MOBILITY DATA SPACE

Corso di: Big Data Architectures

Dott. Marco Fanfani

DISIT Lab, Sistemi Distribuiti e Tecnologie Internet

http://www.disit.dinfo.unifi.it/

Dipartimento di Ingegneria dell'Informazione

Università degli Studi di Firenze

CC BY SA

The Need for Data Spaces

- Mobility data, in particular real-time data, are not easy to acquire, due to **different data provides**, that can use **different distribution modalities.**
- Data provides are sometimes also **difficult to find** requiring time-consuming research activities.
- There is a strong need to **increase data discoverability** among different provides.
- Not only to find a dataset, but also to establish relationships of **trust**, **conditions of use** and **interoperable ways of exchange** between organizations.

Mobility Data Landscape: A Fragmented Ecosystem

Mobility data are distributed across a large number of independent data ecosystems.

- Public authorities, transport operators, infrastructure managers, service providers and private companies collect and manage different types of mobility data.
- Data are published through different platforms, portals, APIs and access mechanisms.
- Data formats, metadata, standards, access conditions and update frequencies may differ significantly.
- Existing initiatives often cover specific geographical areas, transport modes or application domains.

National Access Points and NAPCORE

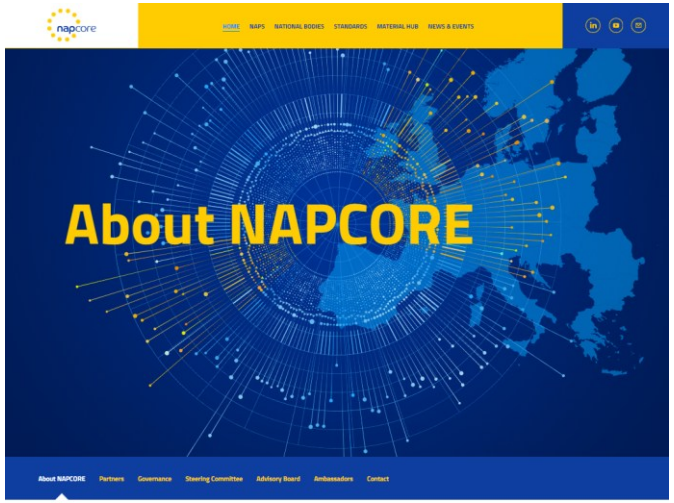
National Access Points (NAPs) provide national entry points for discovering and accessing transport-related data.

- Established by EU Member States under the Intelligent Transport Systems (ITS) framework.
- Provide a **single point of access**, at national level, to relevant static, historical and dynamic transport data covered by the applicable regulations.
- Support data discovery and reuse through metadata, standards and interfaces.
- Connect data holders with organisations developing mobility and transport services.

NAPCORE – Coordinating the European NAP Network

- A European initiative to coordinate and harmonise National Access Points.
- Promotes **common standards**, metadata, procedures and technical approaches.
- Improves **interoperability** and facilitates cross-border discovery and reuse of mobility data.
- Its second phase, launched in July 2025, involves 77 partners across 33 countries.

https://napcore.eu/



PrepDSpace4Mobility

https://mobilitydataspace-csa.eu/

A European preparatory action (years 2022-2023) aimed at laying the foundations for a common European Mobility Data Space.

- Mapped existing mobility and logistics data ecosystems.
- Analysed gaps, overlaps and interoperability challenges.
- Identified common technical building blocks and governance requirements.
- Proposed recommendations for data sovereignty, trust, interoperability and discoverability.

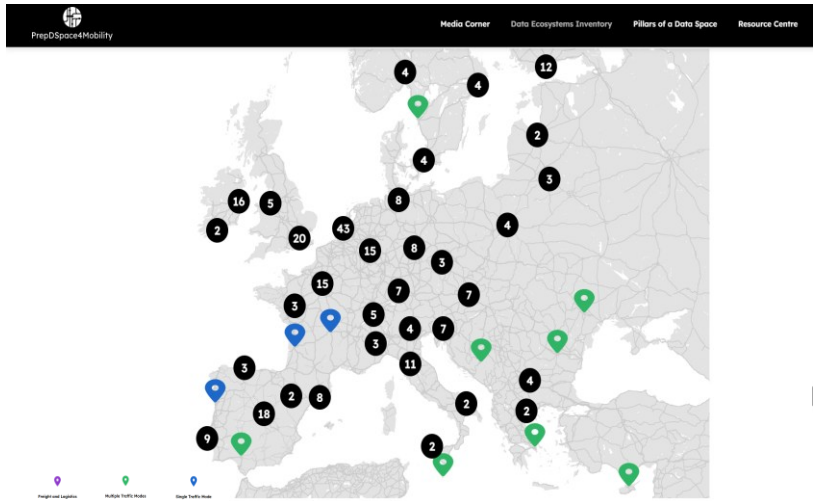
On 29 November 2023, the European Commission adopted the Communication on the creation of a common European Mobility Data Space (EMDS).

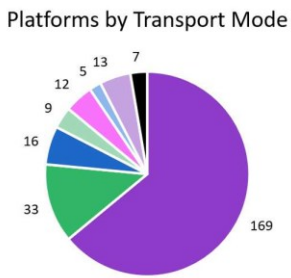


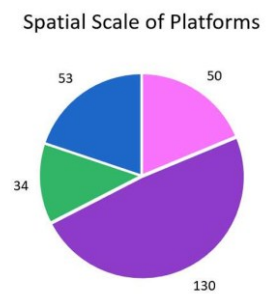
PrepDSpace4Mobility

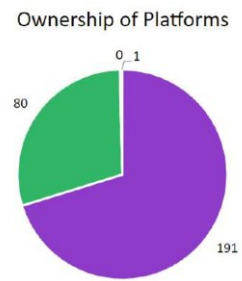
272 data ecosystems included in the inventory

- Observed over **18 sectoral mobility domains**, each with their own standards, data models and protocols
- Observed over **60 different standards**
- Even in each mobility domain, several (overlapping) standards are in use











https://mobilitydataspace-csa.eu/

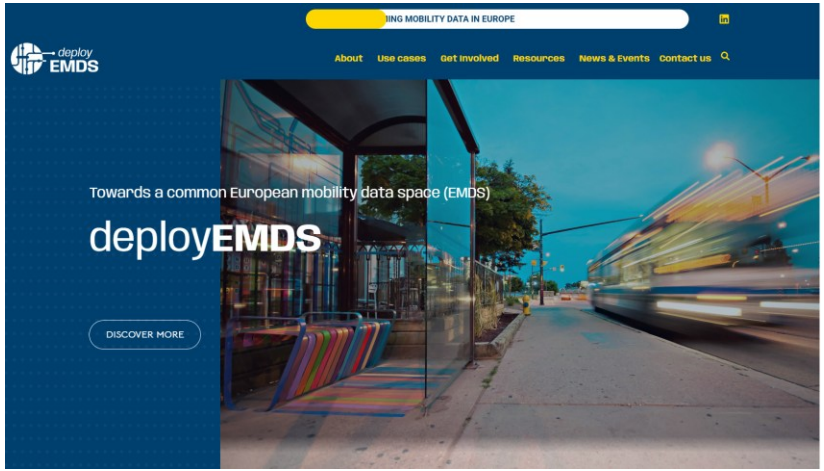
deployEMDS

deployEMDS translates the EMDS vision into **operational infrastructure** and **mobility use cases.**

