D3.2 Urban Freight Data Space Connector Store

Manuel Reis Monteiro, Akkodis Germany Solutions

30/04/2025

Funded by
the European Union

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101103954. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

Table of Contents

Copyright © 2023 by DISCO

Abstract

The Urban Freight Data Space (UFDS) seeks to transform urban logistics by overcoming the industry's fragmented data-sharing practices. This deliverable describes the implementation phase of the Urban Freight Data Space (UFDS), where initial objectives defined in D3.1 were revisited and refined in light of practical development insights. A Minimum Viable UFDS has been implemented based on this –realigned- architecture; a listing of the necessary components is presented with a detailed description and is made available in the Open Software Repository.[1]

This document is the second in a series of three deliverables—D3.1, D3.2, and D3.4—charting the development of the UFDS. While D3.1 laid the architectural foundation, D3.2 focuses on implementation. The final deliverable, D3.4, will complete the development cycle, providing comprehensive documentation and a finalized solution.

[1] https://gitlab.com/disco-horizon-europe/open-software-repository

Copyright © 2023 by DISCO

Summary sheet

Deliverable No.	D1.10
Project Acronym	DISCO
Full Title	DATA-DRIVEN, INTEGRATED, SYNCHROMODAL, COLLABORATIVE AND OPTIMISED URBAN FREIGHT META MODEL FOR A NEW GENERATION OF URBAN LOGISTICS AND PLANNING WITH DATA SHARING AT EUROPEAN LIVING LABS
Grant Agreement No.	101103954
Responsible Author(s)	Manuel Reis Monteiro (AKKA)
Peer Review	INLE, IMEC
Quality Assurance Committee Review	FIT
Date	30/04/2025
Status	Final
Dissemination level	Public
Work Package No.	3
Work Package Title	The Urban Freight Data Space for NetZero cities
Programme	Horizon Europe Innovation Actions
Coordinator	FIT CONSULTING SRL
Website	https://discoprojecteu.com/
Starting date	01/05/2023
Number of months	42 months

Project partners

Copyright © 2023 by DISCO

Organisation	Country	Abbreviation
FIT CONSULTING SRL IT Coordinator	IT	FIT
RUPPRECHT CONSULT-FORSCHUNG & BERATUNG GMBH	DE	RC
INLECOM INNOVATION ASTIKI MI KERDOSKOPIKI ETAIREIA	EL	INLE
PNO INNOVATION SL	ES	PNO
INTERNATIONAL DATA SPACES EV	DE	IDSA
FM LOGISTIC IBERICA SL	ES	FM
Akkodis Germany Solutions GmbH	DE	AKKA
FONDAZIONE ISTITUTO SUI TRASPORTI E LA LOGISTICA	IT	ITL
JLL	UK	JLL
ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTY XIS	EL	CERTH
LINDHOLMEN SCIENCE PARK AB	SE	LSP
KUHNE LOGISTICS UNIVERSITY GGMBH	DE	KLU
INSTITUT DE RECHERCHE TECHNOLOGIQUE SYSTEM X	FR	IRTX
STICHTING BREDA UNIVERSITY OF APPLIED SCIENCES	NL	BUAS
POLIS - PROMOTION OF OPERATIONAL LINKS WITH INTEGRATED SERVICES, ASSOCIATION INTERNATIONALE	BE	POLIS
EUROPEAN PARKING ASSOCIATION EPA EV	DE	EPA
ALLIANCE FOR LOGISTICS INNOVATION THROUGH COLLABORATION IN EUROPE	BE	ALICE
ERASMUS CENTRE FOR URBAN, PORT AND TRANSPORT ECONOMICS BV	NL	ERASMUS
INTERUNIVERSITAIR MICRO-ELECTRONICA CENTRUM	BE	IMEC

Copyright © 2023 by DISCO

VLAAMS INSTITUUT VOOR DE LOGISTIEK VZW	BE	VIL
FUNDACION ZARAGOZA LOGISTICS CENTER	ES	ZLC
BE-MOBILE	BE	BE-MOBILE
STAD GENT	BE	GENT
OPLEIDINGSCENTRUM VOOR HOUT EN BOUW VZW	BE	OHB
CITYLOGIN IBERICA SL	ES	CITYLOGIN
UNIVERSITAT POLITECNICA DE CATALUNYA	ES	UPC
AJUNTAMENT DE BARCELONA	ES	BCN
VENICE INTERNATIONAL UNIVERSITY	IT	VIU
FUNDACION DE LA COMUNIDAD VALENCIANA PARA LA INVESTIGACION, PROMOCION Y ESTUDIOS COMERCIALES DE VALENCIAPORT	ES	VPF
FUNDACION DE LA COMUNITAT VALENCIANA PARA LA PROMOCION ESTRATEGICA EL DESARROLLO Y LA INNOVACION URBANA	ES	LAS NAVES
T-BOX DELIVERY & SOLUTIONS SL	ES	T-BOX
AYUNTAMIENTO DE ZARAGOZA	ES	ZARAGOZA
FUNDACION ZARAGOZA CIUDAD DE CONOCIMIENTO	ES	FZCC
FORUM VIRIUM HELSINKI OY	FI	FVH
KOBENHAVNS KOMMUNE	DK	COPENHAGEN
REGION HOVEDSTADEN DK Partner	DK	REGIONH
COMUNE DI PIACENZA	IT	PIACENZA
MESTSKA CAST PRAHA 6 / District Prague	CZ	PRAHA
REGIONAL MANAGEMENT NORDHESSEN GMBH	DE	RMNH
AARHUS KOMMUNE	DK	AAKS

Copyright © 2023 by DISCO

DIMOS THESSALONIKIS	EL	THESSALONIKI
DIETHNIS EKTHESI THESSALONIKI AE	EL	TIF HELEXPO
ACS TACHIDROMIKES IPIRESIES MONOPROSOPI ANONYM	EL	ACS
ROLAN OY	FI	ROLAN
ASOCIACIÓN LOGÍSTICA INNOVADORA DE ARAGÓN	ES	ALIA
A to B Finland Oy	FI	A2B
GETPLUS srl IT Partner	IT	NEXT
COMUNE DI PADOVA IT	IT	ComPADUA

Copyright © 2023 by DISCO

Document history

Version	Date	Organisation	Main area of changes
0.1	20/03/2025	AKKA	Definition of ToC
0.2	01/04/2025	INLE	Input provided
0.3	09/04/2025	IMEC, IRTX, AKKA	Inputs provided.
0.4	14/04/2025	AKKA	1st Draft
0.5	18/04/2025	AKKA	2nd Draft
1.0	23/04/2025	IMEC	Final Edits
1.1	28/04/2025	AKKA	Addressing comments
1.2	30/04/2025		Final Release

Copyright © 2023 by DISCO

List of acronyms

D	Deliverable
UFDS	Urban Freight Data Space
FOSS	Free and Open Source Software
UI	User Interface
FAIR	Findability, Accessibility, Interoperability, and Reusability
LL	Living Lab
UF	Urban Freight
WP	Work Package

Copyright © 2023 by DISCO

1. Introduction

This document provides a comprehensive overview of the solution architecture adopted by DISCO for the Urban Freight Data Space (UFDS) as part of the WP3 effort and building upon the high-level architecture initially highlighted in D3.1. The architecture is designed to deliver a secured data space platform by adhering to three fundamental design principles: prioritizing the use of existing and open-source technologies, enabling user interaction through an intuitive user interface (UI), and utilizing mature, stable technologies. These core principles are discussed in greater detail in D3.1 - DISCO Data Space Open Software Repository Version 1, Section 2.2.

Section 2 provides a recapitulation of the Design Principles announced in D3.1; based on these principles, a set of choices was selected and implemented.

Section 3 focusses on the architecture of the overall dataspace. Firstly Section 3.1 reiterates the high-level architecture as per D3.1 original definition and highlights the modifications that were required to improve UFDS functionality for the project; this represents an additional five components. Next, Section 3.2 provides, in layman's terms, a description of the UFDS solution architecture by using consumer/producer participant roles to describe the purpose and function of each component integrated into the solution; Appendix 1 enhances this section by providing a detailed description of the participant on-boarding process; Appendix 2 provides a detailed description of the data communication flow between UFDS Connectors.
Section 4 details all components necessary within a UFDS environment; starting with UFDS Dataspace main component (Section 4.1) describes in its sub-sections all extensions and sub-components to Sovity Dataspace framework; Section 4.2 describes the Logging House app which is responsible to log an audit trail of transactions occurring between connectors; Section 4.3 depicts the Vocabulary Hub service, which used to maintain semantic interoperability among organisations and systems; Section 4.4 describes the Trajectory Data Anonymization, a pipeline to help visualize and quantify logistics activities across urban environments; finally Section 4.5 provides an overview of the Smart Data Platform used to create, manage and execute data transformation pipelines.

Section 5 provides a short overview of the deployment capabilities offered as this stage; finalised deployment setups will be delivered in D3.4 DISCO Data Space open software repository Ver 2, & Implementation Guide.

Section 6 documents DISCOEYE, a proof-of-concept that highlights how a dataspace can be leveraged to develop end-user applications. DISCOEYE demonstrates the ability of data usage from different living labs and consume data into a map-oriented visual webapp that consolidates all relevant user data. DISCOEYE is constituted of three different app components: UFDS PoC Control App (Section 6.1), which helps configure the ingestion of data; UFDS PoC Adapter App (Section 6.2), which transform input data assets to GeoJSON data for visualization; finally the UFDS PoC Visualiser App (Section 6.3) provides the rendering of GeoJSON datasets to the user.

Copyright © 2023 by DISCO

Finally, Section 7 summarizes the deliverable.

Copyright © 2023 by DISCO

2. Design Principles

Since the inception of the UFDS, we established three key design principles: (i) prioritizing existing and open-source technology, (ii) enabling user interaction through a UI, and (iii) utilizing mature, "general release" technologies. These foundational principles are outlined in detail in D3.1 - DISCO Data Space Open Software Repository Version 1, Section 2.2.

