Ref. Ares(2025)11062671 - 12/12/2025

HORIZON –CL4-2022-TWIN-TRANSITION-01-06
HORIZON Innovation Actions
Grant Agreement No.: 101092295

Circular and Dynamic Manufacturing Supply Chain Orchestration and OptimiSation

D4.5 Final version of Integrated Platform

Report Identifier:	D4.5		
Work-package:	WP4	**Task:**	T4.4
Responsible Partner:	ED	**Version Number:**	1.0
Due Date	M24	**Document Date:**	12.12.2025
Distribution Security:	PU	**Deliverable Type:**	DEM
Keywords:	Federation, NGSI-LD, FIWARE, Orion-LD, Mintaka, Keycloak, Data Platform, Data Interoperability, Data Synchronisation, Local/Factory Deployment, Central Platform		
Project website: https://circuloos.eu/			

Document History

Version	Content & Changes	Issue Date
0.1	**Document created**	**09.06.2025**
0.2	**Document sent for review**	**28.11.2025**
0.3	**Document reviewed**	**04.12.2025**
0.4	**Document reviewed**	**12.12.2025**
0.5	**Reviews are combined**	**12.12.2025**
0.6	**Sent for Quality Assurance**	**12.12.2025**
1.0	**Quality Assurance and Submission**	**12.12.2025**

Quality Control

	Organisation	Date
Editor	ED	09.06.2025
Peer review 1	MWCB	04.12.2025
Peer review 2	F6S	12.12.2025
Authorised by (Technical Coordinator)	ED	12.12.2025
Authorised by (Quality Manager)	ED	12.12.2025
Submitted by (Project Coordinator)	ED	12.12.2025

Legal Disclaimer

Copyright notice

Table of Contents

List of Figures

Abbreviations

Acronym	Description
AI	Artificial Intelligence
API	Application Programming Interface
CDM	Common Data Model
CE	Circular Economy
CIM	Context Information Management
CM	Circular Manufacturing
CMRA	Circular Manufacturing Reference Architecture
CSV	Comma-Separated Values
DT	Digital Twin
ETSI	European Telecommunications Standards Institute
EXD	Experiment Demonstrator (from the Open Calls)
GUI	Graphical User Interface
IAM	Identity and Access Management
IIoT	Industrial Internet of Things
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
JWT	JSON Web Token
KPI	Key Performance Indicator
LCA	Life Cycle Assessment
LD	Linked Data
MSME	Manufacturing Small and Medium Enterprise
NGSI	Next Generation Service Interfaces
NGSI-LD	Next Generation Service Interfaces – Linked Data
PEP	Policy Enforcement Point
SDM	Smart Data Model
SSO	Single Sign-On
SSL / HTTPS	Secure Sockets Layer / HyperText Transfer Protocol Secure
UN/CEFACT	United Nations Centre for Trade Facilitation and Electronic Business
URI / URN	Uniform Resource Identifier / Name
SCDT	Supply Chain Digital Twin Tool

SCOPT	Supply Chain OPTimization
VC / VP	Verifiable Credential / Verifiable Presentation
WP	Work Package

Executive Summary

This deliverable (D4.5) introduces the federation capabilities of the CIRCULOOS Data Platform, extending the centralised architecture presented in D4.4 into a distributed, multi-node environment. The new federation mechanism allows each factory or partner to operate its own local Orion-LD broker while securely synchronising selected NGSI-LD entities with the central CIRCULOOS platform.

Federation is implemented using only the components available in the project's GitHub repository, including the Registration-to-Entities Service, Keycloak, and Kong. Local Orion-LD brokers manage factory-level data, trigger NGSI-LD notifications for monitored entity types, and forward updates to the central platform through the registration service after obtaining authentication tokens.

This deliverable provides an overview of the federation concept and architecture, a detailed workflow based on actual repository code, configuration guidance for partners, and an executed example showing how locally created entities become available to central tools such as GRETA, SCDT, and SCOPT. The work presented here establishes a practical, standards-based foundation for decentralised deployments across the CIRCULOOS ecosystem.

The repository is publicly accessible (offered as Open Source code) in Github at this link: https://github.com/european-dynamics-rnd/circuloos-data-platform

1 Introduction

1.1 Project Introduction

The overall vision of CIRCULOOS is to deliver the tools to enable MSMEs become full members of the Circular Manufacturing value chain. These tools orchestrate and continuously optimise the supply-chain end-to-end and integrate planning and execution monitoring to enable transparent and on-time communication. Combining these with direct calculation of the product sustainability and circularity profile, for both internal and external partners, this environment will enable them to configure and execute disruptive circular manufacturing processes for sustainable production that covers the entire life cycle of products; either by recovering the value of product that ended-up as waste or from recycled and remanufactured products.