A European Deployment Action

- Launched in November 2023 as part of the EU Digital Europe Programme.
- Builds on the preparatory work of PrepDSpace4Mobility.
- Involves **16 use cases across 9 European cities and regions**.
- Brings together public authorities, transport operators, mobility providers and technology organisations.

https://deployemds.eu/

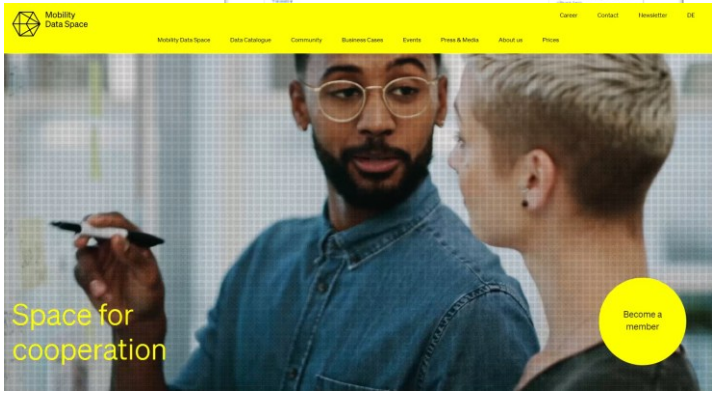


Mobility Data Space (MDS)

The most advanced project today is the German Mobility Data Space

- Brings together public and private mobility stakeholders.
- Provides a data catalogue and a portal to discover and exchange data.
- Uses EDC-based connectors to support secure and sovereign data exchange.
- Offers Connector-as-a-Service (CaaS), allowing participants to use a connector without deploying it in their own infrastructure.
- Established a full connection with Mobilithek, the German platform for exchanging digital information, in 2025.





https://mobility-dataspace.eu

Data Spaces

Traditional data management architectures have historically relied on **centralized paradigms** such as Cloud Data Warehouses and Data Lakes, which require physical data replication and rigid ETL pipelines when scaled across distinct corporate boundaries.

Data spaces introduce a **decentralized**, **sovereign** framework of autonomous participant nodes where communication shifts from merely establishing technical network links to **enforcing governance rules**—specifically defining

- who is permitted to share what data,
- under which exact terms, and
- for what purpose.

Rather than consolidating data into a single physical repository, data spaces establish a **virtualized data sharing ecosystem** where participants retain **physical control** and **legal custody** over their local databases

Data spaces strictly decouple metadata signaling and contract management (**Control Plane**) from actual payload transfer execution (**Data Plane**)

To avoid the necessity of centralized control over data schemas and storage layers, data spaces provide **semantic interoperability** over distributed heterogeneous nodes.

Data Spaces vs Traditional Systems

Aspect	Traditional systems	Data spaces
Architecture	Mostly centralized	Distributed
Data control	Limited or transferred	Sovereign
Schema	Rigid	Semantic interoperability
Governance	System-defined	Federated
Usage control	Limited, system-level	Fine-grained, policy-based
Scope	Data management	Data ecosystems
Scale	Organizational	Cross-organizational

Definition of a European Data Space

*As formalized by ISO/IEC 20151 and the International Data Spaces Association (IDSA), a Data Space is defined as an **interoperable**, **federated** infrastructure based on **shared governance** principles, common **technical standards**, and **enabling software services** that facilitate **trustworthy**, **policy-driven** data transactions between **autonomous** participant organizations*

Definition of a European Data Space

Data Space is defined as an **interoperable**, **federated** framework based on

- shared governance principles
- open technical standards
- common enabling services to facilitate trusted data transactions

Instead of static extract-transform-load (ETL) routines, data spaces bridge syntactic and semantic heterogeneity by establishing **context-aware schemas** and **Linked Data metadata** envelopes (such as ETSI NGSI-LD)

Definition of a European Data Space

Transactions within a data space are **legally and technically constrained** by

- mutually accepted rulebooks
- service level agreements (SLAs)
- machine-readable policy definitions

Bare REST endpoint are not adequate! A raw data stream (e.g., a real-time vehicle positioning feed) is packaged into a **formal Data Product,** a digital asset that couples data endpoint with

- rich semantic metadata
- unambiguous data quality attributes
- explicit Service Level Agreements (SLAs)
- machine-readable usage rights

Data Sovereignty vs. Data Ownership





Data ownership

Refers to the **legal rights and entitlements** of an **entity** over a digital asset.

- Grant usage rights
- Share and transfer data
- Define conditions for data usage

Data sovereignty

Represents the **effective technical capability** of a **data provider** to:

- Enforce data usage policies and constraints
- Retain control over how, when, by whom, and for what purposes digital assets are processed, even after sharing or transfer

Data Sovereignty vs. Data Ownership Example

Scenario: A public transport operator shares real-time vehicle location data through a Mobility Data Space.

Data Ownership – Public Transport Operator

- Owns or holds the rights to the vehicle location dataset.
- Authorizes its use by third parties.
- Defines the permitted purposes and conditions of use.

Data Sovereignty – Data Space Infrastructure

- Enforces the data owner's usage policies through technical mechanisms.
- Controls who can access and process the data.
- Restricts unauthorized use, redistribution, or further sharing.
- Ensures that usage policies remain enforceable after data sharing.

Data Owner: Defines rights, permissions and usage policies.
Data Space Provider / Infrastructure: Provides technical mechanisms to enforce usage policies.
Data Consumer: Uses data according to the granted permissions and constraints.

Decentralized Trust Architecture

Conventional web trust architectures typically rely on centralized trust authorities, such as Certificate Authorities (CAs) and OAuth 2.0 / OpenID Connect Identity Providers, creating dependencies on **centralized entities** for identity **verification**, **authentication**, and **trust management**.

Data Space ecosystems can leverage **Self-Sovereign Identity (SSI)**, enabling autonomous participant nodes to establish and verify identities, and exchange verifiable credentials and claims through cryptographic proofs, **without relying on real-time queries to a central identity authority**.

Self-Sovereign Identity (SSI)

SSI is a digital identity management model in which the individual to whom an identity belongs **retains control** over their credentials and decides when and to whom to present them.

In the traditional approach:

- An Identity Provider manages identities.
- Users authenticate themselves with a central authority.
- Applications rely on that authority to verify identity.