As a result of these principles, two significant decisions were made: first, to utilize Sovity's "data space package" due to its near "out-of-the-box" functionality compared to other alternatives; and second, to open-source any components developed during the project to ensure broader access and reusability. This inclusive approach is intended to foster collaboration and innovation within the community.

The consequence of these choices resulted in: (i) The UFDS components being available as open-source releases in the DISCO GitLab repository; our partners may decide to open- or close-source their applications that are built on top of the dataspace. (ii) A UI is provided based on the existing UI available in the Sovity Dataspace framework. (iii) Finally, the adoption of best-in-class components such as Keycloak, Apache APISIX, Dagster and further open-source frameworks described in section 4 helped create a Minimum Viable UFDS, which was demonstrated with the DISCOEYE application described in section 6.

Copyright © 2023 by DISCO

3. Architecture Overview

This section highlights the original high-level architecture as described in D3.1 - DISCO Data Space Open Software Repository Version 1, emphasising the necessary changes required to implement the UFDS (Section 3.1). Based on this update an architecture is proposed that describes a cohesive environment to enable a UFDS (Section 3.2). The details of each component and applications are detailed in the following Section 4.

3.1. High Level Architecture

The first version of UFDS's high-level architecture was presented in D3.1 - DISCO Data Space Open Software Repository Version 1, Section 2.2.1. Specifically, we outlined the anticipated high-level architecture for UFDS's final version (Q4 2025), as illustrated in Figure 1. While this architecture has been largely followed, some minor deviations have occurred, which we discuss below. Additionally, an updated version of the high-level architecture is illustrated in Figure 2.

Metadata Broker / Federated Catalog

As initially reported in D3.1, we planned to extend EDC's framework and develop this component using Datahub, an open-source metadata platform that helps organizations discover, manage, and govern their data assets through automated lineage, search, and collaboration[2]. However, following the decision to adopt Sovity's Connector components, we determined that Sovity's own Broker implementation (the Broker Server) was better suited for our needs. Moreover, this choice reduced the development effort towards this component, allowing us to reallocate the extra resources toward enhancing other data space components. This component is discussed in Section 4.1.4.

Vocabulary Hub

Initially, and as reported in D3.1, we planned to use TNO's Semantic Treehouse[3]; however, this presented several challenges. Firstly, the software is not directly deployable and requires some preparatory steps, increasing the effort needed for deployment. More importantly however, adapting it to our needs, ensuring seamless integration with the Meta Model Suite, App and

[2] https://datahubproject.io/
[3] https://www.semantic-treehouse.nl/

Copyright © 2023 by DISCO

Connector Store, Open Software Repository, and DISCOLLECTION from a user's perspective, is difficult without deep familiarity with its codebase.

For this reason, we are shifting toward developing a new component from the ground up. This approach will not only integrate smoothly with the other components of the DISCO project as envisioned but will also allow users to compare samples of their datasets with the vocabulary used in UFDS in a clear and intuitive manner. We further discuss the revised component in Section 4.3.

Logging House

In the initial high-level architecture reported in D3.1, we did not include a Logging House. However, during implementation, it became evident that a central logging facility would enable transparency, accountability, and auditability throughout the data space by creating a structured and secure way of logging transactions between connectors. This new component is explained in detail in Section 4.2.

Dataspace manager

A new addition to the UFDS architecture is the Dataspace manager; its main purpose is to simplify the onboarding of new participants. This is done by supporting the exchange and installation of certificates from the participants involved in a UFDS. This new component is explained in more details in section 4.1.7.

DISCOLLECTION / Smart Data Platform

This service constitutes an essential component to support the transformation of datasets into streamlined anonymous data consumable across a dataspace. The Smart Data Platform provides an extract-transform-load (ETL) pipeline that simplifies data transformation for the participants.
The processed data can be then listed and shared in the dataspace federated catalogues to authorized participants.

Copyright © 2023 by DISCO

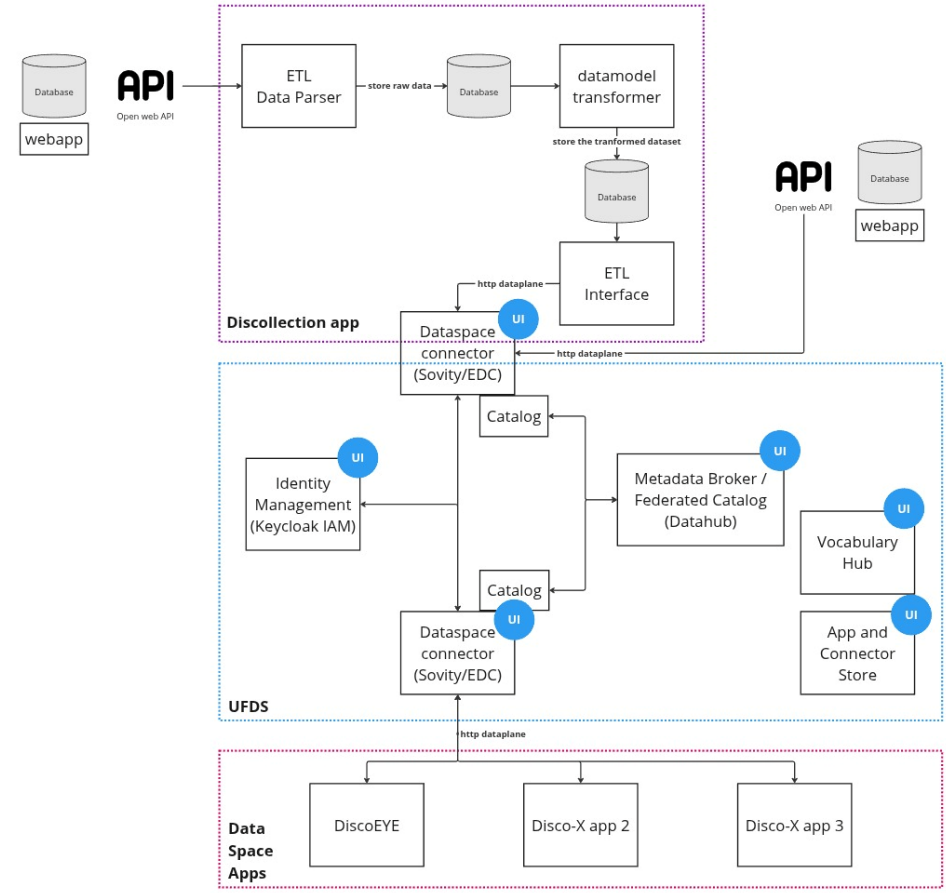

Figure 1: Anticipated high-level architecture reported in D3.1

Copyright © 2023 by DISCO

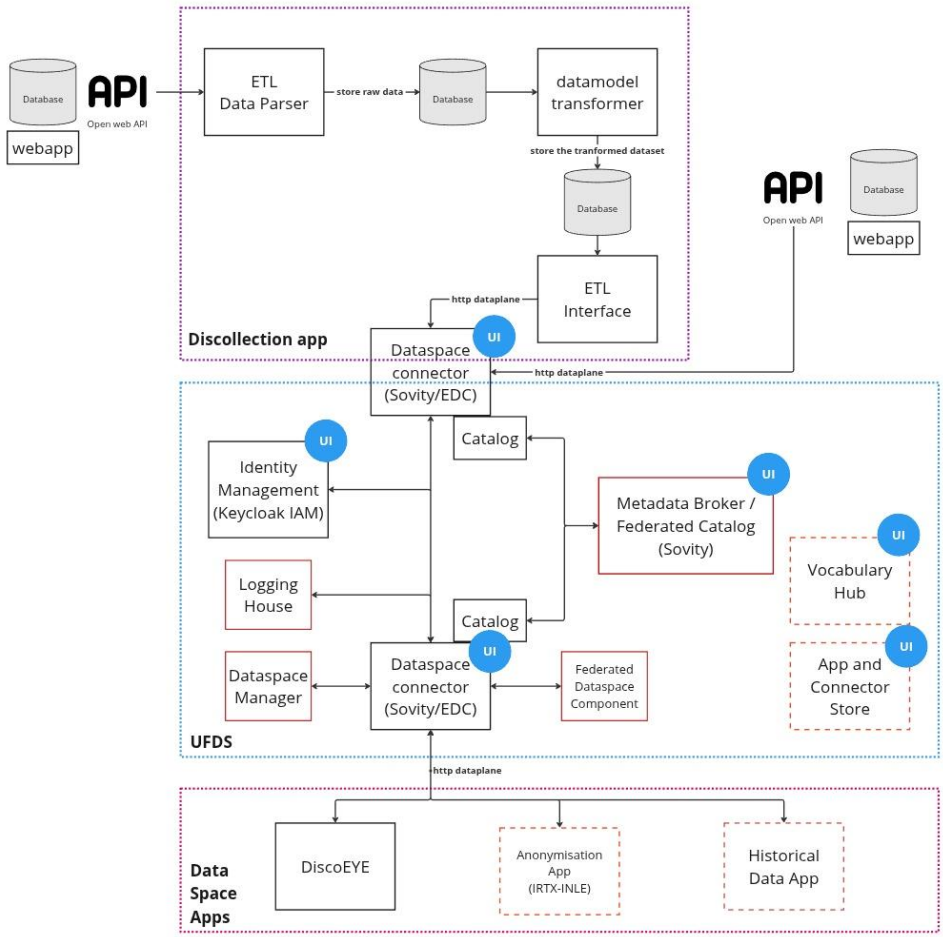

Figure 2: Current high-level architecture

3.2. Proposed Solution architecture

Copyright © 2023 by DISCO

The proposed technical solution layout is provided in Figure 3 in terms of components and systems, which depicts the solution as participants assuming consumer or producer roles within a UFDS environment. Participants can have both roles if required.