To achieve this objective the project aims to deploy:

- Circular end-to-end supply chain orchestration for collaborative workflows which incorporates planning and execution metrics and integrates advanced and multimodal visualisation and analytics. The visualisation is delivered by comprehensive Digital Twins of the supply chains formulated, the factory processes and product design phases.
- Supply Chain Optimisation that monitors the global (across the supply chain) and local (within the factory) processes and execution, inputs and outputs and configuration parameters, to enable data-driven AI decision making, this way supporting continuous optimisation of targeted and measured performance and sustainability parameters.
- Dynamic Sustainability Assessment functionalities that investigate alternative supply-chain scenarios (varying in terms of materials used, processing technologies, suppliers involved and/or activated circular economy practices) in place of the existing schemes, quickly measuring their performance in terms of environmental sustainability and circular economy profile.
- Supply Chain Data Spaces for seamless, multi-level data flow across the supply chain partners, supporting the reuse of materials in novel products, the extension of the life-cycle of finished products (remanufacturing), and data-driven decisions for collaboration of parties offering matching services in the most dynamic and efficient way.
- Cybersecure and trustworthy data sharing across the supply chain by employing a distributed, trusted and efficient Identity and Access management system, that together with the associated trust framework will coordinate the identities of all IoT objects and ensure trustworthy data sharing among its members, aligned with the trust framework that is being implemented in EBSI.
- CM specific tools for the automatic recognition of recyclable parts by modern Machine Vision tools and Advanced Robotics, to enable optimised flows in the selection process.
- Novel circular business processes will be demonstrated supporting reusing, reducing, and recycling material in production and consumption systems. The new collaborative production models will provide quantifiable results on the sustainability increase across the supply chain, in terms of efficient use of raw materials, of by-products, of waste and energy and of emissions reduction. CIRCULOOS leverages the above with the RAMP integrated innovation IOT platform and the European network around it to deliver a CM ecosystem and platform for Manufacturing SMEs.
- Skills upskilling and reskilling will be provided in RAMP and through online courses, webinars, and best practice guides and success stories based on the pilots and Experiments for Demonstration (EXDs).

1.2 Deliverable Purpose

The purpose of Deliverable D4.5 Final version of the integrated platform is to present the updated and extended version of the CIRCULOOS Data Platform, incorporating new functionalities that enhance interoperability, security, and distributed data management across the project's ecosystem.

Building on the initial release described in D4.4, this version introduces the Federation Feature, which enables local factory instances of the platform to interconnect with the central CIRCULOOS infrastructure. Through this mechanism, each factory can operate its own local Orion-LD broker while sharing selected data with the central platform using the Registration-to-Entities Service. This ensures both data sovereignty at the partner level and cross-pilot collaboration at the project level.

2 Overview of the Federation Concept in CIRCULOOS

2.1 Federation Concept and Motivation

The Federation Concept in CIRCULOOS represents the evolution of the platform from a single, centralised data environment into a distributed network of interconnected nodes, each operated by a factory, pilot, or partner organisation. Its goal is to ensure that every partner can manage and control its own operational data locally, while still contributing selected information to the common CIRCULOOS ecosystem for collaboration, analytics, and circular-value-chain optimisation.

In the first version of the platform (D4.4), all entities were created and stored directly in the central Orion-LD Context Broker. While this model supported interoperability and rapid deployment, it required that all data be sent to one shared instance. The new federation mechanism, introduced in D4.5, extends this approach by enabling local Orion-LD brokers, installed at factory level, to synchronise chosen data with the central platform automatically and securely.

This is achieved through the Registration-to-Entities Service, implemented in the open-source repository. The service acts as a bridge between local and central brokers:

- Each local Orion-LD instance holds operational entities (e.g., production data, sensor readings, or material descriptions) relevant to the partner's processes.
- A subscription is configured locally to monitor specific entity types or updates.
- When a change occurs, the Registration-to-Entities Service receives a notification, authenticates through Keycloak, and forwards the data to the central Orion-LD via secure APIs routed through Kong.
- The central broker then stores or updates the corresponding entity, preserving timestamps and relationships according to the NGSI-LD standard.

This architecture enables a hybrid model where data can stay local for privacy or regulatory reasons but still be shared selectively for project-wide analytics, digital-twin synchronisation, or sustainability evaluation in GRETA.

Key advantages of this approach include:

- Data sovereignty: factories decide which data are published to the central platform;
- Scalability: each pilot can operate autonomously without overloading the central broker;
- Interoperability: all instances remain NGSI-LD compliant, ensuring seamless integration;

- Security: authentication and access control are managed centrally through Keycloak and Kong.

2.2 How Federation works in CIRCULOOS

The CIRCULOOS Federation establishes the technical process by which data created in local factory environments are synchronised with the central CIRCULOOS platform. This mechanism ensures that distributed data sources remain interoperable while preserving autonomy for each factory node. At its core, the federation operates through a publish–subscribe and forwarding model, implemented using FIWARE Generic Enablers and the Registration-to-Entities Service available in the GitHub repository.

Each factory runs a local Orion-LD Context Broker, which manages entities such as products, sensors, or production processes. When a new entity is created or updated, the local Orion-LD generates a notification based on an active subscription. This notification is sent to the Registration-to-Entities Service, a lightweight component that handles all communication with the central system. The service authenticates with Keycloak, obtains a valid token, and uses Kong as the secure API gateway to transmit the update to the central Orion-LD instance hosted in the main CIRCULOOS environment. The central broker then performs an upsert operation, creating or updating the entity according to the NGSI-LD standard. This flow ensures that data remain consistent between local and central systems and that every exchange is verified, logged, and compliant with access control policies. In addition, the same mechanism supports future extensions for multi-pilot interoperability, allowing several local brokers to federate simultaneously with the central platform.

