias issues or unbalanced data by completing the training dataset.	DAdgtF13	Ready
Data augmentation module will help improve the accuracy and stability of the model.	DAdgtN29	Ready
Data augmentation module shall consume resources only during the training phase with minimal impact.	DAdgtN33	Ready

Table 12. Requirements for the DataOps.

### 7.3.2. AIOps

Requirement	Code	Status
Provide tools and frameworks for hyperparameter tuning using techniques like grid search, random search, and Bayesian optimization.	AlaiF01	Partially ready
Serializing the model into a suitable format (ONNX, PMML, Pickle, TensorFlow SavedModel) and serving it through an API for publishing for global and local application use.	AlaiF08	Partially ready
The results provided by AI algorithms and models should be semantically mapped with ontologies and thus with the IKB.	AlaiF09	Partially ready
Ensure support for various ML frameworks, secure distributed computation, and efficient data handling.	AlaiF10	Partially ready
Shall provide algorithms that cover supervised and unsupervised learning, Transfer Learning and Federated Learning.	AlaiF12	Partially ready
Develop decentralized AI algorithms using Federated Learning that will run on edge devices.	AlaiF16	Partially ready
There should be criteria for retrieving algorithms and models from the repository.	BZaiF18	Partially ready
The models developed should be as open and flexible as possible so that they can be used in any context and topic.	BZaiF19	Partially ready
Shall be able to send Metadata to the IKB.	DAaiF24	IT-2
Shall provide a user interface that will allow users to interact with its components within AIOps.	DAaiF25	Partially ready
Shall be able to store, version, and manage machine learning algorithms along with their metadata, such as descriptions and dependencies.	DAaiF28	Partially ready
Shall be able to store, version, and manage AI models along with their metadata, such as descriptions and dependencies.	DAaiF30	Ready
The model should be able to run on edge, resource-constrained devices. Pipeline: optimization libraries will be used to make models more lightweight (i.e. pruning, quantization) User Story: An ML developer needs to make lightweight models to work efficiently on devices without the need of big computational resources.	DAaiF34	Partially ready
Optimize algorithms for resource efficiency, support for cloud and on-premises deployment.	AlaiN41	Ready
Shall ensure quick access to and execution of AI models stored in the repository.	AlaiN47	Ready
Shall be able to support version control for models, enabling users to track changes and revert to previous versions through APIs. Implement secure access controls and ensure models comply with relevant regulations.	AlaiN48	IT-2
Use methods for federated, transfer, and privacy-preserving learning to ensure the robustness of AI approaches.	AlaiN50	Partially ready
Shall provide clear and understandable explanations and metadata for each AI model stored in the repository.	BZaiN57	Partially ready
Shall deliver a consistent and easy-to-understand UI with clear workflows for model management.	BZaiN58	Ready
The components should be able to function independently.	DAaiN68	Ready

Requirement	Code	Status
Align with standards for deploying models in various environments (e.g., cloud, edge). Provide APIs for deploying models and monitoring their performance.	DAaiN75	Partially ready
Shall be able to maintain data integrity, preventing unauthorized modifications to models and their metadata.	SEaiN79	IT-2
Features for users to add detailed descriptions and comments to algorithms for better explainability.	TRaiN81	Partially ready
Provide tooltips and explanatory notes to help users understand the impact of different optimization methods.	TRaiN82	Partially ready
Include clear explanations and visualizations to help users understand deployment status and any issues that arise.	TRaiN86	Partially ready
The results provided by the algorithms and models should be explainable and traceable, and compliant with ontology in order to allow the interoperability.	TRaiN101	Partially ready
Shall be able to provide detailed logs and records for tracking model activities and changes.	TRaiN107	IT-2
Published data generated by the AI algorithms will include a specific disclaimer indicating they have been produced by AI systems.	TRaiN114	IT-2
Provide robust APIs for accessing and sharing algorithms with standardized metadata.	TRaiN119	Ready
Shall suggest suitable AI algorithms based on the selected dataset.	AlaiF39	IT-2
Data drift monitoring module shall use metrics such as accuracy or F1-Score to assess whether the module has a good performance.	DAdgtN65	IT-2
The data drift monitoring module shall be able to detect inputs (during inference phase) that are not similar to the data used to train the AI model to be protected.	DAdgtF03	Partially ready
The data drift monitoring module shall act just before the main AI model; it shall serve as a filter for potential unexpected data.	DAdgtF10	Partially ready
The data drift monitoring module shall provide insights of the reasons for a particular input to be considered anomalous. This shall help the developer to understand the main reasons of the anomalies.	DAdgtF14	Partially ready
The data drift monitoring module shall help reduce the frequency of failures and stabilize the AI model, as it shall not allow any unexpected data to reach the model being protected.	DAdgtN30	IT-2
The data drift monitoring module must manage the false positives ratio in order to not discriminate data that are legitimate.	DAdgtN32	IT-2
Data drift monitoring module shall consume resources only during the inference phase, yet the impact is expected to be minimal in terms of resources, latency, etc.	DAdgtN34	Partially ready
Data drift monitoring module shall ensure accuracy, scalability, real-time monitoring, adaptability to changing data distributions, robustness of AI model, and alerting mechanisms.	DAdgtN48	Partially ready
Data drift monitoring module shall use metrics such as accuracy or F1-Score to assess whether the module has a good performance.	DAdgtN52	IT-2
Shall provide APIs for accessing the optimization functions' metadata, ideally in JSON format.	DAaiF40	Partially ready