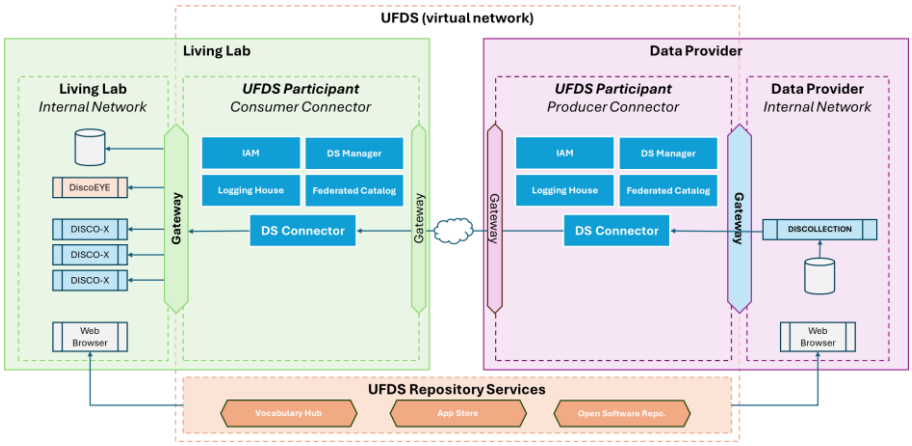

Figure 3: Solution Architecture

The Dataspace manager's purpose is to initialise participants into a UFDS; it simplifies the setup of certificates from each participant to enable the Identity Access Module (IAM). The purpose of the IAM is authenticate and authorize data requests and secure communication across dataspace connectors.

Each Participant maintains its own UFDS connector that needs to be associated into a "*virtual network*"; this is done by registering each participant connector with one another following the on-boarding process described in Appendix 1: Onboarding Process in Dataspace Manager. Each UFDS Connector should possess an API Gateway facing the public internet; this public gateway is connected to the Logging House component service to ensure that ingress and egress data traffic between connectors can be logged. Logging occurs on each DS Connector endpoint.

A UFDS Data Provider feeds data from its internal or private network through the DISCOLLECTION service that transforms internal data in a form consumable by the data consumer; the transformed data is then provided to the DS Connector and shared with the Consumer connector as configured between the participants. The main vehicle to advertise datasets is the Federated Catalogue that enables the visibility of datasets across participants in the UFDS across to the defined access policies agreed between participants.

Copyright © 2023 by DISCO

A UFDS Data Consumer consumes the data and redispatches it according to internal applications or databases on the Consumer side. The communication mechanisms are detailed in Appendix 2: Communication and data exchange flow in a participation Data Space. To facilitate the integration of DS connectors with internal applications (in our case DISCO-X services) an internal API gateway can be leveraged to provide access to DS Connector in a familiar way to software developers developing the DISCO-X services (for example providing REST Apis to the DISCO-X services so that DS Connector endpoints should not be directly integrated in the applications).

While DS Connectors are decentralised by nature, UFDS Repository services are required, as a centralised service, to share across the UFDS Participants, the applications, data models, and app store necessary to design and code data transformation pipelines as well as a prompt installation and setup of a dataspace for future Participants. The UFDS Repository service also provides documentation and guidelines to support UFDS adopters in understanding how to obtain and deploy software, structure of the datasets, etc. This documentation is planned to be improved and be included or referred to in Deliverable D3.4.

Copyright © 2023 by DISCO

4. Components

This section provides a detailed description of all the components within the UFDS environment. Section 4.1 describes all extensions and additional components necessary to create and execute a UFDS environment. UFDS relies on the framework provided by Sovity; this includes the connector, IAM, Oauth2-proxy, Metadata Broker, and UI components. The framework was also extended to accommodate data pulling, which is not available on the standard framework. Additionally, the Dataspace manager is described in detail.
Sections 4.2 and 4.3 describe the Logging House and the Vocabulary Hub components, while sections 4.4 and 4.5 describe applications, namely the Trajectory Data Anonymization and the Smart Data Platform (aka DISCOLLECTION).

4.1. UFDS Data Space Main Components

Purpose:
Provides the foundation for the UFDS data space and provides base configurations and extensions to facilitate UFDS creation. This component is based on the Sovity data space framework which provides the central parts of the UFDS data space.

Description:
Through robust security measures and adherence to established protocols, the Sovity EDC CE Stack facilitates the secure exchange of data assets, enhancing interoperability and maintaining efficient communication channels, thereby bridging various data assets and partners seamlessly, enabling streamlined data exchange. It bridges various data assets and partners seamlessly, enabling streamlined data exchange.

Repository:
https://github.com/sovity

License:
Apache License, V2.0

Supplier:
Sovity GmbH (https://sovity.de)

4.1.1. Sovity Connectors

Purpose:
Communication setup and data transfer within a data space

Description:
This component provides the following Connector capabilities:

Copyright © 2023 by DISCO

- **Provider Connector**: Define and publish data assets in data catalogues which could be provided with limitation policies.
- **Consumer Connector**: Search for data assets in released data sets, negotiate a contract and start a data transfer.

Data exchange process between a Consumer and a Provider within a Dataspace

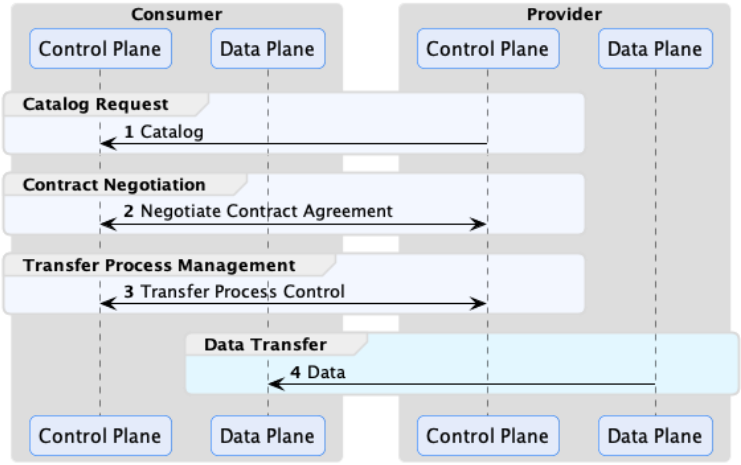

Figure 4: Data exchange process between Consumer and Provider.

Figure 4 illustrates a **data exchange process** between a **Consumer** and a **Provider** within a Data Space The interaction is structured around a **Control Plane** and a **Data Plane** for both entities.

- The **Control Plane** is responsible for **negotiation, management, and control** of the interaction.
- The **Data Plane** is responsible for the **actual transfer of data** once agreements are established.

The process consists of four distinct stages:

1. **Catalog Request**

- **Purpose:** The Consumer initiates a request to retrieve the available catalog from the Provider.
- **Flow:**
  - The request originates from the **Control Plane** of the Consumer and is sent to the **Control Plane** of the Provider.
  - Labeled as: **"1 Catalog"**.
  - Represents **discovery and resource availability negotiation**.

Copyright © 2023 by DISCO

2. **Contract Negotiation**

- **Purpose:** Both parties negotiate and establish a contractual agreement governing the data exchange.
- **Flow:**
  - Initiated by the Consumer's **Control Plane** and processed by the Provider's **Control Plane**.
  - Labeled as: **"2 Negotiate Contract Agreement"**.

3. **Transfer Process Management**

- **Purpose:** To establish control mechanisms for actual data transfer.
- **Flow:**
  - Triggered by the Consumer's **Control Plane** and managed by the Provider's **Control Plane**.
  - Labeled as: **"3 Transfer Process Control"**.
  - Ensures **pre-transfer validations and preparations** are in place before data exchange.

4. Data Transfer

- **Purpose:** To execute the actual transfer of data between the Consumer and Provider.
- **Flow:**
  - Occurs directly between the **Data Planes** of both entities.
  - Labeled as: **"4 Data"**.
  - Represents the **actual payload transfer**, where the negotiated terms are enforced.

Repository:
https://github.com/sovity/edc-ce

License:
Apache License, V2.0

Supplier:
Sovity GmbH (https://sovity.de)

4.1.2. Connector Extension for Data Pull Support

Purpose:
Extension of *Sovity EDC CE to support Data Pull*

Description:

Copyright © 2023 by DISCO

This component extends the Sovity connector by adding the Eclipse Data Space Components Extensions, which provides additional communication patterns. The resulting integration enables the HTTP-Proxy / Data Pull communication pattern within the UFDS environment.
Namely, a UFDS consumer requests data to a UFDS provider, which fetches the requested data and responds synchronously to the UFDS consumer.

Errore. L'origine riferimento non è stata trovata. illustrates the data exchange process within the **UFDS architecture**, highlighting interactions between multiple components through **HTTP Data** and **HTTP Proxy** communication channels. The flow is divided into **two categories**: **HTTP Data (Red)** depicts Direct data transfer per push, and **HTTP Proxy (Blue)** depicts the Proxy-based data requests per pull.

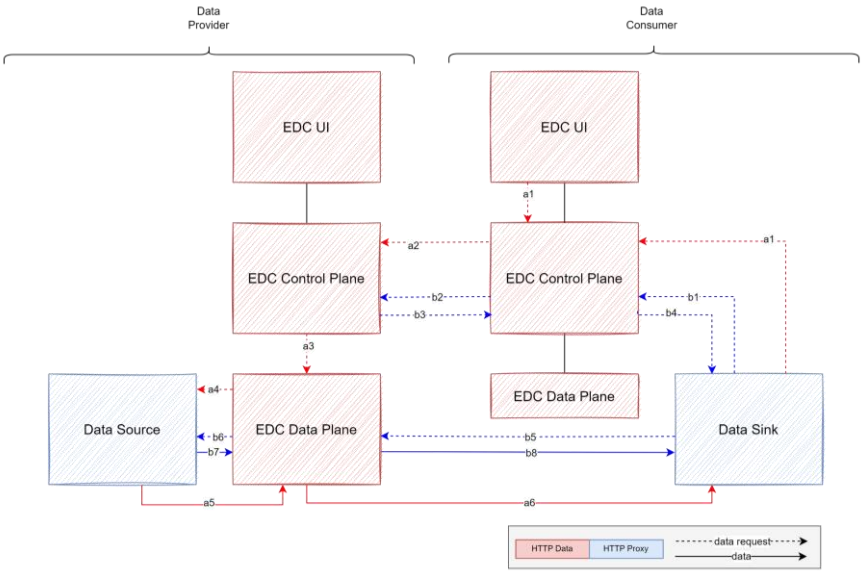

Figure 5: Data Transfer Data flow.