2.3 Local Orion-LD (factory-level)

At the factory level, each pilot or manufacturing partner operates a local Orion-LD Context Broker, which manages contextual data within its own domain. This broker is responsible for storing, updating, and exposing NGSI-LD entities that describe production processes, materials, machines, and measurements.

The local Orion-LD serves as the entry point for all local data ingestion. Entities can be created either automatically (e.g., via IoT or machine systems) or through helper tools such as the CSV-to-Orion Agent, which converts CSV datasets into NGSI-LD entities and uploads them directly to Orion-LD.

Configuration and deployment of the local broker follow the standard FIWARE Docker Compose setup defined in the repository.
In this setup:

- The Orion-LD container listens for NGSI-LD API requests (POST, GET, PATCH, DELETE).
- It stores contextual entities locally and supports subscriptions to monitor specific entity types or attributes.
- When changes occur (creation or updates), notifications can be sent to external services such as the Registration-to-Entities Service for federation.

This local broker acts as a self-contained CIRCULOOS node, allowing each partner to operate independently while remaining semantically compatible with the central platform through the shared CIRCULOOS Common Data Model (CDM).

2.4 Central Orion-LD (CIRCULOOS platform)

The Central Orion-LD is the main Context Broker instance deployed in the official CIRCULOOS cloud environment. It operates as the core of the platform's data management layer, providing a unified view of all entities federated from local installations.

The central Orion-LD:

- Receives NGSI-LD entities via REST API calls (mainly POST /ngsi-ld/v1/entityOperations/upsert and PATCH /ngsi-ld/v1/entities/{id}/attrs),
- Maintains the latest contextual state of all federated entities,
- Exposes query endpoints for tools like GRETA, SCDT, and SCOPT, and
- Integrates with Mintaka for historical and time-series data retrieval.

In the centralized setup currently available in the GitHub repository (chain_of_transaction examples), the central Orion-LD directly handles product and transaction entities (e.g., creation, updates, ownership transfers). When federation is enabled, it also receives data from external local brokers through the Registration-to-Entities Service, ensuring both data consistency and traceability across distributed nodes.

2.5 Registration-to-Entities service (bridge between them)

The Registration-to-Entities Service is the core component enabling federation between local and central Orion-LD instances. Its implementation can be found in the repository under:
`/circuloos-registration-to-entities/registrationToEntities.py`.

This lightweight Python service acts as a bridge:

1. It receives notifications from the local Orion-LD whenever monitored entities are created or updated (as defined in the local subscription).
2. It then requests a security token from Keycloak using stored credentials.
3. After authentication, it forwards the updated entity to the Central Orion-LD via a secure REST call through the Kong API Gateway.

The service uses an "upsert" mechanism—if the entity exists, it is updated; if not, it is created.
This ensures that both sides remain synchronised with no duplication or data loss.

The process supports all FIWARE-compliant NGSI-LD payloads and leverages the standard endpoints:

```
/ngsi-ld/v1/entityOperations/upsert
/ngsi-ld/v1/entities/{id}/attrs
```

By relying on JSON-LD contexts and the shared CIRCULOOS CDM, this service guarantees semantic alignment between local and central brokers.

This component essentially makes CIRCULOOS federation-ready, allowing multiple factories to participate in a shared, distributed data space

2.6 Use of Keycloak for authentication and Kong for secure routing

Keycloak and Kong are the two central components ensuring security, authentication, and controlled access across all platform communications.

Their configuration and integration are visible in the Docker setup files and are used by all services, including Orion-LD, Mintaka, and the Registration-to-Entities service.

- Keycloak (Authentication): Keycloak handles user and service authentication for both local and central environments.
  Each service (e.g., Orion-LD, Mintaka, or the federation bridge) is registered as a Keycloak client.
  When a service (like the Registration-to-Entities) needs to send data to the central broker, it first obtains an access token via Keycloak's `/protocol/openid-connect/token` endpoint.
  The token is a JWT (JSON Web Token) that carries user identity and permissions.
- Kong (API Gateway and Routing Layer): All data exchange between local and central brokers passes through Kong, which acts as the API Gateway and Policy Enforcement Point.
  Kong validates the tokens issued by Keycloak, checks access permissions, and routes the requests securely to the appropriate Orion-LD endpoint.
  This guarantees that only authenticated and authorised messages are processed, preventing unauthorised access or tampering.

Together, Keycloak and Kong provide the foundation for secure, federated, and multi-tenant data exchange within the CIRCULOOS ecosystem. This ensures that each factory's data remain protected while still enabling collaboration across the circular manufacturing network.

2.7 Federation Architecture Diagram

The CIRCULOOS platform implements federation using a lightweight mechanism based on local subscriptions and a dedicated registration service. The figure 1 below , taken from the official CIRCULOOS GitHub repository, provides a high-level overview of how local factory data are propagated to the central platform using Keycloak authentication and Kong API Gateway protection.