With SSI:

- Each subject has a verifiable digital identity.
- Identity information is contained in cryptographically signed digital credentials.
- Credentials can be verified directly by other parties, without necessarily questioning the authority that issued them.

SSI does not mean that every entity can self-certify any information. Credentials are normally issued by recognized and trusted third parties.

Self-Sovereign Identity (SSI)

Three main actors:

Issuer: The entity that issues a digital credential and certifies its contents.

Example: A public authority certifies that an organization is a public transportation entity.

Holder: The person who receives, stores and controls his digital credentials.

Example: A transport operator keeps the credential that certifies his identity.

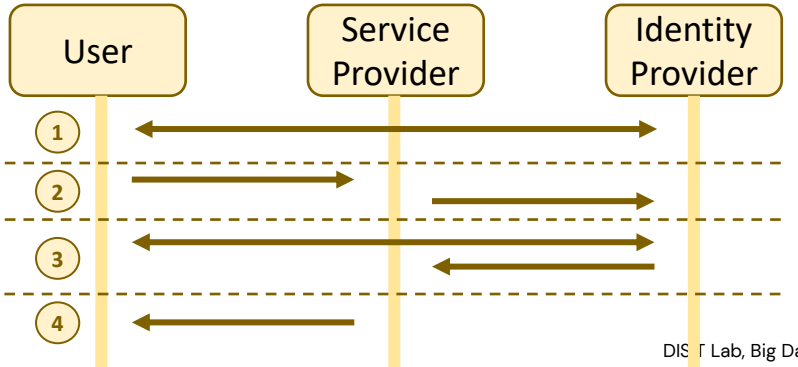
Verifier: The entity that requests and verifies a credential to ascertain the identity or specific characteristics of another entity.

Example: A Data Space verifies that a participant is an authorized transportation operator.

Centralized Identity Management vs SSI example

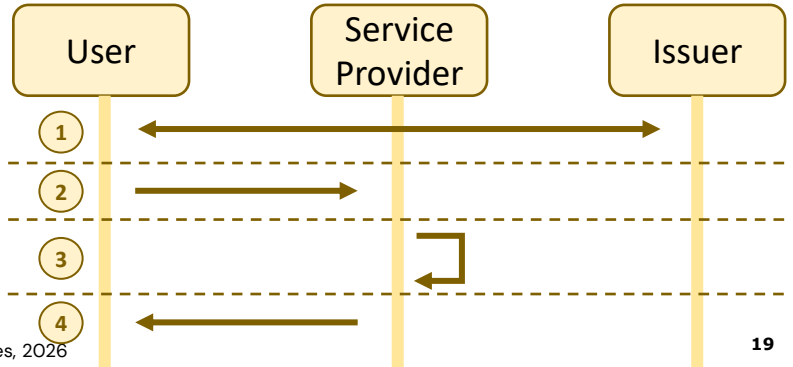
Login via centralized Identity Provider

1. The User registers with the central Identity Provider. The Identity Provider manages the organization's identity and credentials.
2. When accessing the Service, the Service hands off auth to the Identity Provider.
3. The Identity Provider validate the User identity and send confirmation to the Service (issues an access token).
4. The Service validates the token (relying on the central Identity Provider) and allow the user to log in.



Login via SSI-based architecture

1. A trusted authority (Issuer) issues a Verifiable Credential certifying the user. The User stores the credential in its digital identity wallet.
2. When accessing the Service, the User presents the credential.
3. The Service acts also as Verifier, and cryptographically verifies the credential, its issuer and its validity.
4. Access is granted to the User.



SSI: DID and VC

- A DID identifies a subject and enables cryptographic verification of its identity
- VCs enable independent verification of digital claims without requiring direct interaction with the original Issuer at every verification

DID – Decentralized Identifiers:
Identify people, organizations, devices, or services without necessarily relying on a centralized Identity Provider.
Associated with a **DID Document**:

- Public cryptographic keys
- Verification methods
- Authentication mechanisms
- Service endpoints (optional)

DID Resolution allow other participants to retrieve the corresponding DID Document.

VC – Verifiable Credentials:

- Issued by a trusted entity (Issuer).
- Contain certified information about a subject (Holder) and the DID of the Issuer.
- Cryptographically signed by the Issuer.
- Stored and managed by the Holder in a digital wallet.
- Presented to a Verifier through a Verifiable Presentation (VP).

Verification process:

1. User shows VC (o VP) to Service.
2. The Service retrieves the Issuer's DID Document and uses the Issuer's public key to verify the signature on User's VC.
3. The Service verifies that the Issuer is trustworthy (Trust Anchors).
4. The Service verifies that the credential is valid and that the attributes meet the requirements.

Participant Roles

A data space architecture distinguishes between four essential roles

Data Owner: The legal entity possessing the legal rights and authority to grant or revoke access and usage terms for a digital asset.

Data Provider: The entity operating a connector instance that packages, exposes, and transfers data assets under authorized conditions.

Data Consumer: An entity operating a client connector that discovers catalogues, negotiates agreements, and ingests payloads.

Data Space Authority: The governance entity responsible for developing, maintaining, and enforcing the operational rulebook and operating trust anchors.

An organization can act as **both Data Owner and Data Provider** when it owns or holds the relevant rights over the data and has the technical capabilities to operate and maintain its own connector within the Data Space.

Alternatively, the Data Owner **can delegate connector management to an external organization**, which may act as a Data Provider on its behalf.

Depending on the services it provides and the applicable legal requirements, this organization may also qualify as a **Data Intermediary** under the EU Data Governance Act (DGA).

NGSI-LD and Context Broker

In a Data Space, data must be **semantically interpretable** to ensure interoperability across participants.

However, complex ontology-based reasoning (e.g., OWL description-logic reasoning) can introduce significant computational overhead and latency, particularly in high-velocity data environments.

NGSI-LD addresses this challenge by providing a lightweight, JSON-LD-based information model and API.

A **Context Broker** can act as stateful, event-driven middleware to manage context information and support real-time data access and subscriptions.

DataSpace Protocol: Coordinating Data Sharing

DSP 2025-1 defines interoperable interactions between independent Data Space participants.