Note that in this diagram, EDC depicts the sublayers of the UFDS.

Flow Description

Data Request Flow	
a1-a3	The Consumer's EDC Control Plane sends a request to the Provider's EDC Control Plane via HTTP Data with a reference to its Data Sink (solid red line).

Copyright © 2023 by DISCO

a4	The Provider's EDC Data Plane resolves the requested Data Source by Consumer's EDC Control Plane
a5-a6	The Provider's EDC Data Plane sends the requested data directly to the Consumer's Data Sink via HTTP Data (solid red line).

Proxy-based Data Request Flow	
b1-b2	The Consumer's EDC Control Plane initiates a proxy request to the Provider's EDC Control Plane via HTTP Proxy with a reference to its Data Sink (dashed blue line).
b3-b4	The Provider's EDC Control Plane pushes the proxy data to Consumer's Data Sink for processing.
b5	The Consumer's EDC Client requests the HTTP Proxy Data (dashed blue line).
b6-b7	The Provider's EDC Control Plane resolves the requested data source and gives it back to Consumer's EDC Client.
b8	The Provider's EDC Client completes the data exchange process via HTTP Proxy (dashed blue line).

Repository:
https://github.com/sovity/edc-ce
https://github.com/eclipse-edc/Connector
Company specific repository: IMEC and AKKA

License:
Apache License, V2.0

Supplier:
Sovity GmbH (https://sovity.de)
Eclipse Dataspace Components (https://github.com/eclipse-edc)
AKKA

ha formattato: Italiano (Italia)
ha formattato: Italiano (Italia)
ha formattato: Italiano (Italia)
Codice campo modificato

4.1.3. IAM (DAPS / Keycloak)

Purpose:
Establishing trusted connections for every connector-to-connector interaction and authentication and securing the Connector-UI via OAuth2.

Copyright © 2023 by DISCO

Description:
The IAM groups two deferent components that are instrumental in asserting Identity of a participants and performing authentication and authorization of participants.
The OAuth2 Proxy component ensures that only authorized users or systems of a participant can access the UFDS Connector and acts as the authentication middleware between the Connector UI and the UFDS Connector.
The DAPS (Dynamic Attribute Provisioning Service) support authentication and secure connector access between participants and generates identity tokens based on OAuth2 as well as certificates for inter-connector communication.

Participant OAuth2 Proxy
The OAuth2 Proxy acts as an authentication intermediary between the user and the Connector-UI. When a user attempts to access the Connector -UI, the proxy first checks whether the user is authenticated. It does this by verifying if a valid authentication token is present, usually provided by Keycloak, the identity and access management system in use.

- **Authentication Check:** The proxy inspects the incoming request to determine if the user has an active session or a valid authentication token. If the token is missing, expired, or invalid, the proxy identifies the user as unauthenticated.
- **Redirection for Authentication:** If the user is not authenticated, the proxy automatically redirects the user to the Keycloak login page. This is done via an HTTP redirect to Keycloak's authentication endpoint.
- **Keycloak Authentication:** Keycloak serves as the central authentication service that handles the user login process. Once the user successfully logs in, Keycloak issues an authentication token (typically a JWT token) that confirms the user's identity. After authentication, Keycloak redirects the user back to the proxy, which in turn grants them access to the requested resource — in this case, the Connector-UI.

Seamless Access Control: After the authentication process, the proxy continues to check for token validity on subsequent requests to ensure the user maintains valid access rights throughout their session. Any future access requests to EDC-UI will be granted based on the validity of the authentication token without requiring the user to re-authenticate unless the session expires.

Dynamic Attribute Provisioning Service (DAPS)
DAPS provides the Dynamic Attribute Provisioning Service based on the Keycloak IAM platform. For this purpose, an authentication takes place using a private public key between the connector and DAPS. The connector-specific data is recorded as DAT (Dynamic Attribute Token) in signed JWT token claims for validation by the counterparty.

DAPS also offers the possibility of OAuth2 Authentication with the corresponding client to secure the Connector-UI via oauth2-proxy.

Copyright © 2023 by DISCO

For further details on the purpose and technical specification of this component please see D3.1, DISCO Data Space open software repository Version 1, Section 2.3.3.

Repository:
https://github.com/sovity/sovity-daps
https://gitlab.com/disco-horizon-europe/open-software-repository/ufds-daps

License:
Apache License, V2.0

Supplier:
Sovity GmbH (https://sovity.de)
AKKA
INLE

4.1.4. Metadata Broker / Federated Catalog

Purpose:
Aggregate and index connectors and their data offerings using the IDS protocol.

Description:
Built upon the Eclipse Dataspace Components (EDC) framework, Sovity's Broker Server extends EDC's capabilities to provide enterprise-ready managed services. It is designed to operate in tandem with Sovity's Broker UI, a different "flavour" of Sovity EDC UI, facilitating efficient management and oversight of data exchange processes within a dataspace. Although we also host this component in our Open Software Repository, we have made no changes to it and are using it as-is.

Repository:
https://github.com/sovity/edc-broker-server-extension/tree/v4.2.0
https://gitlab.com/disco-horizon-europe/open-software-repository/ufds-metadata-broker

License:
Apache License, V2.0

Supplier:
Sovity GmbH (https://sovity.de)

4.1.5. Sovity EDC UI

Purpose:
Dataspace Administration user interface.

Description:

Copyright © 2023 by DISCO

The **UFDS Connector-UI** provides a comprehensive and user-friendly interface tailored for **Connector Administrators** and is designed to offer efficient management and oversight of various key aspects of the **UFDS Connectors**, enabling administrators to maintain control and monitor the data exchange process effectively.

Key Features:

- **Asset Management:**
  - Administrators can view and manage all registered assets within the dataspace.
  - The UI provides detailed information on asset properties, metadata, and status.
  - It supports adding, updating, and removing assets as required.
- **Policy Management:**
  - The UI displays a list of data usage policies associated with the assets.
  - Administrators can define, modify, and assign policies to regulate data access and transfer.
  - It ensures compliance with data governance standards and contract terms.
- **Catalog Overview:**
  - The Connector-UI offers an organized view of the **data catalog**, listing available resources and datasets.
  - It supports **catalog search** functionality, allowing admins to quickly find specific assets or data services.
  - The catalog also shows metadata and access rights, facilitating efficient data discovery.
- **Contract Management:**
  - Administrators can initiate **contract negotiations** through the UI.
  - The platform supports the creation, editing, and monitoring of contract agreements between data consumers and providers.
  - The contract lifecycle, from initiation to finalization, is tracked directly within the interface.
- **Transfer Management:**
  - The UI allows the **initialization of data transfers** after successful contract negotiations.
  - Transfer status, logs, and progress are available in real-time, offering transparency and traceability.
  - Completed transfers are archived for auditing and compliance purposes.

Repository:
https://github.com/sovity/edc-ui
https://gitlab.com/disco-horizon-europe/open-software-repository/connector-ui

License:
Apache License, V2.0

Supplier:
Sovity GmbH (https://sovity.de)

Copyright © 2023 by DISCO

INLE

4.1.6. OAuth2-proxy

Purpose:
Secure the access to EDC UI via OAuth2

Description:
The Sovity OAuth2 Proxy (which is merely a fork of the original OAuth2Proxy) acts as an authentication intermediary between the user and the EDC-UI. When a user attempts to access the EDC-UI, the proxy first checks whether the user is already authenticated. It does this by verifying if a valid authentication token is present, usually provided by Keycloak, the identity and access management system in use.

- **Authentication Check:** The proxy inspects the incoming request to determine if the user has an active session or a valid authentication token. If the token is missing, expired, or invalid, the proxy identifies the user as unauthenticated.
- **Redirection for Authentication:** If the user is not authenticated, the proxy automatically redirects the user to the Keycloak login page. This is done via an HTTP redirect to Keycloak's authentication endpoint.
- **Keycloak Authentication:** Keycloak serves as the central authentication service that handles the user login process. Once the user successfully logs in, Keycloak issues an authentication token (typically a JWT token) that confirms the user's identity. After authentication, Keycloak redirects the user back to the proxy, which in turn grants them access to the requested resource — in this case, the EDC-UI.
- **Seamless Access Control:** After the authentication process, the proxy continues to check for token validity on subsequent requests to ensure the user maintains valid access rights throughout their session. Any future access requests to EDC-UI will be granted based on the validity of the authentication token without requiring the user to re-authenticate unless the session expires.

Repository:
https://github.com/oauth2-proxy/oauth2-proxy

License:	***Supplier***:
MIT	AKKA

4.1.7. Data Space Manager

Purpose:
Onboarding and management of connectors in UFDS data space

Copyright © 2023 by DISCO

Description:
The **Dataspace Manager** is a critical component designed to streamline the management and onboarding process of connectors and their associated certificates within a data space ecosystem. It offers a suite of resources that ensure seamless integration, validation, and continuous management of connectors, making it easier to connect, secure, and scale data-sharing operations within the data space.