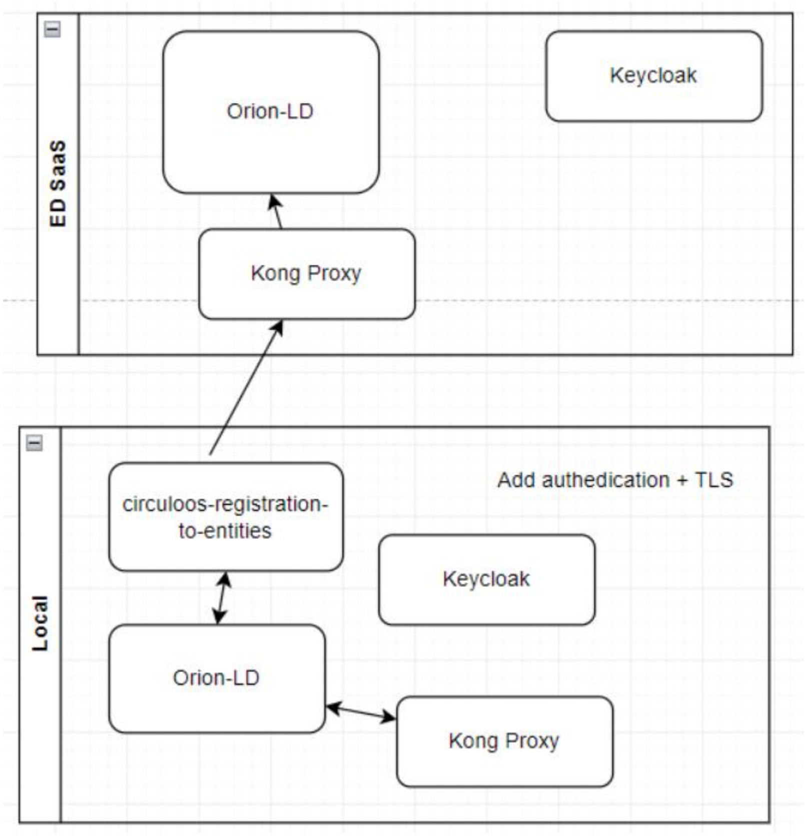


Figure 1 Federation Architecture

Figure 1 illustrates the full path from the local Orion-LD to the central Orion-LD, including the authentication step and the secure routing layer.

3 Federation Workflow: Generic synchronisation process

The CIRCULOOS federation workflow defines how data created in a local factory installation are securely synchronised with the central CIRCULOOS data platform.
The process relies entirely on FIWARE Generic Enablers and the lightweight Registration-to-Entities Service implemented in the official repository: circuloos-data-platform/circuloos-registration-to-entities at master • european-dynamics-rnd/circuloos-data-platform • GitHub

This mechanism ensures that all local contextual updates, such as new products, materials, or process results, are automatically replicated in the central Orion-LD broker while maintaining authentication, authorisation, and semantic consistency through the NGSI-LD standard.

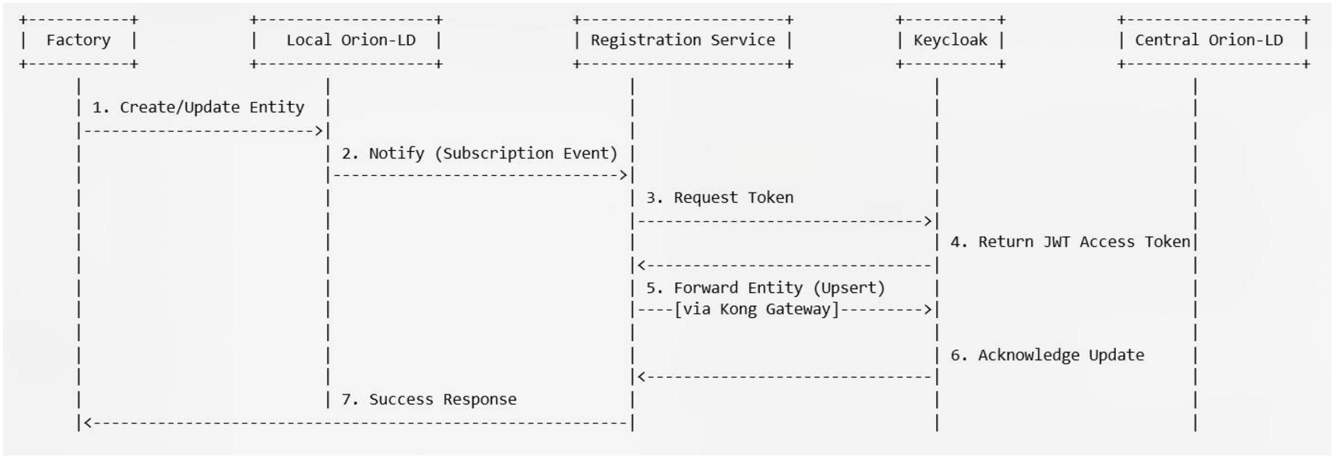
a. Factory creates or updates an entity locally
The local Orion-LD instance at the factory manages all contextual data (e.g., Product, Device, or Process entities). When a factory system or agent (like the CSV-to-Orion agent) performs a POST or PATCH operation on an entity, the update is stored in the local context broker.