Table 13. Requirements for the AIOps

### 7.3.3. Data and Execution Abstraction

Requirement	Code	Status
The DEA component should be able to execute in GPUs.	AldeaF01	Ready
Capability to execute the AIOps algorithms/workflows.	AldeaF02	Partially ready
Must have an API (and protocol) to enable the communication between this component and data spaces (at least 5).	BZdeaF03	IT-2
Must have an API (and protocol) to enable the communication between this component and open-sourced data spaces.	BZdeaF04	IT-2
Support for distributed environments (e.g. clusters, HPC, etc.).	DAdeaF05	Ready
Divide datasets into blocks and distribute those blocks across the resources.	DAdeaF06	Ready
Accommodate datasets that do not fit in the memory of a single node.	DAdeaF07	Ready
Interact with the large-scale data management transparently to the user.	DAdeaF08	Ready
Direct access to intermediate results from DataOps.	DAdeaF09	IT-2
Should be able to exploit data locality.	DAdeaF10	Ready
Interoperability with the Large-scale Data Management component.	DAdeaF25	Ready
Interface should provide an easy/simple programming model.	AldeaN11	Ready
Has to be transparent to the algorithms used by the user.	BZdeaN14	Partially ready
Provide documentation.	BZdeaN15	Partially ready
Ease of use from DataOps and AIOps.	BZdeaN16	Partially ready
Interface should be independent to the underlying data format.	DAdeaN17	Ready
The data abstraction layer has to be flexible enough to provide support for the higher level layers/components.	DAdeaN18	Ready
Heterogeneity on format, structure and semantics of different sources of data.	DAdeaN19	Ready
Hardware computational capabilities should be enough to handle the data operations required for each use case.	DAdeaN20	Ready
The underlying infrastructure must provide enough resources to facilitate the scalability of the use case solutions.	DAdeaN21	Ready
Support for large datasets (Big Data) through distribution.	DAdeaN22	Ready
Mirror the legal requirements defined by the Use Case data.	DAdeaN23	IT-2
Capability of ingesting data coming from DataOps pipelines.	DAdeaN24	Partially ready
Satisfy the requirements given by the user regarding execution time, performance, scalability, etc.	DAdeaN26	IT-2
CyclOps should leverage distributed and parallel processing to boost its performance.	AlpfN13	Ready

Table 14. Requirements for the DEA.

## 7.4. Knowledge Layer

Requirement	Code	Status
Access to the information in the KB should be through natural language requests.	AldgtF01	Ready
Allow for query answering services over metadata specific to provided services (cybersecurity, privacy, etc.).	DAdgtF02	IT-2
The entities extracted should be those related to the ontology defined to allow semantic interoperability.	DAdgtF04	Ready
The resources must be traceable and the metadata will be described and published. Information about the datasets across different domains will be organized and shared.	DAdgtF05	Partially Ready
Allow to automatically create components of the IKB from the data sources (e.g., for query answering).	DAdgtF07	Ready
Automatic creation of metadata constructs in the IKB from the data sources (e.g., source schema, mappings).	DAdgtF11	Ready
Allow for interpretability of data and metadata by CyclOps tool developers.	DAdgtF12	Partially ready
Shall recognise and indicate sensitive data to allow for possible anonymisation.	DAdgtF16	IT-2
The results provided by Information extraction should be easily linkable to the part of the plan text where they were extracted.	TRdgtF18	Ready
The results of an analysis should be justified, or at least the reason for the inference should be explainable.	TRdgtF19	Partially ready
There should be classification modules that respond in times on the order of a few seconds.	TRdgtF20	Partially Ready
There should be entity extraction modules that respond in times on the order of a few seconds.	TRdgtF21	Partially Ready
Minimal user interaction to create the IKB.	DAdgtN27	Partially Ready
The IKB should be consistent with available data and metadata.	DAdgtN28	Ready
Virtual RDF Graph Database shall be used for Integrated Knowledge Base.	DAdgtN41	Ready
Data Lake shall be used for Integrated Knowledge Base.	DAdgtN43	Partially Ready
Heterogeneous sources, integrated under the RDF data model.	DAdgtN44	Partially ready
Re-use of existing ontologies, known ontology design methodologies, well-founded mapping design principles should be followed.	DAdgtN45	Partially Ready
Adhere to the Semantic Web Standards to guarantee interoperability.	DAdgtN46	Ready
The text to be analysed should have enough content for it to be categorized and analysed.	DAdgtN49	Ready
Domain experts should validate the content of the Integrated Knowledge Base to ensure accuracy and relevance.	DAdgtN51	Partially ready
The system should integrate data source schemas, upper and domain ontologies relevant to the use case domains, and incorporate input from domain experts (e.g., use case providers) to support efficient query processing.	DAdgtN53	Partially ready

Requirement	Code	Status
There should be classification modules with high reliability (F-score above 75%).	TRdgtN61	IT-2
There should be entity extraction modules with high reliability (F-score above 75%).	TRdgtN62	IT-2
Retrieve of the entities and relationships from the unstructured resource into a semantic format.	DAdoF26	IT-2
Retrieve of the entities and relationships from a structured data source into a semantic format.	DAdgtF22	IT-2
Shall provide explainability for the developed AI models.	AldgtF22	Ready

Table 15. Requirements for the Knowledge Layer.