Key Features and Functionalities:

- **Connector Management:**
  - The Dataspace Manager provides robust resources for managing connectors that enable communication and data exchange between systems or organizations within a data space.
  - It facilitates the registration, configuration, and management of connectors, ensuring they adhere to the necessary security and compatibility standards required for integration.
  - Connectors can be associated with different dataspaces allowing for flexible and scalable connectivity options.
- **Certificate Management:**
  - Security is a key concern in any data-sharing network, and the Dataspace Manager offers resources to manage the certificates necessary for authenticating and securing data exchanges.
  - It manages the onboarding of certificates tied to connectors, ensuring that each connector can authenticate itself securely within the dataspace.
- **Onboarding Process:**
  - The onboarding process is integral to ensuring that all connectors and certificates are correctly integrated into the dataspace. The Dataspace Manager provides automated and streamlined resources for the onboarding of new connectors.
  - During the onboarding process, connectors are validated to confirm that they meet the necessary criteria for integration, such as certificate validity and metadata standards.
  - The system also ensures that the corresponding certificates are valid and properly associated with each connector, facilitating secure and trusted communication within the data space.
- **Validation Resources:**
  - **Connector Validation:** Dataspace Manager validates each connector's capabilities, checking its security measures (including certificates), and compliance with the data space's governance policies. Validation ensures connectors operate as expected and are valid.
- **Catalog Crawling:**

Copyright © 2023 by DISCO

- As part of the onboarding process, the Dataspace Manager provides resources for **catalog crawling registration**, which involves automatically scanning the data space for available datasets, services, and connectors.

- **Seamless Integration and Discovery:**
  - The combination of connector validation and catalog crawling allows the Dataspace Manager to offer a seamless integration experience. After successful onboarding, connectors are cataloged and can be easily discovered and accessed by authorized users or systems within the dataspace ecosystem.
  - This ensures that new connectors are on-boarded and validated and are automatically made available for further interaction and integration with other services or data sources in the data space.
- **Governance and Compliance:**
  - The Dataspace Manager ensures that all connectors and certificates comply with the data space's governance and data-sharing policies.

Repository:
https://gitlab.com/disco-horizon-europe/open-software-repository/dataspace-manager

License:	***Supplier***:
Apache 2.0	AKKA

4.1.8. Federated Data Space Component

Purpose:
Allow establishing trusted connections between connectors using different DAPS services.

Description:
An extension to the Sovity Connector facilitating identity decentralisation by allowing connectors to function on a dataspace architecture consisting of multiple DAPS. Enables the connection of different dataspaces and the creation of a dataspace federation without compromising on the data sovereignty, a central pillar of the data space infrastructure.

A federation plugin component is required to be deployed for every connector wishing to participate in the federated data space. Moreover, an API Gateway (Apache APISIX, in our case) is required for request interception and forwarding to the federation plugin.

The token verification process, necessary for connector-to-connector trusted connections is slightly modified. First, each connector sets a “primary” DAPS provider, that it requests JWT tokens from, prior to communicating with a different connector. Then:

Copyright © 2023 by DISCO

1. Each connector instead of requesting the DAPS's public keys from their primary DAPS, it does so from the federation plugin, which has been preloaded with the public keys of all DAPS in the federation. This allows for the connector to trust certificates issued by multiple DAPS providers in addition to their primary.
2. Each request sent to the connector (by some other connector on the federation) is intercepted by the API Gateway and sent to the federating plugin for JWT token verification. This verification is more elaborate than the standard verification performed by Sovity's Connector, and –loosely speaking- requires the client-id to follow the naming pattern `<DAPS signing the token ID>:<Connector ID>` instead of the standard `<Connector ID>`. Essentially, this guarantees that a malicious DAPS on the federated data space, will only be able to falsify tokens impersonating connectors having that DAPS as primary.

We illustrate the modified connector operation procedure in Figure 6.

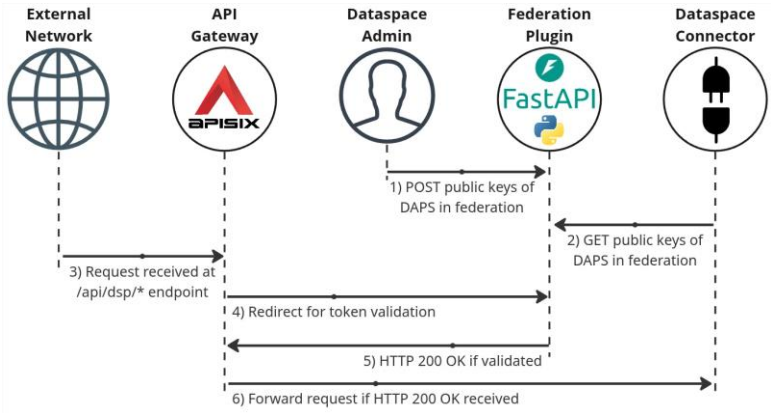

Figure 6: Connector operation procedure.

Repository:
https://gitlab.com/disco-horizon-europe/open-software-repository/ufds-daps-keys-manager
https://gitlab.com/disco-horizon-europe/open-software-repository/ufds-api-gateway

License:	***Supplier***:
Apache License, V2.0	INLE

Copyright © 2023 by DISCO

4.2. Logging House

Purpose:
A trusted entity that records and stores data transactions for the purposes of transparency, accountability, and compliance. It provides an unalterable audit record of who shared or accessed data and when. This allows organizations to meet regulatory requirements (e.g., GDPR), settle disputes, and uphold data usage policies.

Description:
The Logging House implementation consists of two main components: the Logging House and the Logging House Client. The Logging House is deployed once per data space, typically under the authority of the Data Space Authority, as is the case with identity management infrastructure like Keycloak DAPS. One Logging House Client is required for every connector making use of the Logging House and is deployed alongside it. This client was written to avoid the need to modify the Data Space Connector codebase, expediting development. That said, in a more production-quality implementation, this logic would be integrated directly into the connector itself to avoid the need for an external client. Similarly, an API Gateway (Apache APISIX, in our case) is required for request interception and log forwarding, though this component would also not be needed if logging were handled by the connector itself.

Logging of requests follows the sequence diagram in Figure 7, which ensures that requestor and responder identities in each log entry are authenticated with JWT tokens issued by the DAPS. For the purposes of DISCO, we assume that log authenticity relies on the Data Space Authority, but in the future, a blockchain-based shared ledger between the participants would be capable of enhancing security and transparency. Also, data space members can view only logs in which they are explicitly involved as requestors or responders, while administrators (in our case, the Data Space Authority) can view all logs in the Logging House.

Copyright © 2023 by DISCO

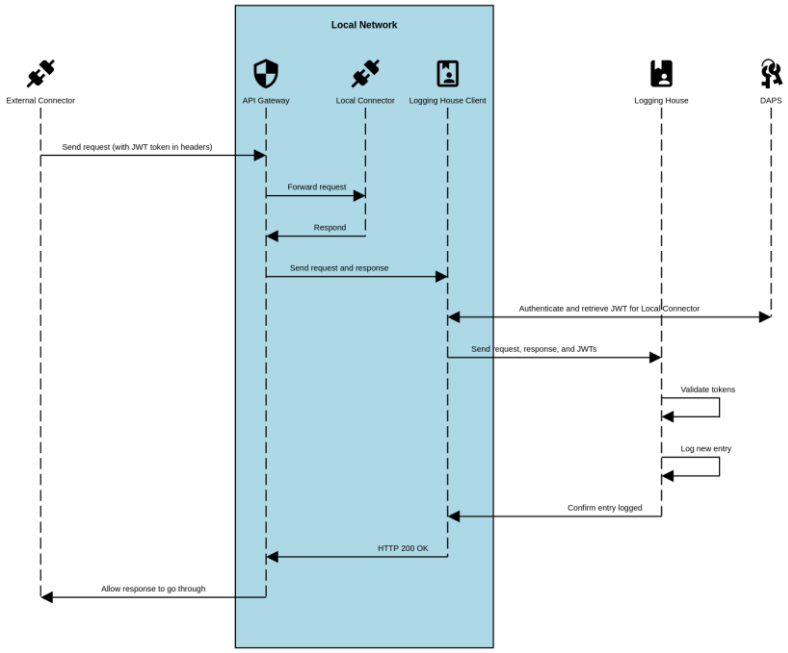

Figure 7: Logging House Sequence Diagram

Repository:
https://gitlab.com/disco-horizon-europe/open-software-repository/ufds-clearing-house

License:
Apache License, V2.0 (to be provided)

Supplier:
INLE

4.3. Vocabulary Hub

Purpose:
A Vocabulary Hub serves as a common repository for the coordination of domain vocabularies, ontologies, and taxonomies. It facilitates semantic interoperability enabling data exchange among organizations and systems. Furthermore, it enhances data consistency, enabling effective collaboration in the data space.

Copyright © 2023 by DISCO

***Description**:*
The Vocabulary Hub will be an independent component of the data space, which can be accessed by any participant after they have authenticated to use it. It is being developed using MkDocs, a static site generator designed specifically to construct lightweight, simple-to-use websites and will provide semantic explanations for all the various data models used by various applications in the data space, allowing participants to comprehend and coordinate their data structures in the best possible manner.

Besides documentation, the Vocabulary Hub will also enable users to compare their own sample data against existing data models, highlighting structural and semantic differences using an inlined JSON comparison tool (specifically, an independent deployment of jsondiff.com, which visually compares JSON objects). This functionality is designed to integrate seamlessly with DISCO's Meta Model Suite, Open Software Repository, App and Connector Store, and DISCOLLECTION, offering a single solution for data semantics management. Development is still ongoing with initial feasibility tests having confirmed that the envisioned functionality can be implemented; a working version is expected to take shape over the next couple of months. The final version of the application will be reported in D3.4 - DISCO Data Space open software repository Ver 2, and Implementation Guide (M36).