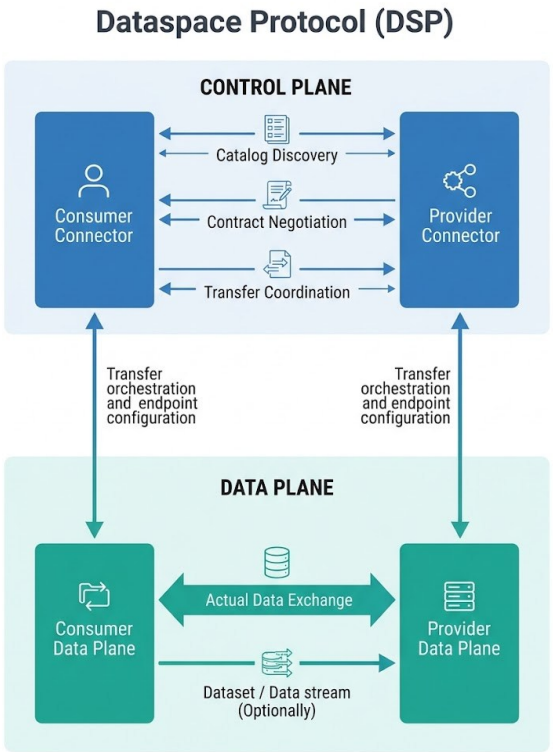
- **Discovery:** Consumers request providers' data catalogues.
- **Contract negotiation:** Participants exchange offers, requests and agreements defining data usage conditions.
- **Transfer orchestration:** Participants coordinate the initiation, monitoring and termination of data transfers.

Control Plane vs. Data Plane

- **Control Plane**: manages discovery, agreements and transfer coordination.
- **Data Plane**: executes the actual data exchange using a selected transport mechanism.

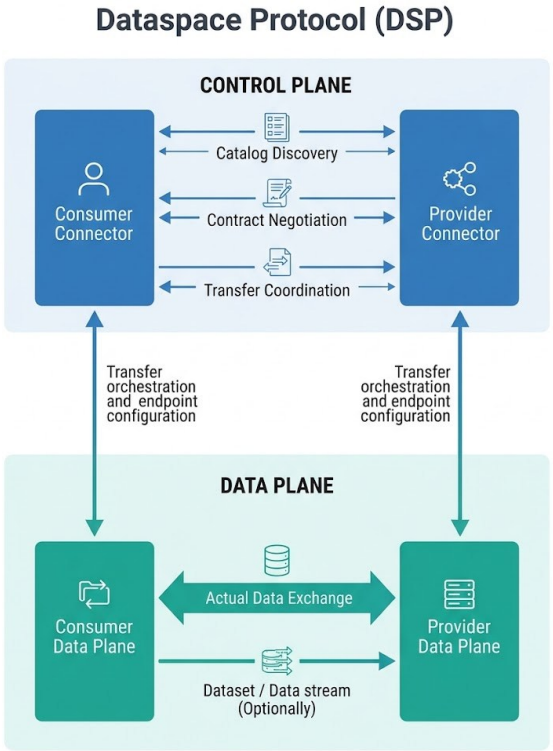
Example – Urban Mobility

A municipality publishes real-time traffic data. A mobility service discovers the dataset, agrees on its usage conditions and obtains access through an HTTP endpoint or streaming service.



Control Plane vs. Data Plane

- Control Plane operates as an **asynchronous signalling plane** that executes DSP (*Catalog Protocol*, *Contract Negotiation*, *Transfer Process*) state machines.
- At the end of the Transfer Process, the Provider's Control Plane dynamically programs a secure route on an API Gateway (Apache APISIX/PEP) and returns to the Consumer the **Endpoint Data Reference (EDR)** containing ephemeral URL and Bearer token to get access to data.
- The Data Plane is activated only later, when the **Consumer client uses the EDR to download data** directly from the Gateway without going through the Control Plane



Standards

DSP reuses established standards

- **DCAT v3:** Describes catalogues, datasets, distributions and data services.
- **ODRL 2.2:** Expresses permissions, prohibitions, duties and usage constraints.
- **Dublin Core Terms (DCT):** Provides common descriptive metadata, such as title, description and publication date.
- **DIDs / Verifiable Credentials:** Can support decentralized participant identification and trust, when adopted by the Data Space identity framework.

Example – Bus Telemetry Dataset

DCAT describes the dataset and its available distributions.

DCT provides its title, description and publication date.

ODRL specifies who can access it and under which conditions.

Participant credentials establish the identity and attributes of the provider and consumer.

Dataset Entry Example

Dataset Metadata and Identification

```
{
"@context": {
  "dcat": "http://www.w3.org/ns/dcat#",
  "dct": http://purl.org/dc/terms/
},
"@id": "https://data.example.org/dataset/florence-bus",
"@type": "dcat:Dataset",
"dct:title": "Florence Urban Bus Positions",
"dct:description": "Real-time positions and speed of urban buses in Florence.",
"dct:issued": "2026-01-15",
"hasPolicy": [
  {
    "@id": "https://data.example.org/policy/bus-research",
    "@type": "Offer",
    "permission": [{
        "action": "use",
        "constraint": [{
            "leftOperand": "purpose",
            "operator": "eq",
            "rightOperand": "academic-research"
        }]
    }],
    "prohibition": [{
        "action": "distribute"
    }]
  }
],
"dcat:distribution": [{
    "@type": "dcat:Distribution",
    "format": "HttpData-PULL",
    "accessService": {
      "@id": "https://provider.example.org/connector",
      "@type": "DataService",
      "endpointURL": "https://provider.example.org/connector"
    }
}]
}
```

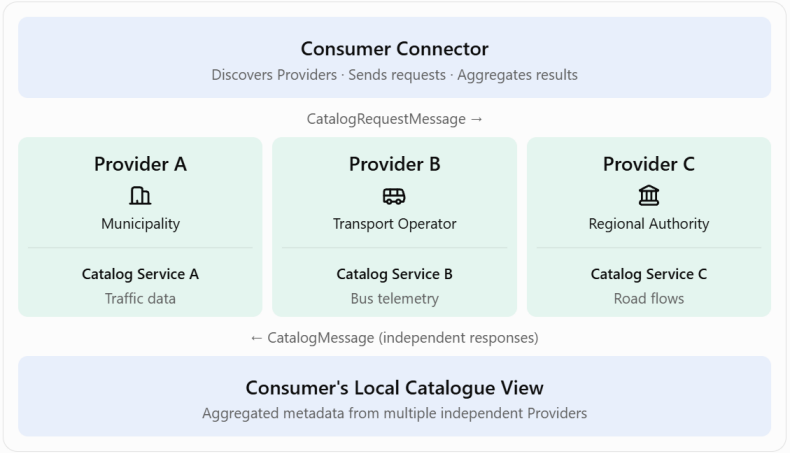
ODRL Policy(Permission)

Data Distribution / Access Information

Finding the Right Data Product

Each Data Space participant manages and exposes its own Catalog Service. There is no need for a single central catalogue: data products remain under the control of their respective Providers.

1. **Provider discovery:** The Consumer identifies available Providers and their Catalog Service endpoints through the Data Space's **discovery mechanism**.
2. **Catalog requests:** The Consumer sends a `CatalogRequestMessage` to one or more Providers.
3. **Catalog responses:** Each Provider returns a `CatalogMessage` containing the data products and offers accessible to that Consumer.
4. **Local aggregation:** The Consumer collects and searches the received catalogues to identify relevant data products.
5. **Selection:** The Consumer selects a data product and proceeds with contract negotiation with the corresponding Provider.