b. Local Orion-LD sends a notification to the Registration Service
A subscription is pre-configured in the local Orion-LD to monitor specific entity types or attributes.
Upon detecting a change, Orion-LD automatically triggers a notification to the Registration-to-Entities Service endpoint, including the updated entity payload in NGSI-LD JSON format.

c. The registration service authenticates with Keycloak
Before forwarding the update, the service authenticates with Keycloak, the identity and access management system.
It retrieves a valid access token by sending credentials to the /protocol/openid-connect/token endpoint, as shown in the GitHub configuration file registrationToEntities.py.

d. The service forwards the entity to the central Orion-LD
Using the obtained token, the Registration Service securely transmits the entity to the central Orion-LD via the Kong API Gateway.
Kong validates the JWT token and ensures that only authorised requests reach the Orion-LD /ngsi-ld/v1/entityOperations/upsert endpoint.
If the entity already exists, it is updated; otherwise, it is created—ensuring seamless synchronisation.

e. Confirmation and synchronisation success
Once Orion-LD successfully processes the request, a confirmation response (HTTP 204/201) is returned to the Registration Service, which logs the transaction. This confirms that the entity is now synchronised between the local and central environments. All exchanges are timestamped and traceable, enabling transparent monitoring of the federation process.
The ASCII diagram shown in Figure 2 provides a visual representation of the federation workflow process.



```
+-----------+          +-----------------+          +----------------------+          +----------+          +------------------+
|  Factory  |          |   Local Orion-LD |          | Registration Service |          | Keycloak |          | Central Orion-LD |
+-----------+          +-----------------+          +----------------------+          +----------+          +------------------+
      |                          |                              |                            |                         |
      | 1. Create/Update Entity  |                              |                            |                         |
      |------------------------->|                              |                            |                         |
      |                          | 2. Notify (Subscription Event) |                            |                         |
      |                          |----------------------------->|                            |                         |
      |                          |                              | 3. Request Token           |                         |
      |                          |                              |--------------------------->|                         |
      |                          |                              |                            | 4. Return JWT Access Token|
      |                          |                              |<---------------------------|                         |
      |                          |                              | 5. Forward Entity (Upsert) |                         |
      |                          |                              |----[via Kong Gateway]----->|                         |
      |                          |                              |                            |                         |
      |                          |                              |                            | 6. Acknowledge Update   |
      |                          |                              |<---------------------------|                         |
      |                          | 7. Success Response          |                            |                         |
      |<--------------------------------------------------------|                            |                         |
```

Figure 2 ASCII UML Sequence Diagram of Federation Workflow

4 Configuration and Setup

This section provides a practical guide to deploying and validating the **CIRCULOOS federation mechanism** using the open-source components available in the project's official repository:
GitHub - european-dynamics-rnd/circuloos-data-platform

This directory contains the Python service, documentation, and Docker configuration required for forwarding entity updates from a local Orion-LD instance to the central CIRCULOOS platform.

The federation setup connects a local Orion-LD instance (operated by a factory or partner) with the central CIRCULOOS platform, using the Registration-to-Entities service as a secure bridge. All configurations are provided through Docker and environment variables, enabling rapid deployment and reproducibility.

The federation relies on three components available in the repository:

- registrationToEntities.py → the Python service that receives NGSI-LD notifications and forwards them to the central context broker.
- Readme_federation.md→ documentation describing how the federation flow works and how the service interacts with local Orion-LD, Keycloak, and the central platform.
- The Docker Compose YAML file→ used to run the federation service as a container (name appears in the folder; typically includes the registration service configuration).

These files together enable a partner to activate federation between a factory's local environment and the central CIRCULOOS broker.

4.1 Using the Federation Docker Compose File

The project repository provides a Docker Compose file that launches the Registration-to-Entities service. This service exposes an HTTP endpoint used by the local Orion-LD subscription to send notifications.
The repository documentation (`Readme_federation.md`) explains that:

- The registration service listens for NGSI-LD notifications from the factory's local broker.

- It forwards the corresponding entity update to the central Orion-LD.
- It authenticates using Keycloak, which is also referenced in the Python script.

Running the service is done using the standard command: docker-compose -f <compose-file>.yml up as indicated in GitHub

4.2 Starting the Service and Enabling Federation

The process follows the workflow documented in Readme_federation.md and relies on the `registrationToEntities.py` service running in a Docker container. Once the service is started, the local Orion-LD instance can forward entity updates to the central CIRCULOOS platform through NGSI-LD subscription notifications.

The steps to enable and validate this federation behaviour are as follows:

1. Start the Registration Service

From the circuloos-registration-to-entities/ directory:

```
docker-compose -f <compose-file>.yml up
```

This starts the Python service that will process incoming notifications.