Repository: (currently private)
https://gitlab.com/disco-horizon-europe/open-software-repository/ufds-vocabulary-hub

License:	***Supplier***:
Apache License, V2.0 (to be confirmed)	INLE

4.4. Trajectory Data Anonymization (Copenhagen Digital Twin)

Purpose:
This pipeline helps visualize and quantify logistics activities across urban environments by transforming raw GPS trajectory data into meaningful spatial analytics.
It processes OpenStreetMap data, study area boundaries, and logistics operator trajectory data to analyze delivery patterns. It maps vehicle trajectories to the road network, determines probable routes via shortest-path calculations, and generates two key outputs:

(1) a hexagon grid analysis of average daily stops by operators, and
(2) a road segment analysis showing average daily vehicle counts throughout the network.

Description:
The repository provides a pipeline that takes the following inputs:

- An OpenStreetMap cut-out of a region (here Denmark)
- The shape of a study area (here Copenhagen)
- Trajectory data by logistics operators

The trajectory data per operator is structured as a CSV file containing at least the following columns: `vehicle_id`, `date`, `longitude`, `latitude`.
For each operator, we, hence, need to know the (ordered) stop sequence of one or multiple vehicles over one or multiple days. Each day per operator should have a unique identifier.
The pipeline will then perform the following processing steps:

- A road network, based on the study area shape, will be extracted from OpenSteetMap and prepared for routing.
- The trajectories and stops of all operators will be mapped to the road network of the study area.
- The trajectories of all operators will be routed by the shortest path to identify the probable routes that the vehicles may have taken in reality.

Finally, the pipeline aggregates the obtained information. The following outputs are generated:

- A geographic analysis file that counts the number of performed stops by the operators for an average day on a hexagon grid (size is configurable).
- A geographic analysis file that counts the number of passing vehicles on each road segment in the road network on an average day.

Input data
The minimal input of the pipeline is the following: download an OpenStreetMap snapshot of Denmark from `Geofabrik` and place the file `denmark-latest.osm.pbf` into resources.
A shape file of the Copenhagen area is already provided in `resources/study_area.gpkg`. Note that the paths to those files can be configured in `resources/config.yml`.
If specific (real) operator data is available, each operator should obtain a folder with its name in `resources/operators/{name}` and the operator should be added to the list of operators inside `resources/config.yml`.
By default, two synthetic operators, `synthetic1` and `synthetic2` are automatically generated based on some high-level statistic information, an operating area and depot locations defined in `resources/synthetic.gpkg`.

Copyright © 2023 by DISCO

Output data

After running the pipeline, the aggregation files are updated, either analysing the two synthetic operators or any additional operator that has been added in `resources/operators/{name}` and which has been added to the operators list in `resources/config.yml`.

The geographic data can be visualized, for instance using QGIS. In case you want to use QGIS, a readily prepared project file that accesses `results/aggregation/stops.gpkg` and `results/aggregation/flows.gpkg` is available in `workflow/Mapping.qgz`.

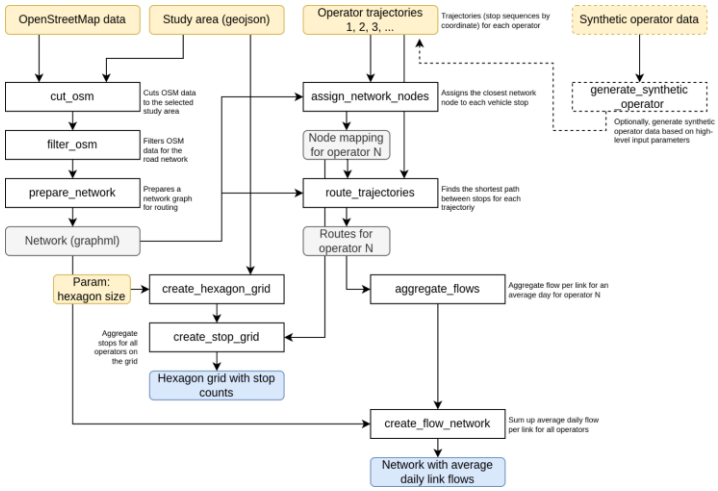

Figure 8: Digital twin flowchart.

Repository:
https://github.com/IRT-SystemX/DISCO

License:
MIT-0 license

Supplier:
IRTX

4.5. Smart Data Platform

Purpose:
A data transformation tool that supports cities in preparing their data and offering it to the data space. The integrated UFDS connector reduces the technical expertise required to connect and offer their data to the data space.

Description:

Copyright © 2023 by DISCO

The Smart Data Platform (SDP) combines an open-source ETL pipeline orchestrator (Dagster) with an Urban Freight Data Space (UFDS) connector, integrated through a REST middleware. This architecture significantly improves the efficiency with which cities can connect their data to the data space.

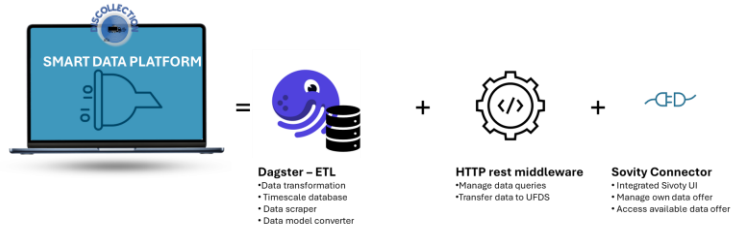

Figure 9: The Smart Data Platform components

Dagster is a cloud-native data pipeline orchestrator that manages the entire development lifecycle, featuring integrated data lineage visualization, comprehensive observability, a declarative programming model, and good testability.

Designed specifically for developing and maintaining data assets, such including tables, datasets, machine learning models, and reports, Dagster allows you to declare your desired data assets as Python functions. The system then intelligently schedules these functions and maintains your assets in an up-to-date state.

Dagster orchestrator is deployed alongside the Sovity UFDS connector, with communication through an HTTP REST middleware. To streamline operations between these components, the Smart Data Platform features an integrated user interface that automates commands and provides unified control over both the Dagster orchestrator and the Sovity connector.

More information on the Smart Data Platform will be included in Deliverable 2.5, DISCO-X & Meta Model Suite Interoperability (M30). For this reason, and since the components is still under development, it is not yet available to on the Open Software Repository.

Copyright © 2023 by DISCO

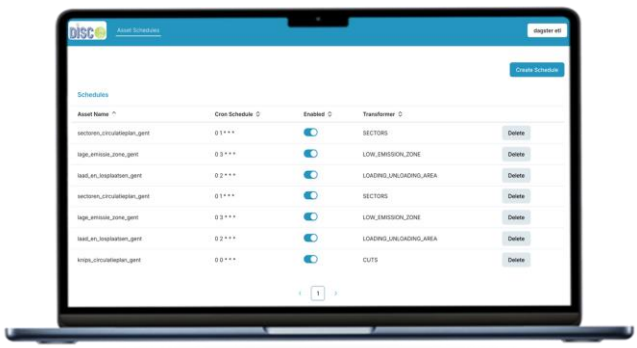

Figure 10: The Smart Data Platform has a user-friendly UI to manage data transfer between the ETL pipeline and Sovity connector

Repository:
Dagster:
https://github.com/dagster-io/dagster

License:
Apache v2.0

Supplier:
Dagster: Elementl, Inc. d.b.a. Dagster Labs.
SDP UI: IMEC

Copyright © 2023 by DISCO

5. Deployment

The current deployments amongst the WP3 partners encompass Docker Compose setups as well as Kubernetes deployments. All Docker files and Helm charts are under development but currently available (in a development state) in the DISCO Gitlab repository. They will be finalised and reported in Deliverable D3.4 DISCO Data Space open software repository Ver 2, & Implementation Guide.

Copyright © 2023 by DISCO

6. Proof-of-Concept: DISCOEYE

During DISCO's General Assembly in Cologne (October 2024), a proof-of-concept (PoC) demonstration of the Urban Freight Data Space (UFDS) was shown. The UFDS network consisted of four connectors, one for each of Thessaloniki, Copenhagen, and Ghent living labs, which were data producers, each providing datasets to the data space. The fourth connector was a data consumer, which utilized these datasets.

In addition, this data consumer used three data space applications: the Control Application, the Adapter Application, and the Geodata Visualizer Application. Together, as described in the following subsections 6.1, 6.2, and 6.3. These applications backed the PoC use case for DISCO, DISCOEYE.

6.1. UFDS PoC Control App

The Control Application was developed with Python and FastAPI. It provides a simple REST API interface so that the users can set up one-off or scheduled consumption of particular data assets their connector is capable of accessing in the data space. Users may choose to specify the "type" of the consumed dataset, and the Adapter Application will subsequently transform it into Visualizer Application-compliant form.

This capability came in handy while processing real-time parking data from the Copenhagen and Ghent living labs. By allowing for regular data downloading—up to every minute—it kept the Visualizer Application continuously updated with close-to-real-time data.

The application is available at our Open Software Repository:
https://gitlab.com/disco-horizon-europe/open-software-repository/control-app

6.2. UFDS PoC Adapter App

Similar to the Control Application, the Adapter Application was developed using Python and FastAPI with a MongoDB deployment for its storage needs. Its functionality is to transform input data assets, each with a specific format, into a standard GeoJSON format. The transformed data assets were then saved in user-defined "folders" (MongoDB collections), with a standard format for consumption by the Control Application. As the data assets contained geo-located data, such standardization allowed for effortless integration with the Visualizer Application for data visualization. Towards this purpose, we developed a "data reshape" task per data asset type, also

Copyright © 2023 by DISCO

understanding that future scalability required us to construct the codebase in a way that would easily accommodate future addition of data reshaping tasks.

It must be mentioned that the basic functionality of this application is like that of DISCOLLECTION (see Section 4.5), but with two important differences. First, unlike DISCOLLECTION, this application is a proof of concept (PoC) and has not been fully tested. Second, while DISCOLLECTION performs data transformation on the Data Producer's side, normalizing data assets before sharing on UFDS, the Adapter Application performs this transformation on the Data Consumer's side, i.e., the data assets are shared on and consumed from UFDS in their raw state, before being converted.