Federation at Scale: A **Catalog Broker** can optionally aggregate catalogues from multiple Providers, offering a unified discovery interface without replacing the Providers' ownership and control of their catalogues.

ODRL: Defining Data Usage Policies

Three fundamental rule types

- **Permission**: What the Consumer is allowed to do.
- **Prohibition**: What the Consumer must not do.
- **Duty**: What the Consumer is required to do.

A policy combines:

- **Target**: The dataset to which the policy applies.
- **Action**: The operation being regulated (e.g. use, distribute).
- **Constraint**: The conditions under which the action is permitted.
- **Party**: The provider (assigner) and consumer (assignee), where applicable.

Policy Lifecycle in DSP

- **Offer**: Provider's proposed usage conditions, published in the catalogue.
- **Request**: Consumer's proposed or accepted terms.
- **Agreement**: Final agreed policy resulting from negotiation.

Operational Usage Constraints

Temporal restrictions: Limit access to a defined time period.

Example: Historical traffic data available until 31 December 2026.

Geographical restrictions: Define the authorized geographic scope of data usage.

Example: Traffic observations may be processed only within the Florence metropolitan area.

Purpose restrictions: Limit usage to an agreed purpose.

Example: Data may be used for academic research, but not for commercial profiling.

Additional obligations: Require anonymization or aggregation before further processing.

Example: Vehicle trajectories must be spatially aggregated before publication.

Contract Negotiation: Asynchronous Interaction

Contract negotiation is a distributed process between independent participant connectors.

- The Consumer initiates a negotiation by sending a `ContractRequestMessage`.
- The Provider creates and manages its negotiation process.
- Both participants maintain their own **process state**.
- **Messages** and **callbacks** allow the participants to coordinate subsequent steps.

Process Correlation

- `consumerPid`: Identifies the negotiation on the Consumer side.
- `providerPid`: Identifies the negotiation on the Provider side.
- Both identifiers correlate messages belonging to the same negotiation.

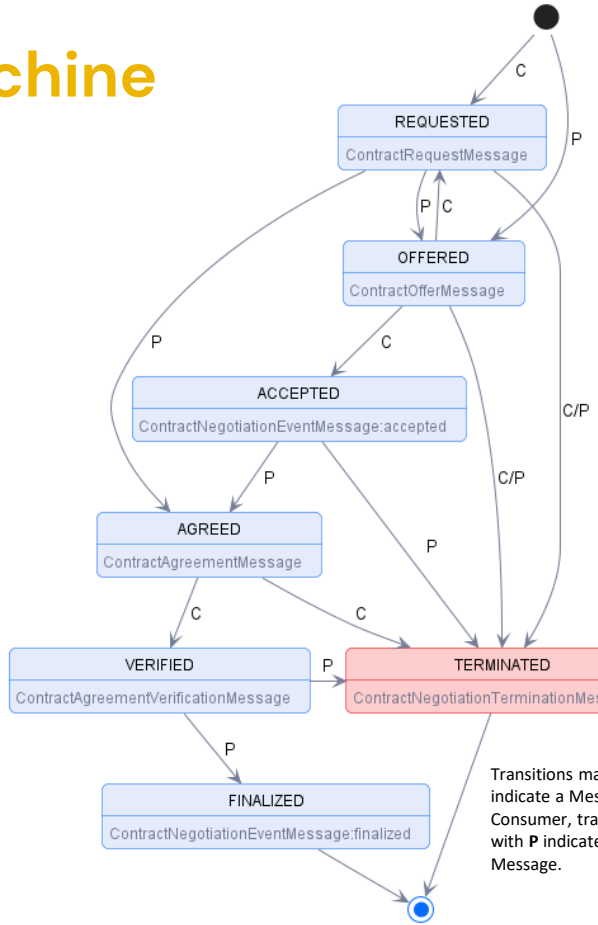
DSP defines a **finite state machine** to coordinate the negotiation lifecycle.

1. **REQUESTED** – Consumer requests an Agreement based on a Provider's Offer.
2. **OFFERED** – Provider issues an Offer or counter-offer.
3. **ACCEPTED** – Consumer accepts the latest Offer.
4. **AGREED** – Provider issues the definitive Agreement.
5. **VERIFIED** – Consumer verifies the received Agreement.
6. **FINALIZED** – Provider confirms finalization; the Agreement is finalized.
7. **TERMINATED** – Negotiation is cancelled by either participant.

Contract Negotiation State Machine

DSP defines a **finite state machine** to coordinate the negotiation lifecycle.

1. **REQUESTED** – Consumer requests an Agreement based on a Provider's Offer.
2. **OFFERED** – Provider issues an Offer or counter-offer.
3. **ACCEPTED** – Consumer accepts the latest Offer.
4. **AGREED** – Provider issues the definitive Agreement.
5. **VERIFIED** – Consumer verifies the received Agreement.
6. **FINALIZED** – Provider confirms finalization; the Agreement is finalized.
7. **TERMINATED** – Negotiation is cancelled by either participant.



From Agreement to Data Exchange

A finalized Agreement authorizes the next step: coordinating the **data transfer**.

Transfer Lifecycle

REQUESTED: Consumer requests a transfer referencing the Agreement (`TransferRequestMessage`).

STARTED: Provider makes the data available or starts pushing it to the Consumer (`TransferStartMessage`).

SUSPENDED: Transfer is temporarily paused.

COMPLETED: Finite data transfer has successfully finished.

TERMINATED: Transfer process has been stopped.

Finite data
- Example: *Downloading a daily traffic dataset or a GTFS archive*.
- The process reaches **COMPLETED** when transmission finishes.

Non-finite data
- Example: *Continuous access to real-time bus positions*.
- The process remains **STARTED** until explicitly suspended or terminated.

Accessing the Data Plane

The Consumer needs a concrete endpoint to access the data after transfer initialization.

For HTTP-PULL transfers, the Provider returns a `DataAddress` in the `TransferStartMessage`.

The `DataAddress`, i.e., Endpoint Data Reference (EDR), may contain:

- **endpoint**: URL of the data access service.
- **endpointType**: Identifier of the selected transfer protocol. It does not point directly to the data broker; it is a temporary route to the Provider's internal API Gateway for this specific transfer process.
- **endpointProperties**: Protocol-specific parameters, potentially including temporary authorization credentials (Bearer Token).

Example – Real-Time Bus Positions

1. Provider provisions an access endpoint through its API Gateway.
2. Provider returns the endpoint and access credentials to the Consumer.
3. Consumer sends HTTP requests directly to the authorized endpoint.
4. Gateway validates access and forwards the request to the underlying data service.

Data Transfer Patterns: PULL vs PUSH

DSP supports two complementary data transfer patterns:

PULL – Consumer retrieves data

- The Provider exposes a data endpoint.
- The Consumer initiates data retrieval.