## 7.5. Interoperability Layer

Requirement	Code	Status
Provide an SSI Wallet for user to store, manage and present Verifiable Credentials.	SEdgtF57	Ready
Support SSI Agent for issuing Verifiable Credentials with Consumer & Provider Organizations.	SEdgtN58	Ready
Support SSI Agent for verifying Verifiable Presentations for Provider Organizations.	SEdgtN59	Ready
Support Verifiable Credentials for members of the Consumer & Provider organizations to prove their role within an organization according to an agreed trust model for the Data Space.	SEdgtN60	Ready
Standard data collectors (via dataspace connectors) and understanding of standard schemes to describe data (e.g. dcat) is required to collaborate with EU/other data spaces.	DAdoF35	Partially ready
CyclOps should provide a set of standard connectors to interoperate with dataspace.	DApfF09	Ready
CyclOps should allow to explore the content of the IKB.	DApfF11	Partially ready
CyclOps should allow to explore the data it stores (intermediate results).	DApfF12	Partially ready
Cover the lifecycle of high-quality semantic interoperable data production and exploitation within the context of Data Spaces.	DAdgtF06	Partially ready
Shall provide interoperability with standard data formats, protocols, and data spaces using standards like FIWARE's NGSI-LD and Context Broker implementations.	BZaiN60	Ready

Table 16. Requirements for Interoperability Layer



## 8. Conclusions

---

In this first integrated release of the CyclOps platform, we have successfully brought together the architectural blueprints and component developments from WP2 through WP5 into a cohesive prototype release. Building on the requirements (D2.1) and architectural design (D2.2), the essential AI/data modules (D3.1) and the knowledge layer modules and human-centric interfaces (D4.1) have been brought together by the integration work in WP5 (T5.5). Through Tasks T5.1–T5.5, we have analysed the interoperability requirements with European Data Spaces, embedded decentralized identity and access controls, analysed requirements for the DOME marketplace, and carried out in-lab validation.

It is important to note that while some mappings have already been established to enable communication between certain layers, full cross- and inter-layer communication across all components has not yet been achieved at this stage. This is expected, as this is the first integration iteration, which focuses primarily on establishing the essential inter-layer communications rather than complete end-to-end integration. Several mappings are scheduled for completion in the second iteration, based also on the outcomes of WP2, WP3, WP4, and WP6. Quantitatively, against the 160 component requirements defined in WP2, we report the overall readiness level being 78% with 125/160 requirements being ready or partially ready. In detail, the CyclOps Platform covered 78% (7/9) of requirements, the User Intent Layer 63% (14/22), the Runtime Layers DataOps 85% (23/27), AIOps 75% (30/40), DEA 80% (20/25). Furthermore, the Knowledge Layer covered 77% (21/27) of the requirements, while the Interoperability Layer 100% (10/10). In addition to these metrics, through D5.1 we are also tracking KPI-4.1: Release of two intermediate CyclOps software versions (one release delivered; partially ready).

Moving forward, we will focus on closing the gaps identified in D3.1 and D4.1 in order to finalize the second intermediate release under KPI-4.1. WP2 will continue to steer updates to the platform architecture, ensuring that newly emerged requirements arising from both consortium feedback and evolving markets and regulations, as well as from the validation efforts coming from the use cases (WP6). Overall, the ongoing collaboration and thorough validation efforts will advance the project towards achieving a comprehensive FAIR-compliant, governed, and interoperable AI-driven data lifecycle platform. The achievement realized in this initial release establishes a robust platform for future advances, ensuring ongoing advancements and innovations in the CyclOps project.

## ANNEX: List of Modules Included in CyclOps Release 1

---

### A.1. User Intent Layer

#### A.1.1. User-assisted Intent Interface

**Description:** The User-assisted intent interface supports user login and enable intent creation either through natural language or a structured format, where users can select their desired execution pipeline, made available by the Orchestrator, as well as explore and interact with data retrieved by the Exploratory Analysis Engine. This interface includes functionalities for managing user identities, directing the users to the NLP Chat interface, managing sessions, and assisting with the creation of execution pipelines.

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/user-assisted-intent-interface>

**Installation Instructions:**

<https://gitlab.com/cyclops4100006/user-assisted-intent-interface/-/blob/main/README.md>

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.3 / 24.04
- ii) Docker, version 28.1.1
- iii) Docker Compose version v 1.29.2
- iv) Git version 2.43.0

#### A.1.2. Natural Language Intent Interface

**Description:** The Natural Language Intent Interface is responsible for parsing users' high-level intents in natural language and extracting the actual intention and valuable information in the request in a structured format, by means of an NLP chat that extracts all necessary information from user during their interaction and an Info Retrieval component that translates user requests into structured intents processed by other CyclOps modules. The intents could be exploratory or analytical. For exploratory intents, the user inputs in natural language are translated into queries to the IKB, and the requested data is then returned to the user. For the analytical intents, the intents identified from the user input are forwarded to the Pipeline Generator.