2. Create a Subscription in Local Orion-LD

The Readme_federation.md file includes an example subscription that sends notifications to the registration service endpoint:

```
"endpoint": {
  "uri": "http://registrationtoentities:8888/proxy",
  "accept": "application/json"
}
```

This defines where local Orion-LD sends updates.

3. Perform Local Entity Updates

Any entity created or updated in the local Orion-LD instance that matches the subscription triggers a notification.

4. Federation Service Forwards the Update

According to the Python script:

- the service receives the notification,
- requests a token from Keycloak,
- forwards the update to the central Orion-LD.

5. Verify the Forwarded Entity on the Central Platform

The entity should now be visible in the central broker when queried.

All these steps are aligned with the behaviour described in the repository documentation.

5 Reference implementation of the federation mechanism from GitHub

Real example from the repository:

5.1 Overview and purpose of the Federation Bridge component (registrationToEntities.py)

The CIRCULOOS repository includes a working implementation of the federation mechanism located in:
`/circuloos-registration-to-entities/registrationToEntities.py`
This script acts as the bridge between a local Orion-LD instance and the central CIRCULOOS platform, as documented in: `/circuloos-registration-to-entities/Readme_federation.md`
The purpose of the module is to:

- Receive NGSI-LD notifications from the local Orion-LD.
- Request an authentication token from Keycloak.
- Forward the received entity payload to the central Orion-LD (via Kong).

This mechanism was created for synchronising entity updates from a factory to the central platform.

5.1.1 Functional behaviour of the federation bridge -registrationToEntities.py

1. The service exposes an HTTP endpoint

The Python script starts a small web server (Flask) that listens for POST requests on a configurable port (commonly 8888). This endpoint receives NGSI-LD notification payloads from the local Orion-LD subscription.

2. It extracts the notified entity payload

When Orion-LD triggers a subscription, the body of the POST request contains a standard NGSI-LD notification structure, including:

- `data`: list of updated entities
- metadata fields such as `subscriptionId`, `notifiedAt`, `type`, etc.

The script parses the `data` element to extract each updated entity.

3. It requests an access token from Keycloak

Using values defined in environment variables (KEYCLOAK URL, CLIENT_ID, SECRET, USERNAME, PASSWORD), the script makes an HTTP POST request to Keycloak's token endpoint to obtain an OAuth2 bearer token.

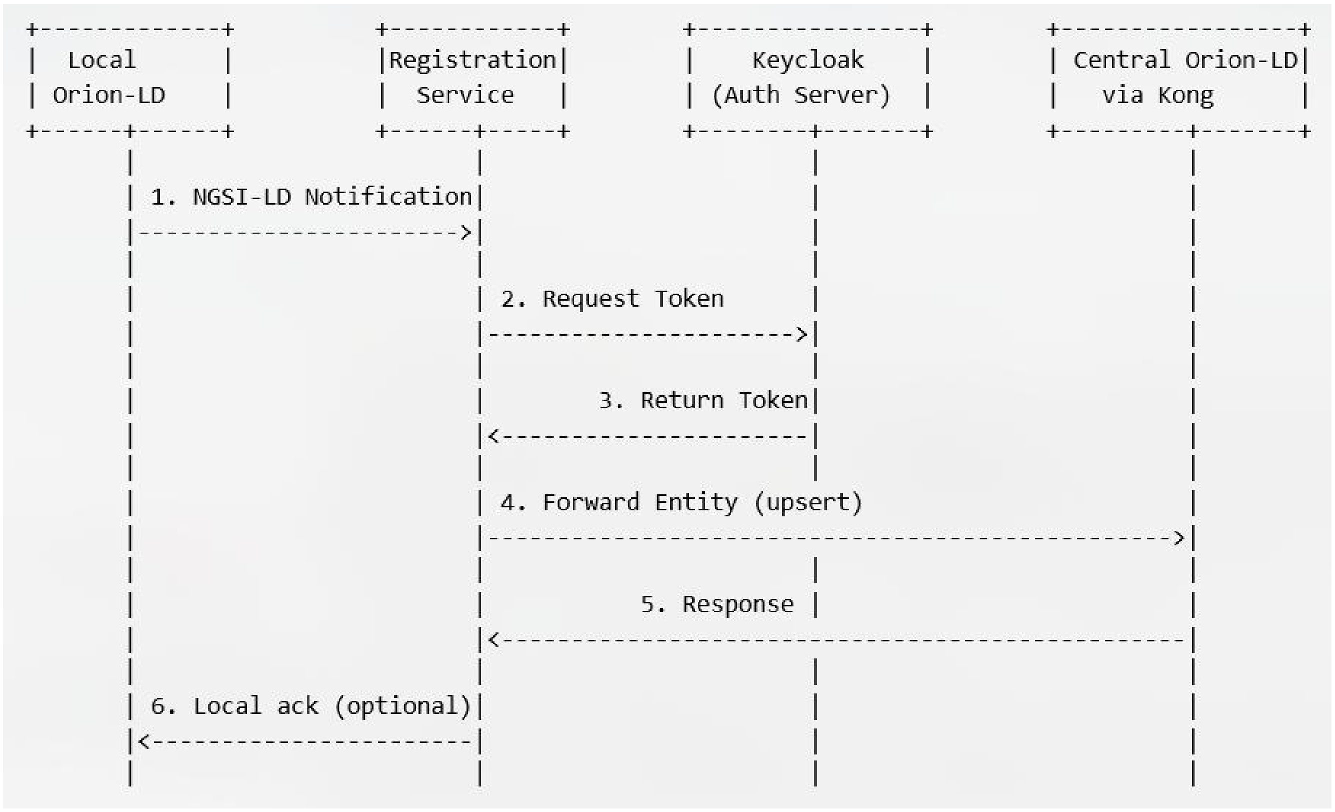
4. It forwards the entity to the central Orion-LD