Example: *HTTP GET to a Provider's API Gateway.*

PUSH – Provider sends data

- The Consumer provides a destination endpoint.
- The Provider initiates data delivery.

Example: *HTTP POST notifications to a Consumer endpoint.*

Mobility Example – Real-Time Bus Positions

- **PULL**: The Consumer periodically retrieves the latest bus positions from the Provider.
- **PUSH**: The Provider automatically sends bus position updates to the Consumer as they become available.

Key Difference: **PULL is Consumer-initiated**; **PUSH is Provider-initiated**.

End-to-End Data Exchange: A Mobility Example

1. Discovery
- Consumer requests the Provider's catalogue.
- Finds the `FlorenceBusTelemetry` dataset and its ODRL Offer.

2. Contract Negotiation
- Consumer requests access under the proposed conditions.
- Provider and Consumer negotiate and finalize the Agreement.

3. Transfer Initialization
- Consumer requests a transfer referencing the Agreement.
- Provider provisions the data access endpoint and returns a `DataAddress`.

Control Plane

4. Data Consumption
- Consumer uses the endpoint (EDR) and authorization information.
- Provider's API gateway validates the request and forwards it to the data service.
- Consumer receives real-time bus position observations.

Data Plane

*DSP standardizes **discovery**, **negotiation** and **transfer** coordination, while the Data Plane provides the actual data access mechanism.*

From Policy Negotiation to Policy Enforcement

An agreed policy is not automatically an enforced policy.

A Data Space agreement may specify:

- Who can access the data.
- Which operations are permitted.
- For which purposes the data can be used.
- Whether data can be stored, redistributed or retained.
- Which obligations the Consumer must fulfil.

The Fundamental Challenge

- Access restrictions can often be enforced by the Provider's connector.
- Usage restrictions must also be enforced within the Consumer's environment.
- Once data has been transferred, the Provider may lose direct control over it.

Key Question: *How can a Provider verify what happens to data after it has been shared?*

Three Different Levels

1. **Policy Specification**: Usage conditions are expressed, for example, using ODRL.
2. **Policy Enforcement**: Technical and organizational mechanisms apply the agreed restrictions.
3. **Policy Compliance**: The Consumer actually respects the agreed conditions throughout the data lifecycle.

How Can Data Usage Policies Be Enforced?

Policy requirement	Possible enforcement mechanisms
Only accredited organizations can access the data	Verifiable Credentials, authentication and access control
Data can only be used for non-commercial purposes	Organizational controls, purpose-specific processing environments, monitoring and audits
Data must not be redistributed	Controlled access, technical restrictions, monitoring and contractual obligations
Data must be deleted after use	Controlled execution environments, retention management and deletion evidence
Only aggregated results can be shared	Provider-side processing, controlled analytics and output filtering

The Limits of Technical Enforcement

Example 1 – Non-Commercial Use Only

A university obtains real-time traffic data under the condition that it can only be used for non-commercial research.

- The Provider can verify the university's identity and credentials.
- The Consumer can be required to declare the research project and use an approved processing environment.
- Access and processing activities can be logged and audited.

However, the connector cannot independently establish that the Consumer will never use the received data for commercial purposes.

Example 2 – No Persistent Storage

A Provider shares real-time bus positions under the condition that the Consumer must not store the data.

- An ordinary HTTP endpoint can restrict access but cannot prevent the Consumer from saving received data.
- A controlled processing environment can keep raw data within a protected infrastructure.
- Only authorized results can be returned to the Consumer.

However, once results leave that environment, their subsequent use cannot always be controlled.

Implementing Data Space Connectors: FIWARE Example

The Dataspace Protocol can be implemented through different open-source connector solutions.

FIWARE Data Space Connector (FDSC)

Provides the main capabilities required by a Data Space participant:

- Participant authentication and trust management.
- Catalog publication and discovery.
- Contract negotiation through DSP.
- Transfer orchestration and controlled data access.

Integrates with FIWARE technologies, including **NGSI-LD Context Brokers**, and can expose other HTTP-based services.

Other Open-Source Solutions: different implementations can provide interoperable connectors by supporting common protocols and trust frameworks.

- Eclipse Dataspace Components (EDC).
- SIMPL-Open – European Commission's open-source middleware for Data Spaces.
- TRUE Connector – an IDS-oriented connector implementation.

FIWARE DSC

FIWARE DSC implements the data-sharing workflow by **integrating different components**, each with a specific responsibility.

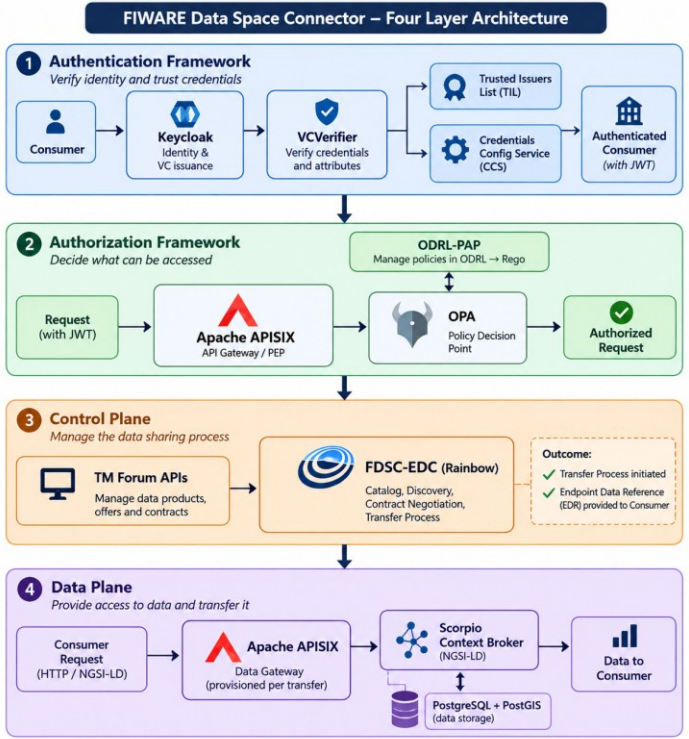
Function	FIWARE component	Role
Product publication and catalog	TM Forum APIs + Marketplace Portal	Manage product specifications, offers and catalog information.
DSP interactions	EDC Framework	Implements catalog discovery, contract negotiation and transfer protocols.
Identity and trust	OID4VC / DCP + VCVerifier + Trusted Issuers List	Authenticate participants and validate their credentials.
Contract management	TM Forum APIs + Contract Management	Manage product orders and connect contracting events with connector configuration.
Transfer provisioning	EDC + APISIX	Configure the data-access route and provide the endpoint and required access credentials.
Runtime authorization	APISIX + OPA	Intercept incoming requests, evaluate applicable policies and enforce access decisions.
Data provision	Provider's data service (e.g. Scorpio, Orion-LD)	Expose the actual data, such as NGSI-LD entities.