##### A.1.2.1. NLP Chat

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/chatnlp>

**Installation Instructions:**

[https://gitlab.com/cyclops4100006/chatnlp/-/blob/main/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/chatnlp/-/blob/main/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

Externally provided

##### A.1.2.2. Info Retrieval

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/info-retriever>

**Installation Instructions:**

[https://gitlab.com/cyclops4100006/info-retriever/-/blob/main/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/info-retriever/-/blob/main/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

- FastAPI: Web framework for building the API.

- uvicorn: ASGI server to run the API.
- openai: Interaction with OpenAI's GPT models.
- instructor: Library for structured outputs with OpenAI.
- pydantic: Data validation and settings management.

### A.1.3. Intent Compiler

**Description:** The Intent compiler is responsible for compiling structured user intents received from the info Retrieval to actionable parameters called policies. These policies can be either exploratory or analytical. Exploratory policies define the parameters needed to retrieve relevant data or information from the IKB, while analytical policies specify the data analytics functions to be performed on selected data.

**Link to repository:** <https://gitlab.com/cyclops4100006/user-assisted-intent-interface>

**Installation Instructions:** [https://gitlab.com/cyclops4100006/user-assisted-intent-interface/-/blob/main/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/user-assisted-intent-interface/-/blob/main/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

- v) Ubuntu 22.04.3 / 24.04
- vi) Docker, version 28.1.1
- vii) Docker Compose version v 1.29.2
- viii) Git version 2.43.0

### A.1.4. Exploratory Analysis Engine

**Description:** The Exploratory Analysis Engine allows users to interact with the data and metadata available in the integrated knowledge base. It has a JSON-to-SPARQL translator that automatically generates SPARQL queries based on policies provided by the Intent Compiler, which supplies them in JSON format. After generating the SPARQL query, the Exploratory Analysis Engine sends the query to an endpoint in the IKB, which returns the related data or no data when no correspondence exists in the dataset.

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/user-assisted-intent-interface>

**Installation Instructions:** [https://gitlab.com/cyclops4100006/user-assisted-intent-interface/-/blob/main/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/user-assisted-intent-interface/-/blob/main/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

- ix) Ubuntu 22.04.3 / 24.04
- x) Docker, version 28.1.1
- xi) Docker Compose version v 1.29.2
- xii) Git version 2.43.0

### A.1.5. Orchestrator

**Description:** The Pipeline Orchestrator receives the user intents in a structured format from the NLP and User-assisted Intent interfaces and compiles them into pipelines. It contains two major sub-components which are Intent Compiler and Pipeline Orchestrator respectively. Intent compiler is responsible for parsing and storing the users' high-level inputs, verifying and matching the intents with adequate policies and compliance checking. The orchestrator on the other hand manages and dictates the actions a lifecycle controller performs in setting up data processing or task execution pipelines to fulfil the so-called user intents.

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/pipeline-orchestrator>

**Installation Instructions:** <https://gitlab.com/cyclops4100006/pipeline-orchestrator/-/blob/main/README.md>

**List of dependencies and third-party software Included in the current release:**

- i) Docker, version 28.0.4
- ii) Docker Compose version v2.27.0
- iii) Git version 2.25.1

## A.2. Runtime Layer

### A.2.1. DataOps

#### A.2.1.1. Data Ingestion Functions

**Description:** The Data Ingestion Function for IT-1 is provided as a **closed-source component (SaaS)**. It enables users to ingest structured data via diverse modalities, including direct file uploads, uploads through the data provider's APIs, and uploads via the component's own APIs. This function supports various data formats, specifically CSV, JSON, XML, and Parquet files, and upon processing, securely stores the uploaded data in the LTS (MinIO).

**Version:** 1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/data-ingestion-function>

**Installation Instructions:** [https://gitlab.com/cyclops4100006/data-ingestion-function/-/blob/main/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/data-ingestion-function/-/blob/main/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

- i) PostgreSQL v15
- ii) Redis v5
- iii) MinIO RELEASE.2025-04-08T15-41-24Z
- iv) Vault v1.5.5
- v) MongoDB v7
- vi) Elasticsearch v7.17.3
- vii) Zookeeper v7.3.2
- viii) Confluent Kafka v7.3.2
- ix) Kafka UI v0.5.0
- x) SpiceDB v1.20

#### A.2.1.2. Data Preprocessing and Cleaning Functions

**Description:** The Data Preprocessing and Cleaning Functions preprocess various data types, including time series and tabular data. These functions facilitate data cleaning, normalization, and standardization, ensuring a high-quality and consistent data transformation. The initial input comprises raw data, which can be either structured or unstructured from the LTS. The final output is pre-processed and cleaned data that is ready for ML processing from the Large-Scale Data Management. Additionally, this component generates dual outputs, producing two JSON files, one for the preprocessed dataset and another for its metadata. The output filenames are generated automatically using the original file name (or dataset identifier), along with a descriptive suffix and a timestamp.