The application is available at our Open Software Repository:
https://gitlab.com/disco-horizon-europe/open-software-repository/adapter-app

6.3. UFDS PoC Visualiser App

The application has two main components: the backend and the frontend, which is typical architecture for UI-based systems.

The backend was developed using Python and FastAPI and interacts with MongoDB to retrieve and supply data to the frontend. It provides:

- a list of the available GeoJSON datasets and allows users to select which dataset they want to visualize,
- the data of an individually selected dataset for visualization and,
- a URL to a public icon that the frontend uses on visualization, for each unique type of data (e.g., car parks, delivery points).

The frontend is built using React and TypeScript. It incorporates OpenStreetMap and Mapbox to display an interactive world map to present various GeoJSON datasets. It supports a range of GeoJSON types, including Points (e.g., the position of a delivery box), Lines (e.g., road traffic paths), and Polygons (e.g., parking spaces).

The backend component is available at our Open Software Repository: https://gitlab.com/disco-horizon-europe/open-software-repository/visualiser-app-backend. The frontend component is currently closed-source and not public but its codebase is available upon request.

In Figure 11, Figure 12, and Figure 13 we present the usage of the tool when displaying different datasets at 3 of the Starring Living Labs (Copenhagen, Ghent, Thessaloniki).

Copyright © 2023 by DISCO

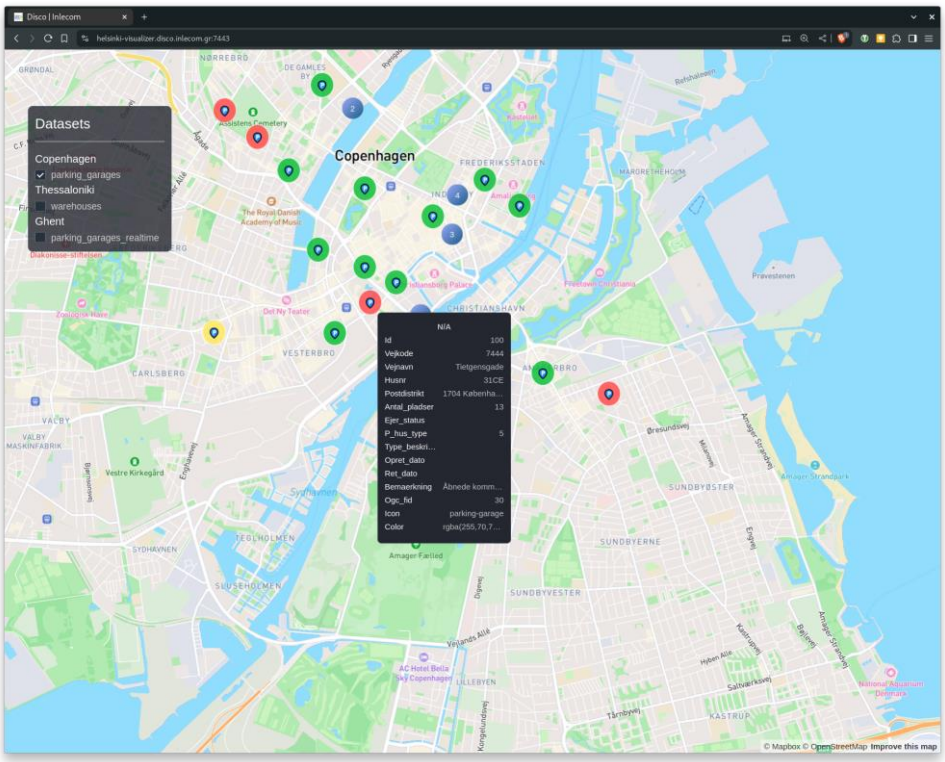

Figure 11: UFDS Visualiser APP (Copenhagen)

Copyright © 2023 by DISCO

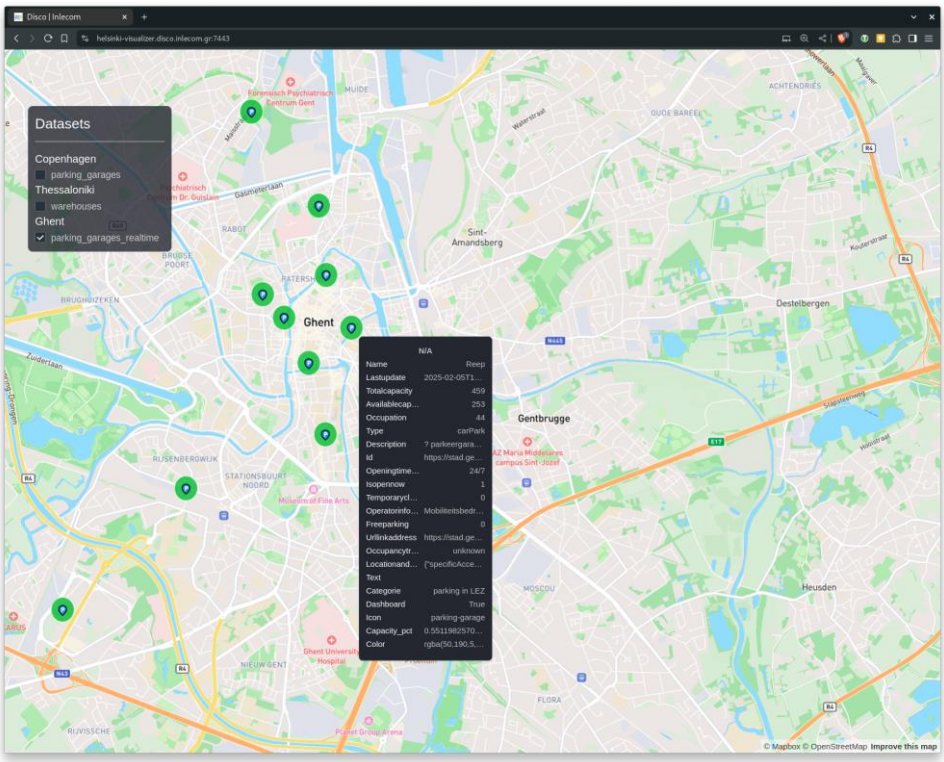

Figure 12: UFDS Visualiser APP (Ghent)

Copyright © 2023 by DISCO

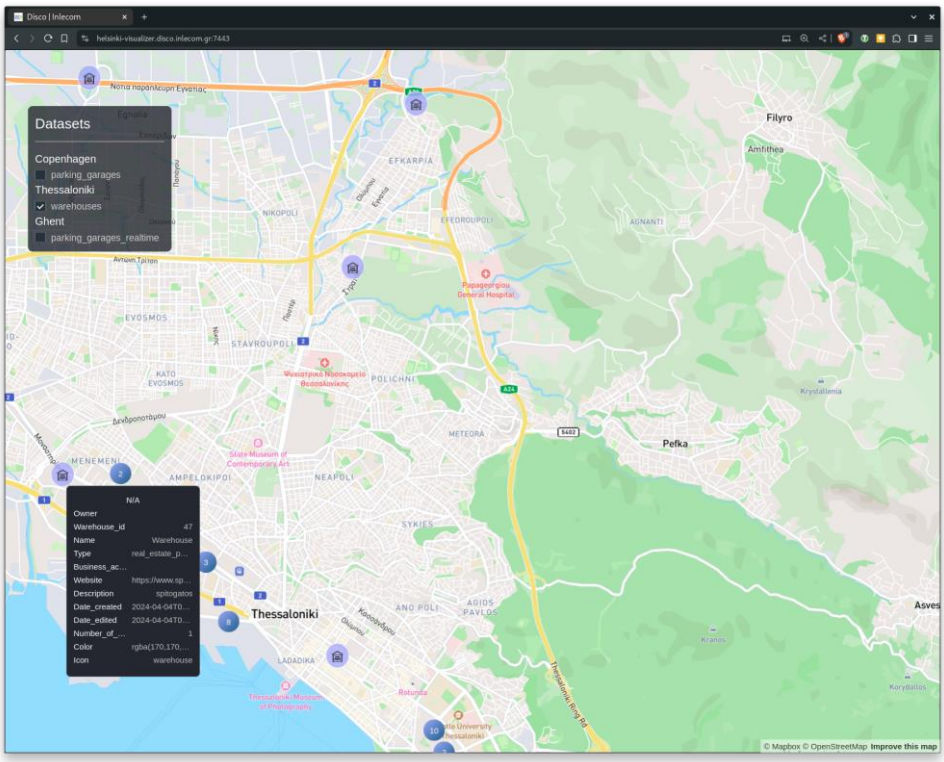

Figure 13: UFDS Visualiser APP (Thessaloniki)

Copyright © 2023 by DISCO

7. Conclusion

This document highlights the iterative developments that have taken place in achieving the layout and implementation of UFDS dataspace architecture. As highlighted in D3.1, the solution has been evaluated in our first milestone (October 2024) and has progressed to enable a secured environment for the second milestone planned for Q4 2025. Additional components have been added to the ecosystem to ensure auditing of data transactions and facilitating the onboarding of additional participants in a UFDS dataspace. The third and final -technical- deliverable of WP3, D3.4, will conclude the development process by consolidating the system, providing full documentation, and validating the final implementation.

Copyright © 2023 by DISCO

Appendix 1: Onboarding Process in Dataspace Manager

The onboarding process refers to all the steps that are necessary for setting up a Dataspace Connector or its participants or accepting new participants.