Microservices in FDSC architecture

Authentication Framework Microservices:

- **Keycloak**: Acts as the enterprise Identity and Access Management (IAM) provider, upgraded with the oid4vc-vci feature to issue cryptographically signed Verifiable Credentials.
- **VCVerifier**: Functions as an EBSI-compliant Relying Party service. It exposes discovery endpoints, issues cryptographic nonces, validates inbound Verifiable Presentations (VPs) against the holder's DID, and exchanges valid claims for short-lived JWT access.
- **Credentials Config Service** (CCS): Stores the specific credential requirements and claim profiles required to access registered services.
- **Trusted Issuers List** (TIL): Interfaces with domain trust anchors and the European Blockchain Services Infrastructure (EBSI) Trusted Issuers Registry to verify that credential issuers are officially accredited.

During the machine-to-machine (M2M) handshake, the consumer acquires a Verifiable Presentation from its local Keycloak wallet and posts it to the provider's VCVerifier. Upon verifying the VP and consulting the TIL, the VCVerifier send a Bearer Token containing the verified participant claims to the consumer.

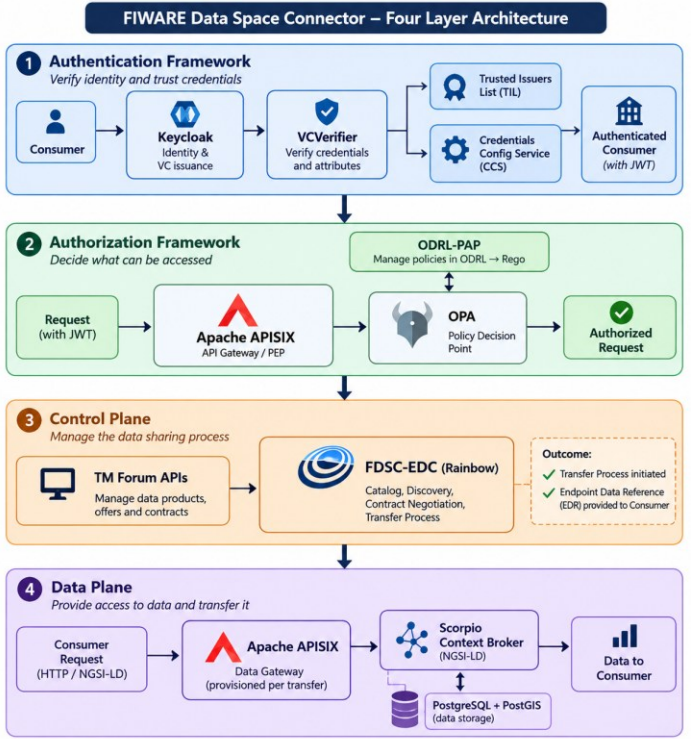


Microservices in FDSC architecture

Authorization Framework Microservices:

- **Apache APISIX**: A high-performance, cloud-native API Gateway acting as the Policy Enforcement Point (PEP). It serves as the sole external ingress point for all incoming HTTP requests.
- **Open Policy Agent** (OPA): Operates as the Policy Decision Point (PDP), evaluating access requests against rule in Rego
- **ODRL-PAP**: Functions as both the Policy Administration Point (PAP) and Policy Retrieval Point (PRP), covering both policy storage and translation from ODRL in Rego.

When APISIX intercepts an authenticated REST call containing a Bearer JWT, its native OPA plugin forwards the request context (HTTP method, URI, caller DID, and token claims) to OPA. OPA evaluates active Rego rules compiled by ODRL-PAP alongside dynamic environmental context from the Policy Information Point (PIP). Only when OPA returns `{"is_allowed": true}` *does APISIX forward the request to downstream internal services.*



Microservices in FDSC architecture

Control Plane Microservices:

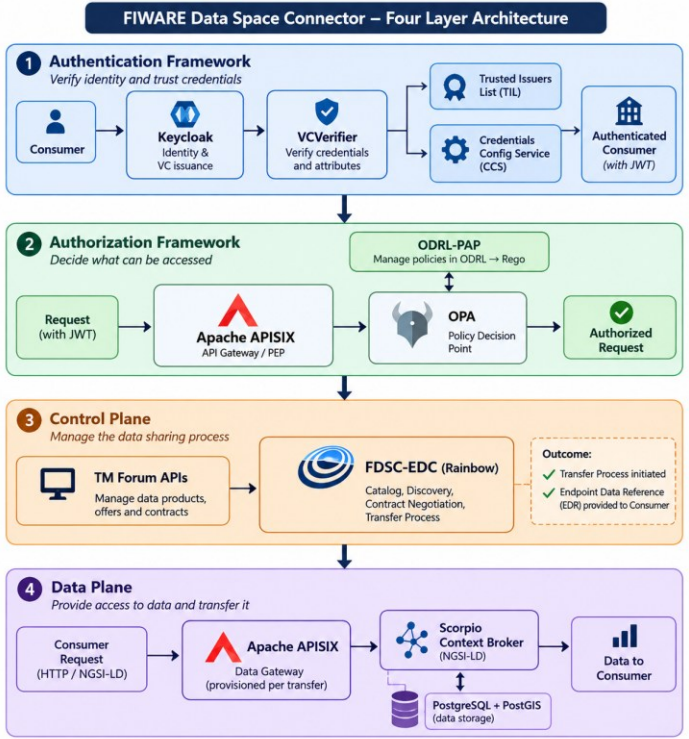
- **TM Forum APIs (TMF-API):** Acts as the internal management dashboard where human administrators package raw technical data (e.g., live bus coordinates or sensor feeds) into commercial data products, setting licensing conditions, prices, and allowed durations.
- **FDSC-EDC Components (Rainbow)**: Converts on-the-fly internal business offerings into open, cross-European standards following DSP (`/api/dsp/catalog`, `/api/dsp/negotiations`, `/api/dsp/transfers`).