**Version:** v1.0

**Link to repository:** [https://gitlab.com/cyclops4100006/CYCLOPS/-/tree/main/Preprocessing%20Module?ref\\_type=heads](https://gitlab.com/cyclops4100006/CYCLOPS/-/tree/main/Preprocessing%20Module?ref_type=heads)

**Installation Instructions:** [https://gitlab.com/cyclops4100006/CYCLOPS/-/blob/main/Preprocessing%20Module/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/CYCLOPS/-/blob/main/Preprocessing%20Module/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 24.04.2 LTS
- ii) Docker version: 27.5.1
- iii) Docker Compose version: v1.29.2
- iv) Git version: 2.43.0
- v) Python 3.13
- vi) Pandas 2.2.3
- vii) NumPy 2.2.6
- viii) Scikit-learn 1.6.1
- ix) SciPy 1.15.3

#### A.2.1.3. Data Discovery

**Description:** The Data-based Data Discovery component aims to efficiently identify datasets that meet specific information needs through a profile-based approach. It specifically discovers joinable attributes for a given query attribute.

**Version:** latest

**Link to repository:** <https://gitlab.com/cyclops4100006/datadiscovery>

**Installation Instructions:**

[https://gitlab.com/cyclops4100006/datadiscovery/-/blob/main/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/datadiscovery/-/blob/main/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.3
- ii) Docker, version 27.3.1
- iii) Docker Compose version v2.30.3
- iv) Git version 2.39.5

#### A.2.1.4. Data Quality Rules Discovery

**Description:** The Data Quality Rules Discovery automated component is aimed at effectively discover data quality rules to ensure coherence and avoid redundancy on a given datasets. It primarily centers on the identification of true denial constraints, which offer flexibility to represent many types of constraints with one single language and offer efficiency of their compliance checking.

**Version:** latest

**Link to repository:** <https://gitlab.com/cyclops4100006/dqrulediscovery>

**Installation Instructions:**

[https://gitlab.com/cyclops4100006/dqrulediscovery/-/blob/main/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/dqrulediscovery/-/blob/main/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.3
- ii) Docker, version 27.3.1
- iii) Git version 2.39.5

#### A.2.1.5. Semantic-based Data Curation

**Description:** This component aims to ensure the quality of the data in semantic format, by using SHACL shape, and to evaluate the semantic resources with FAIR principles. SHACL is a language for validating RDF graphs against a set of conditions, such as the minimum value of a property, the datatype, amongst others. This component also helps to discover implicit constraints, enabling the detection of wrong elements or duplication in the data. Additionally, it considers FAIR principles, in order to calculate the quality of the data following its four pillars, Findability, Accessibility, Interoperability, and Reuse.

**Version:** v1.0

**Link to repository:** [CYCLOPS / semantic-based-data-curation · GitLab](#)

**Installation Instructions:** [README.md · main · undefined · GitLab](#)

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.5 LTS
- ii) Docker version 28.0.4
- iii) Docker Compose version v2.34.0-desktop.1
- iv) git version 2.48.1.windows.1

**A.2.1.6. Semantic-based Data Discovery**

**Description:** This component aims to find similarities amongst datasets in semantic format. To this end, it uses their corresponding ontologies and the textual description of the ontology classes that allow automatically identifying the mappings with the highest score.

**Version:** v1.0

**Link to repository:** [CYCLOPS / semantic-based-data-discovery · GitLab](#)

**Installation Instructions:** [README.md · main · CYCLOPS / semantic-based-data-discovery · GitLab](#)

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.5 LTS
- ii) Docker version 28.0.4
- iii) Docker Compose version v2.34.0-desktop.1
- iv) git version 2.48.1.windows.1

**A.2.1.7. CyclOps Long-Term Storage**

**Description:** The Long-Term Storage (LTS) component is a centralized storage solution for managing structured, semi-structured, and unstructured data across different components. The main goal of LTS is to ensure that CyclOps modules can store, retrieve, and share data efficiently.

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/long-term-storage>

**Installation Instructions:**

[https://gitlab.com/cyclops4100006/long-term-storage/-/blob/main/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/long-term-storage/-/blob/main/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.3
- ii) Minio
- iii) Docker, version 26.1.1
- iv) Docker Compose version v2.27.0
- v) Git version 2.34.1

**A.2.1.8. Data Augmentation**

**Description:** The Data Augmentation component serves as a synthetic data generator, improving the robustness, generalisation, and overall effectiveness of AI models, while helping to reduce overfitting, particularly when working with limited datasets.

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/dataaugmentation>

**Installation Instructions:**

<https://gitlab.com/cyclops4100006/dataaugmentation/-/blob/main/README.md>

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.3
- ii) Docker, version 26.1.1
- iii) Docker Compose version v2.27.0



iv) MinIO

## A.2.2. AIOps

### A.2.2.1. CyclOps Lab

**Description:** CyclOps Lab is a modular, containerized AI development environment designed to streamline collaborative machine learning workflows. It integrates JupyterLab with Elyra for visual pipeline authoring, a GitLab sync mechanism for automated repository data mirroring in the object storage (LTS), and MLflow (Model Deployment) for experiment tracking.