Once the token is retrieved, the script constructs a POST request and sends the updated entity to:
`/ngsi-ld/v1/entityOperations/upsert` exposed via the Kong gateway of the central platform.

5. It logs the success/failure

The script prints log messages to stdout so the user can verify that federation is operating correctly.

The following ASCII diagram (Figure 3) reflects the exact interaction implemented in `registrationToEntities.py` and documented in `Readme_federation.md`.



```
+-------------+          +------------+         +----------------+          +-----------------+
|  Local      |          |Registration|         |    Keycloak    |          | Central Orion-LD|
| Orion-LD    |          |  Service   |         | (Auth Server)  |          |    via Kong     |
+------+------+          +-----+------+         +-------+--------+          +--------+--------+
       |                       |                        |                            |
       | 1. NGSI-LD Notification|                       |                            |
       |---------------------->|                        |                            |
       |                       |                        |                            |
       |                       | 2. Request Token       |                            |
       |                       |----------------------->|                            |
       |                       |                        |                            |
       |                       |        3. Return Token |                            |
       |                       |<-----------------------|                            |
       |                       |                        |                            |
       |                       | 4. Forward Entity (upsert)                          |
       |                       |---------------------------------------------------->|
       |                       |                        |                            |
       |                       |            5. Response |                            |
       |                       |<----------------------------------------------------|
       |                       |                        |                            |
       | 6. Local ack (optional)|                       |                            |
       |<----------------------|                        |                            |
       |                       |                        |                            |
```

Figure 3 Sequence Diagram of the Federation Bridge Workflow (exact flow: Notification → Token request → Forward to central Orion-LD)

5.1.2 Example NGSI-LD notification and forwarding payload

The GitHub repository includes example notification and forwarding structures inside the script itself. For example, the script handles notifications containing structures like:

```
{
  "type": "Notification",
  "data": [
        {
            "id": "urn:ngsi-ld:leather:example123",
            "type": "leather",
            ...
        }
  ]
}
```

The script forwards each entity inside data to the central broker unchanged — exactly as required by NGSI-LD.

This example is essential because it shows, in a concrete and executable form, how the federation bridge processes real NGSI-LD notifications generated by a local Orion-LD instance. It demonstrates how the

Registration-to-Entities service automatically obtains authentication credentials from Keycloak, ensuring secure and authorised communication with the central platform. The example also illustrates the synchronisation of entities between decentralised factory-level deployments and the central CIRCULOOS environment, confirming that updates are forwarded reliably and consistently. Since the implementation is already functional, production-ready, and reproducible by any project partner, it provides the clearest and most accurate reference for understanding interoperability within the CIRCULOOS Data Platform.

6 End-to-End federation scenario: Practical demonstration

To illustrate how the CIRCULOOS federation mechanism operates in practice, this section presents a concise example using the components and logic implemented in the project's open-source repository. This scenario demonstrates how a locally created entity at factory level is automatically propagated to the central CIRCULOOS platform through the Registration-to-Entities service, making the data immediately accessible to higher-level tools such as GRETA, the Digital Twin, or the Supply Chain Orchestrator.

6.1 Step-by-step scenario

1. A factory registers a new product entity locally

A local Orion-LD instance running at the factory receives a new NGSI-LD entity, for example representing a leather material. This entity is created through a standard NGSI-LD POST /entityOperations/upsert request, following the structure defined in the CIRCULOOS Common Data Model.