Packaging & Dynamic Translation: *The administrator publishes a data product in the Business Backend; Rainbow instantly translates it into standard European catalog entries and machine-readable usage policies.*

Catalog Discovery: *A Consumer queries available data; Rainbow retrieves the local offering and returns the catalog containing asset descriptions and binding usage conditions.*

Automated Negotiation: *The Consumer requests access; Rainbow verifies identity claims, checks policy compatibility, and signs an immutable digital agreement machine-to-machine.*

Rule Push & Gateway Handover: *Rainbow stores the signed rules into the policy engine (ODRL-PAP) and triggers the API Gateway to open a temporary route, sending a secure access ticket to the Consumer.*



Microservices in FDSC architecture

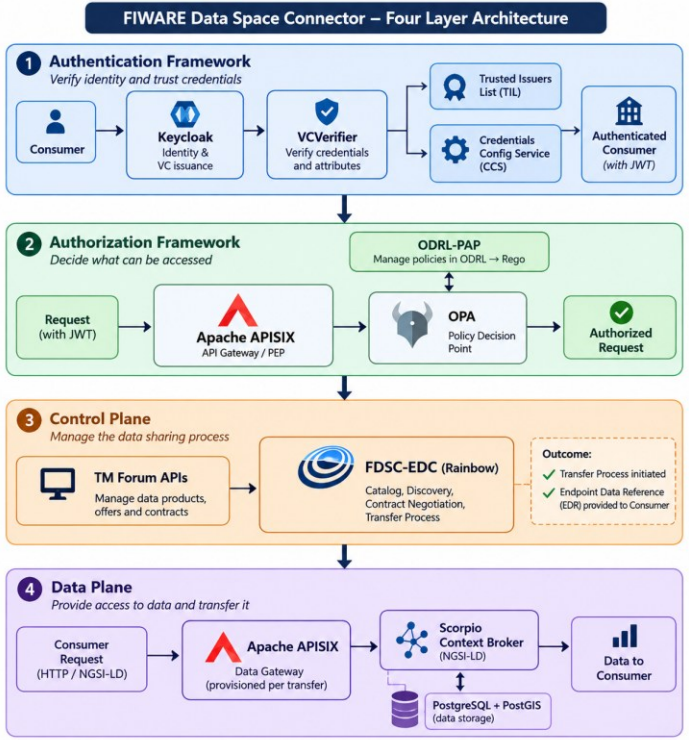
Data Plane Microservices:

- **Scorpio Context Broker:** An ETSI NGSI-LD compliant context broker backed by PostgreSQL and PostGIS. It acts as the actual data container, maintaining entity digital twins with native temporal (`observedAt`) and spatial (`GeoProperty`) capabilities.
- **EDC Dataplane** (APISIX Proxy): Implements the EDC data transfer functionality through APISIX, which dynamically provisions transfer-specific endpoints, enforces access policies and routes Consumer requests to the Provider's data services.

When a transfer process is initiated, Rainbow (FDSC-EDC) instructs APISIX to spin up an ephemeral, transfer-specific ingress route keyed on the `transferProcessId`.

The consumer connector receives an Endpoint Data Reference (EDR) containing the dynamic proxy URL and security token.

The consumer issues an HTTP GET / NGSI-LD query directly to the APISIX route. APISIX validates the token against OPA, logs telemetry, and transparently proxies the call to Scorpio, streaming the contextual payload back to the consumer without Control Plane bottlenecking.



From FIWARE DSC to a Cloud-Native Application

A Data Space Connector is a distributed application, not a standalone service.

The FIWARE DSC architecture integrates several independent services:

- Identity and credential management.
- DSP protocol and contract management.
- API Gateway and policy enforcement.
- NGSI-LD Context Broker and databases.

The Deployment Challenge

Each service **runs independently**, with its own configuration, dependencies and network requirements.

Services must **communicate reliably** and be **started**, **monitored** and **updated** consistently.

Managing these relationships manually becomes complex as the number of components grows.

Cloud-Native Architecture

Moving from individual components to an operational Data Space Connector requires both **container orchestration** and **coordinated deployment management**.

- **Kubernetes** orchestrates the containerized services and manages their interactions.
- **Helm Umbrella Charts** package the connector and its dependencies into a configurable deployment.

Kubernetes and Helm



Kubernetes – Container Orchestration

Kubernetes is an **orchestration platform** that manages containerized applications across a cluster of computing nodes. It automates their deployment and lifecycle, provides service discovery and networking, monitors application health, and maintains the desired state of the system.



Helm – Kubernetes Package Management

Helm is a **package manager** for Kubernetes applications. It uses **Charts** to describe how applications and their resources should be deployed and configured. It supports reusable configurations, versioned releases, upgrades and rollbacks.

Different but complementary roles:

- Kubernetes manages and operates the deployed services.
- Helm packages, configures and installs the application on Kubernetes.

Deploying FIWARE DSC: An Integrated Example

Helm and Kubernetes work together to deploy the multiple services composing the FIWARE Data Space Connector.

1. Define the Deployment – Helm Umbrella Chart

- The `FIWARE DSC Umbrella Chart` groups the charts and dependencies required by the connector into a single deployment package.
- Configuration values specify which components to deploy and how they should interact.

2. Deploy the Components – Kubernetes

Helm submits the resources defined by the charts to Kubernetes, which creates and manages the corresponding application resources:

- Identity and trust services.
- EDC components (discovery, negotiation)
- API data access (APISIX and OPA) NGSI-LD context management (Scorpio, Orio-LD).
- Databases supporting the persistence requirements.

3. Operate the Connector – Kubernetes

Kubernetes manages the resulting Pods and their lifecycle:

- Services provide stable network endpoints for communication between components.
- Namespaces organize the connector resources.
- NetworkPolicies can restrict communication between services.
- Health probes and self-healing mechanisms supporting service availability.

Integrating Snap4City into a Data Space

A Data Space Connector can extend an existing Smart City platform without replacing its internal architecture.

Snap4City already manages heterogeneous urban data through its IoT infrastructure, Knowledge Base, semantic models and analytics services. However, participating in a Data Space also requires mechanisms for:

- Decentralized participant identity and trust.
- Standardized catalog and contract interactions.
- Controlled data access through interoperable protocols.

External Connector Bridge
The FIWARE Data Space Connector is deployed alongside Snap4City and **acts as an interface** between the platform and the Data Space.

Snap4City as Data Provider
- Select urban datasets from the Snap4City Knowledge Base.
- Transform data into NGSI-LD context entities.
- Publish the data through Scorpio and expose it as a Data Product.

Snap4City as Data Consumer
- Discover and negotiate access to external Data Products.
- Retrieve data through the endpoint and credentials provided by the connector.
- Integrate the received information into Snap4City storage, dashboards and analytics.

Bridging Urban Data and NGSI-LD

Node-RED implements the data integration pipelines between Snap4City and the FIWARE Data Space Connector.

Provider Pipeline – Snap4City → Scorpio

1. **Retrieve:** Periodically query the Snap4City Knowledge Base for selected urban measurements.
2. **Transform:** Convert the original JSON records into NGSI-LD entities, including properties and temporal metadata.
3. **Publish:** Send the resulting entities to Scorpio through the NGSI-LD API.

Consumer Pipeline – Data Space → Snap4City

1. **Authenticate:** Extract the endpoint and access token from the Endpoint Data Reference (EDR).
2. **Retrieve:** Execute an HTTP GET request through the Provider's authorized data endpoint.
3. **Transform:** Convert the received NGSI-LD entities into the data representation expected by Snap4City.
4. **Ingest:** Store the information and make it available to dashboards and analytics.