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/cyclops-lab>

**Installation Instructions:** <https://gitlab.com/cyclops4100006/cyclops-lab/-/blob/main/README.md>

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.3
- ii) Docker, version 27.2.0
- iii) Docker Compose version v2.29.2
- iv) Git version 2.42.0
- v) JupyterLab with Elyra Pipeline Editor
- vi) Repo Sync Script (`clone_gitlab_repos.sh`) for GitLab API repo mirroring
- vii) curl, jq (used for API interaction and JSON parsing)
- viii) Python packages (included in container images, assumed based on Elyra and MLflow requirements)

### A.2.2.2. AI Models Repository

**Description:** An object storage solution utilized for storing, managing and versioning trained AI models.

**Version:** v0.2.0

**Link to repository:** <https://gitlab.com/cyclops4100006/ai-models-repository>

**Installation Instructions:**

[https://gitlab.com/cyclops4100006/ai-models-repository/-/blob/main/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/ai-models-repository/-/blob/main/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

- i) Minio: minio:RELEASE.2025-04-22T22-12-26Z
- ii) PostgreSQL: postgres:15
- iii) Docker version: v27.5.1
- iv) Docker compose version: v2.35.1
- v) Git version: v2.43.0

### A.2.2.4. Optimization Functions Repository

**Description:** The Optimization Functions Repository (OFR) provides an extensive collection of methods to improve optimization processes. Key features include: a) a diverse range of compression techniques that reduce model size while maintaining accuracy, leading to faster inference and reduced computational costs, and b) automated hyperparameter tuning to identify optimal parameters for improved model performance. Additionally, OFR functions include metadata that is transmitted to the AI Marketplace for accurate function selection based on user needs.

**Version:** v0.1.0

**Link to repository:** [https://gitlab.com/cyclops4100006/CYCLOPS/-/tree/main/OFR?ref\\_type=heads](https://gitlab.com/cyclops4100006/CYCLOPS/-/tree/main/OFR?ref_type=heads)

**Installation Instructions:**

[https://gitlab.com/cyclops4100006/CYCLOPS/-/blob/main/OFR/Readme.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/CYCLOPS/-/blob/main/OFR/Readme.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 24.04.2 LTS
- ii) Docker version: 27.5.1
- iii) Docker Compose version: v1.29.2
- iv) Git version: 2.43.0
- v) Python 3.13
- vi) Pandas 2.2.3
- vii) NumPy 2.2.6
- viii) Scikit-learn 1.6.1
- ix) SciPy 1.15.3

**A.2.2.5. Feature Engineering**

**Description:** The Feature Engineering component is designed to assist the CyclOps user in selecting the appropriate features for their machine learning tasks based on statistical analysis, correlation analysis, and feature importance metrics. It highlights the best features related to the target of the ML task, enabling users to build more accurate and efficient models.

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/featureengineering>

**Installation Instructions:**

[https://gitlab.com/cyclops4100006/featureengineering/-/blob/main/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/featureengineering/-/blob/main/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.3

**A.2.2.6. AI Marketplace**

**Description:** The AI Marketplace is a web-based service which allows users to easily browse and select suitable algorithms according to their needs.

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/ai-marketplace>

**Installation Instructions:**

<https://gitlab.com/cyclops4100006/ai-marketplace/-/blob/main/README.md>

**List of dependencies and third-party software Included in the current release:**

- ii) Ubuntu 22.04.3
- iii) Docker, version 26.1.1
- iv) Docker Compose version v2.27.0
- v) Git version 2.34.1

**A.2.2.7. Model Deployment**

**Description:** Model deployment in CyclOps provides a structured approach to enable efficient model management, versioning, and accessibility. Models are deployed using MLflow and saved in MinIO in the desired serialization format.

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/model-deployment>

**Installation Instructions:**

[https://gitlab.com/cyclops4100006/model-deployment/-/blob/main/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/model-deployment/-/blob/main/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release**

- i) Ubuntu 22.04.3
- ii) Docker, version 26.1.1

- iii) Docker Compose version v2.27.0
- iv) Git version 2.34.1
- v) MLflow

#### A.2.2.8. AI Model Protection

**Description:** The AI Model Protection component helps protect AI models during production by monitoring incoming data. By detecting unusual patterns and data drift early on, it helps maintain model accuracy, avoid performance issues, and reduce the risk of harmful or unexpected inputs

**Version:** v0.1

**Link to repository:** <https://gitlab.com/cyclops4100006/datadriftmonitoring>

**Installation Instructions:**(IT2)

<https://gitlab.com/cyclops4100006/datadriftmonitoring/-/blob/main/README.md>

**List of dependencies and third-party software Included in the current release**

- i) Ubuntu 22.04.3
- ii) Docker, version 26.1.1
- iii) Docker Compose version v2.27.0

#### A.2.3. DEA

**Description:** The Distributed Execution Abstraction component provides the capability to manage the execution of a given algorithm in a distributed environment. To this end, provides a simple interface that can be used from the cyclops-lab. Internally, it is composed by two subcomponents in charge of the data management, and another in charge of the execution. These subcomponents are included within the same repository and are deployed with the same docker compose.

**Version:** v0.0.1

**Link to repository:** <https://gitlab.com/cyclops4100006/dea>

**Installation Instructions:** <https://gitlab.com/cyclops4100006/dea/-/blob/main/README.md>

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.3
- ii) Docker, version 26.1.1
- iii) Docker Compose version v2.27.0
- iv) Git version 2.34.1
- v) PyCOMPSs 3.3.3
- vi) dataClay 4.2.0

##### A.2.3.1. Large-Scale Data Management

**Description:** The large-scale data management component assists the distributed execution engine with an efficient data storage administration. It is provided by dataClay.

**Version:** v4.2.0

##### A.2.3.2. Distributed Execution Engine

**Description:** The distributed execution engine component manages the execution of the given algorithm and exploit the parallelism if possible. It is provided by PyCOMPSs.