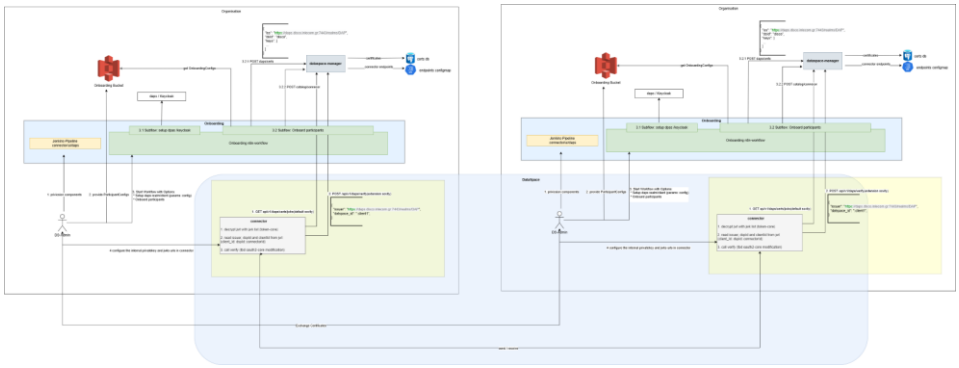

Figure 14: The Onboarding Workflow.

The following components must be created and deployed for each organization:

- **Connector** As a participant, the communication establishes, receives and sends the data.
- **Jenkins Pipeline** – Automates provisioning the components.
- **Onboarding Bucket** – Stores onboarding configuration.
- **DAPS / Keycloak** – Handles security, identity, and access management.
- **Dataspace Manager** – Central service to register and manage connector configurations.

Technical Steps

The process is split / executed into two parallel organisation sides. Each side follows a similar process for onboarding connectors into a Dataspace environment in following steps:

1. **Provision Components**
   - Provisions components like connectors, DAPS and dataspace-manager.

2. **Provide Participant Configuration**

Copyright © 2023 by DISCO

- Admin provides and exchange configurations for each participant, which includes details needed for onboarding and setting up secure communications.

3. **Start Onboarding Workflow**
   - **Setup DAPS / Keycloak**
     The DAPS/Keycloak instance will be configured.

   - **Onboard Participants**
     Uses the provided / persisted configuration from the Onboarding Bucket as well as dataspace-manager to orchestrate the creation of DAPS certs and registration of connector endpoints for catalog crawling using dataspace-manager.

Optional Verification Flow

The optimal verification flow provides an extra ability to verify the certificates and clients due to their dataspace and organization relationship via dataspace-manager and needs an extra customizing of Sovity connector.

Copyright © 2023 by DISCO

Appendix 2: Communication and data exchange flow in a participation Data Space

A **Dataspace** serves as a secure and trusted environment for seamless data exchange between authorized participants. Within this ecosystem, each participant interacts with the data space using their own **connector**, which plays a central role in either providing or consuming data.

- **Provider Connector**: A provider connector is responsible for publishing data that it wishes to share with other participants. It registers the data in its **data catalog** and may apply policies that restrict access based on the recipient's identity or other criteria.

- **Consumer Connector**: A consumer connector, on the other hand, requests and receives data from provider connectors based on the available catalogs. These roles of "*provider*" and "*consumer*" can be dynamically assigned depending on the context of the interaction.

The communication between these connectors takes place via their **endpoints**. These endpoints are secured using **HTTPS** and **JWT**, which are used for authentication and authorization purposes.

Copyright © 2023 by DISCO

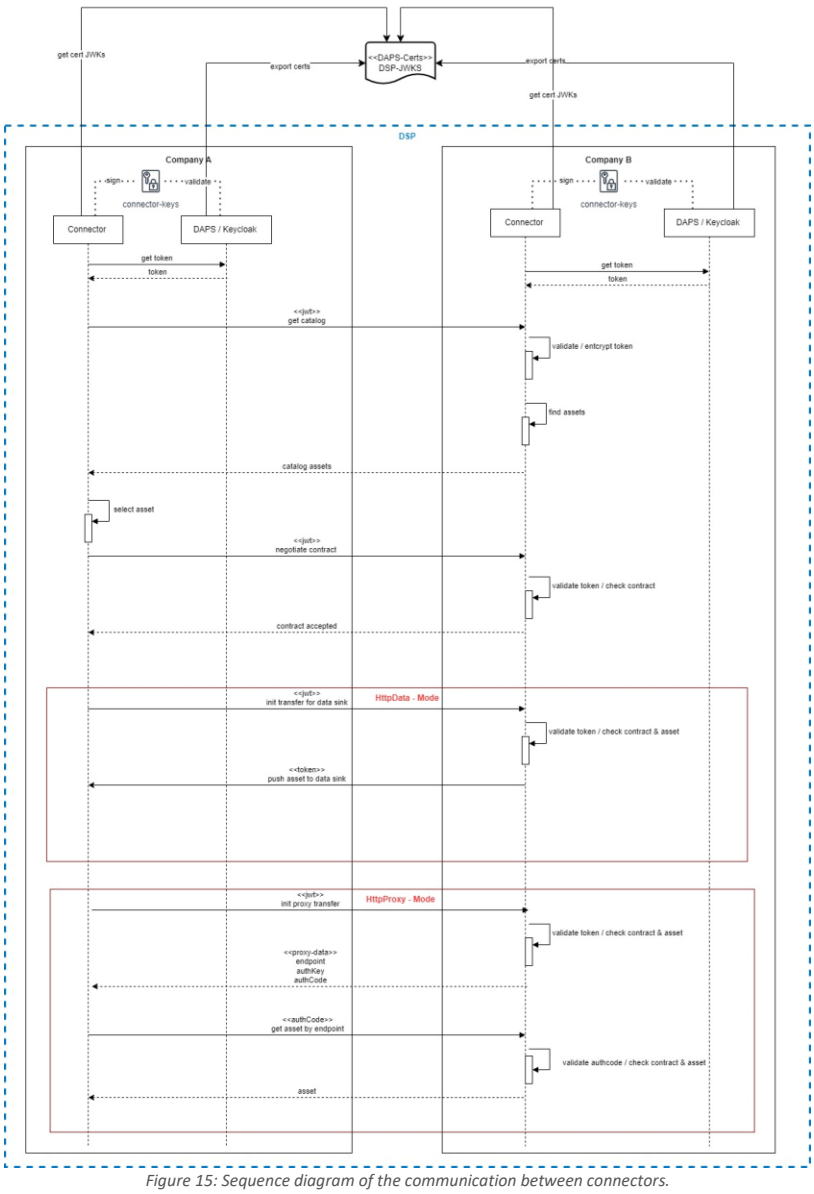

Figure 15: Sequence diagram of the communication between connectors.

Copyright © 2023 by DISCO

Authentication and Token Validation Mechanism

- **Decentralized Authentication**: Authentication within the Dataspace is decentralized. This means each connector has its own **Keycloak instance** and **JWT signing mechanism**. Each connector authenticates itself using its own **private/public key pair** for signing JWT tokens, ensuring that each connector's identity is verifiable.

- **JWT Tokens**: Once a connector authenticates itself with its Keycloak instance, it receives a **JWT token**. This token is used to authenticate the connector in all subsequent interactions with other connectors in the data space. The token is included in the HTTP header of each request and is validated by the receiving connector using the corresponding public key.

- **Certificate Sharing**: To ensure that JWT tokens can be validated between connectors, **Keycloak certificates** are shared among all participants during the onboarding process. This allows the participant connectors to verify the authenticity of JWT tokens issued by other participants' Keycloak instances.

Data Request/Receipt Process

The process of requesting and receiving data within the Dataspace follows a well-defined series of steps:

1. **Authentication**:
    - Each connector authenticates itself with its **local Keycloak** instance and obtains a **JWT token**.

2. **Request for Data Catalog**:
    - The **consumer connector** sends a request to the **provider connector** to retrieve its **data catalog**. This catalog lists the data assets that are available for sharing.

3. **Token Validation and Access Control**:
    - Upon receiving the consumer's request, the **provider connector** validates the **JWT token** of the consumer. It checks if the consumer has the necessary permissions to access the data assets listed in the catalog, based on the **policies** defined by the provider.
    - The provider then delivers the data catalog, which contains the list of data assets available for the consumer, with appropriate policies.

4. **Contract Negotiation**:
    - After receiving the data catalog, the consumer selects the data assets they are interested in and begins a **contract negotiation** with the provider.

    - The contract negotiation includes validating the data transfer terms, such as the data format, transformation options, and other relevant conditions.

Copyright © 2023 by DISCO

- The provider validates the consumer's token and confirms the contract by agreeing to the terms specified by the consumer.

Data Transfer Modes
Once the contract is established, the actual data transfer can happen in different modes, depending on the needs of the consumer and the provider:

- **HTTP Data Transfer**:
  - In this mode, the consumer requests the transformation of the selected data asset by providing a **data sink URL**, which specifies where the data should be transferred.
  - The **provider** validates the consumer's token and contract agreement, then pushes the content of the data asset directly to the **data sink URL** specified by the consumer.
- **HTTP Proxy Data Transfer**:
  - Instead of directly transferring the data, in this mode, the provider responds with **proxy data**. This includes a **recipient URL** that the consumer can use for data transfer, along with **authKey** and **authCode**, which are used to authenticate and authorize the data transfer process.
  - The **consumer** uses the proxy data to request the transformation and transfer of the data asset. It sends a request to the provided recipient URL and includes the **authKey** and **authCode** for validation.
  - The **provider** validates the consumer's token and contract, then responds with the requested data asset, ensuring that the transfer is secure and that all access controls are enforced.

Security and Trust Model
The Dataspace operates on a foundation of strong **security** and **trust** principles:

- **JWT Authentication**: Each connector's identity is verified using JWT tokens, ensuring that only authenticated and authorized participants can request or provide data.
- **TLS Encryption**: All data exchange is secured using HTTPS (TLS), ensuring that data in transit is encrypted and protected from eavesdropping or tampering.
- **Certificate Sharing**: The sharing of **Keycloak certificates** between participants during onboarding ensures mutual trust and allows connectors to validate each other's JWT tokens effectively.

Copyright © 2023 by DISCO