**Version:** v3.3.3

### A.3. Knowledge Layer

#### A.3.1. IKB Metadata Manager

**Description:** Manages the IKB metadata (except ontologies and mappings). It provides a Web API where datasets are registered and annotations are stored.

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/ikb-metadata-manager>

**Installation Instructions:** <https://gitlab.com/cyclops4100006/ikb-metadata-manager#deployment-in-production>

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.3
- ii) Docker, version 26.1.1
- iii) Docker Compose version v2.27.0
- iv) Git version 2.34.1

### A.3.2. Concepts Extraction and Categorization Engine

**Description:** The concept extraction and categorization engine in CyclOps systematically processes unstructured text from the Integrated Knowledge Base (IKB) using a hybrid approach that combines traditional machine learning and large language models. It extracts key concepts, entities, and relationships using ontologies, and categorizes them based on context and predefined schemas. The engine is domain-agnostic, supports automated annotation, and ensures data traceability and consistency across sources. This enhances data interoperability and enables robust, scalable information structuring within dynamic data environments.

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/information-extraction-and-categorization-engine>

**Installation Instructions:** <https://gitlab.com/cyclops4100006/information-extraction-and-categorization-engine/-/blob/main/README.md>

**List of dependencies and third-party software Included in the current release:** External API

### A.3.3. IKB Exploitation tool. Natural Language Understanding (NLU)

**Description:** The NLU component allows CyclOps to enrich texts from the IKB with extra information provided by NLP (Natural Language Processing) algorithms. The algorithms considered here are Named Entity Recognition (NER), Text Classification (TC) and Entity Linking (EL). Given a text, NER scans it for named entities, TC returns the topics the text is about and EL connects the entities returned by NER to the information about them that can be found on DBPedia<sup>22</sup>.

**Version:** v1.0

**Link to repository:** [CYCLOPS / ikb-explotation-tool-nlu · GitLab](#)

**Installation Instructions:** [README.md · main · undefined · GitLab](#)

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.5 LTS
- ii) Docker version 28.0.4
- iii) Docker Compose version v2.34.0-desktop.1
- iv) git version 2.48.1.windows.1

#### A.3.3.1. Multimodal Explainability Module

**Description:** Provides a comprehensive set of explainability techniques for AI models.

**Version:** v1.0

**Link to repository:** <https://gitlab.com/cyclops4100006/multimodal-explainability-module>

---

<sup>22</sup> <https://es.dbpedia.org/>

**Installation Instructions:**

<https://gitlab.com/cyclops4100006/multimodal-explainability-module/-/blob/main/README.md>

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.3
- ii) Docker, version 26.1.1
- iii) Docker Compose version v2.27.0
- iv) Git version 2.34.1
- v) Flask==3.0.3
- vi) numpy==1.26.4
- vii) pandas==2.1.3
- viii) matplotlib==3.10.0
- ix) scikit-learn==1.6.1
- x) lime==0.2.0.1
- xi) shap==0.42.1
- xii) plotly==6.0.0

**A.3.3.2. Ontopic Suite (ONTO)**

**Description:** Components for mapping editing (Ontopic Studio mapping editor) and for processing SPARQL and SQL queries over the IKB (IKB Query Processor)

**Version:** cyclops-0.3.0

**Link to repository:** <https://gitlab.com/cyclops4100006/ontopicsuite>

**Installation Instructions:** <https://gitlab.com/cyclops4100006/ontopicsuite#setup-and-installation>

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.3
- ii) Docker, version 26.1.1
- iii) Docker Compose version v2.27.0
- iv) Git version 2.34.1

**A.3.3.3. MPBoot (UNIBZ)**

**Description:** Automated bottom-up bootstrapper of mapping and ontology from relational databases (without a reference ontology)

**Version:** 0.1

**Link to repository:** <https://gitlab.com/cyclops4100006/mpboot>

**Installation Instructions:** <https://gitlab.com/cyclops4100006/mpboot/-/blob/master/README.md>

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.3
- ii) Docker, version 26.1.1
- iii) Docker Compose version v2.27.0
- iv) Git version 2.34.1
- v) Dremio 24.0
- vi) PostgreSQL 14.2

**A.3.3.4. NextiaMG**

**Description:** Automated bottom-up schema bootstrapper and R2RML mapping generator from structured and semi-structured data (without a reference ontology)

**Version:** Latest

**Link to repository:** <https://gitlab.com/cyclops4100006/nextiamg>

**Installation Instructions:**

[https://gitlab.com/cyclops4100006/nextiamg/-/blob/main/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/nextiamg/-/blob/main/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 22.04.3
- ii) Docker, version 26.1.1
- iii) Docker Compose version v2.27.0
- iv) Git version 2.34.1

## A.4. Interoperability Layer

### A.4.1. SSI Wallet

**Description:** A user-centric identity android mobile application that enables individuals to securely store, manage, and present W3C Verifiable Credentials (VCs). The wallet supports cryptographic key management, credential lifecycle operations (issuance, storage, revocation), and the generation of Verifiable Presentations (VPs) in compliance with OID4VCI & OIDC4VP respectively as well as the EBSI Guidelines.

**Version:** Latest

**Link to repository:** <https://gitlab.com/cyclops4100006/ssi-wallet/>

**Installation Instructions:**

[https://gitlab.com/cyclops4100006/ssi-wallet/-/blob/main/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/ssi-wallet/-/blob/main/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release:**

- i) Android v28

### A.4.2. SSI Agent

**Description:** A core identity service component responsible for supporting issuing of W3C-compliant Verifiable Credentials (VCs) to members of both consumer and provider organizations. It includes Keycloak (modified to support SSI) which users register with to be able to issue employee credentials. It additionally supports the validation of Verifiable Presentations for role-specific credentials to employees. These credentials enable decentralized, attribute-based access control mechanisms to the CYCLOPS platform and data spaces.

**Version:** Latest

**Link to repository:** <https://gitlab.com/cyclops4100006/ssi-agent>

**Installation Instructions:** <https://gitlab.com/cyclops4100006/ssi-agent#deployment>

**List of dependencies and third-party software Included in the current release:**

- i) Ubuntu 24.04
- ii) Docker, version 28.0
- iii) Docker Compose version v2.33.1
- iv) Git version 2.43.0

### A.4.3. VC Verifier

**Description:** Service supporting the OID4VP flow

**Version:** 5.0.0

**Link to repository:** <https://gitlab.com/cyclops4100006/vcverifier>

**Installation Instructions:** <https://github.com/FIWARE/VCVerifier?tab=readme-ov-file#install>

**List of dependencies and third-party software Included in the current release**

- i) Ubuntu 24.04
- ii) Docker, version 28.0
- iii) Docker Compose version v2.33.1



- iv) [Additional libraries](#)

#### A.4.4. Data Access

**Description:** Service executing the OID4VP M2M Flow to retrieve data from a dataspace

**Version:** latest

**Link to repository:** <https://gitlab.com/cyclops4100006/dataspace retrievals script>

**Installation Instructions:** <https://gitlab.com/cyclops4100006/dataspace retrievals script#example-start>

**List of dependencies and third-party software Included in the current release**

- i) Ubuntu 24.04
- ii) Docker, version 28.0
- iii) Docker Compose version v2.33.1
- iv) Curl, 8.14.1-r0
- v) Jq, 1.8.0-r0
- vi) Openssl, 3.5.0-r0

#### A.4.5. Semantic Interoperability

**Description:** Service exposing the smart data models python functionality as Rest API

**Version:** latest

**Link to the repository:** [https://gitlab.com/cyclops4100006/sdm-as-a-service/-/blob/master/sdm\\_as\\_a\\_service/](https://gitlab.com/cyclops4100006/sdm-as-a-service/-/blob/master/sdm_as_a_service/)

**Installation instructions:** [https://gitlab.com/cyclops4100006/sdm-as-a-service/-/blob/master/sdm\\_as\\_a\\_service/README.md?ref\\_type=heads](https://gitlab.com/cyclops4100006/sdm-as-a-service/-/blob/master/sdm_as_a_service/README.md?ref_type=heads)

**List of dependencies and third-party software Included in the current release**

These python packages listed in the file requirements. ([https://github.com/smart-data-models/data-models/blob/master/sdm\\_as\\_a\\_service/requirements](https://github.com/smart-data-models/data-models/blob/master/sdm_as_a_service/requirements)). Recommended to use python 3.11

- i) Faker==37.4.0
- ii) Levenshtein==0.27.1
- iii) annotated-types==0.7.0
- iv) anyio==4.9.0
- v) charset\_normalizer==3.4.2
- vi) fastapi==0.115.13
- vii) fuzzywuzzy==0.18.0
- viii) gunicorn==23.0.0
- ix) h11==0.16.0
- x) httpcore==1.0.9
- xi) httpx==0.28.1
- xii) jsonref==1.1.0
- xiii) jsonschema==4.24.0
- xiv) jsonschema-specifications==2025.4.1
- xv) pydantic==2.11.7
- xvi) pydantic-core==2.33.2
- xvii) pysmartdatamodels==0.8.0.1.1
- xviii) python-Levenshtein==0.27.1
- xix) pytz==2025.2
- xx) rapidfuzz==3.13.0
- xxi) referencing==0.36.2
- xxii) requests==2.32.4
- xxiii) rpds-py==0.25.1
- xxiv) rstr==3.2.2

xxv) ruamel.yaml==0.18.14  
xxvi) ruamel.yaml.clib==0.2.12  
xxvii) starlette==0.46.2  
xxviii) typing-inspection==0.4.1  
xxix) tzdata==2025.2  
xxx) urllib3==2.5.0  
xxxii) uvicorn==0.34.3  
xxxii) validator-collection==1.5.0

#### A.4.6. Data Exchange Broker

**Description:** Context Broker implementing the NGSI-LD Standard

**Version:** 5.0.9

**Link to repository:** <https://gitlab.com/cyclops4100006/scorpiobroker>

**Installation Instructions:**

[https://gitlab.com/cyclops4100006/scorpiobroker/-/blob/main/docker-compose.yaml?ref\\_type=heads](https://gitlab.com/cyclops4100006/scorpiobroker/-/blob/main/docker-compose.yaml?ref_type=heads)

**List of dependencies and third-party software Included in the current release**

- i) Ubuntu 24.04
- ii) Docker, version 28.0
- iii) Docker Compose version v2.33.1
- iv) [Additional libraries](#)