2. The local Orion-LD triggers a subscription notification

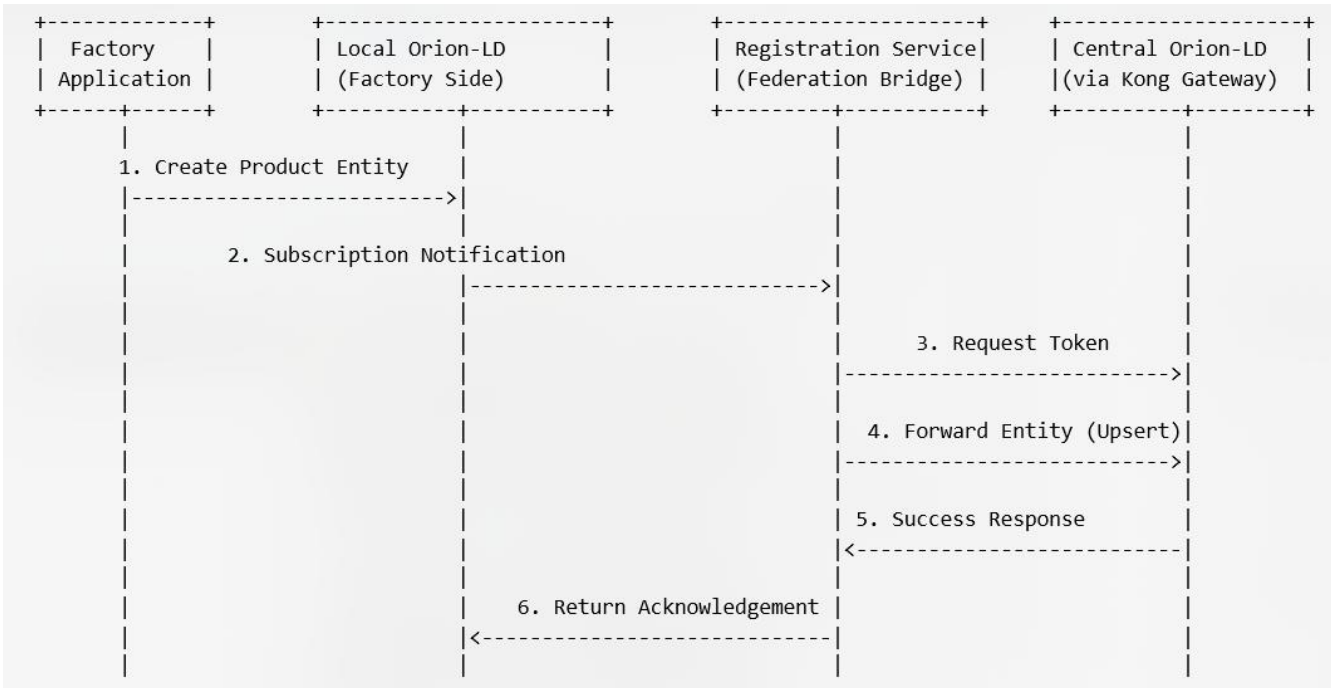
Because the factory has previously configured a federation subscription (as described in Readme_federation.md), the local Orion-LD automatically sends an NGSI-LD notification to the Registration-to-Entities service upon creation or update of the entity.

3. The Registration Service authenticates and forwards the entity

The Registration-to-Entities Python service receives the notification, requests an access token from Keycloak, and forwards the entity to the central Orion-LD through the Kong API gateway. This ensures secure, authenticated, and traceable communication.

4. The entity becomes available to central CIRCULOOS tools

After successful upsert into the central Orion-LD, the product entity becomes immediately accessible to all tools and services connected to the platform, including GRETA, SCDT, SCOPT, and the marketplace logic. These tools can now retrieve, process, or visualise the shared entity based on their functional needs.



```
+-------------+       +-----------------------+       +---------------------+    +--------------------+
|  Factory    |       | Local Orion-LD        |       | Registration Service|    | Central Orion-LD   |
| Application |       | (Factory Side)        |       | (Federation Bridge) |    |(via Kong Gateway)  |
+------+------+       +-----------+-----------+       +---------+-----------+    +---------+----------+
       |                          |                             |                          |
      1. Create Product Entity    |                             |                          |
       |------------------------->|                             |                          |
       |                          |                             |                          |
       |       2. Subscription Notification                     |                          |
       |                          |---------------------------->|                          |
       |                          |                             |                          |
       |                          |                             |     3. Request Token     |
       |                          |                             |------------------------->|
       |                          |                             |                          |
       |                          |                             |  4. Forward Entity (Upsert)|
       |                          |                             |------------------------->|
       |                          |                             |                          |
       |                          |                             | 5. Success Response      |
       |                          |                             |<-------------------------|
       |                          |                             |                          |
       |                          |     6. Return Acknowledgement |                        |
       |                          |<----------------------------|                          |
       |                          |                             |                          |
```

Figure 4 ASCII UML Sequence Diagram – Federation Scenario

The above ASCII diagram (Figure 4 ASCII UML Sequence Diagram – Federation Scenario) summarises the interaction between the local factory environment and the central CIRCULOOS platform during this federation scenario.

7 Conclusion

This deliverable described how the CIRCULOOS Data platform now supports a federated deployment model based on NGSI-LD and FIWARE components, using only the mechanisms and artifact that are implemented and published in the project's GitHub repository.

Starting from the architectural concept of federation, the document has:

- Clarified the roles of local and central Orion-LD instances and how they co-exist within the CIRCULOOS ecosystem.
- Presented the Registration-to-Entities Service as the core bridge that receives NGSI-LD notifications from local brokers and forwards entities to the central platform.
- Explained how Keycloak and Kong jointly provide authentication, authorisation, and secure routing for all federated interactions.
- Described the end-to-end workflow from local entity creation, through subscription and notification, to secure upsert in the central broker.
- Provided concrete, reproducible examples directly based on the registrationToEntities.py implementation and associated configuration files (Docker Compose, federation readme, example subscriptions).

The result is a practical, implementable federation pattern that CIRCULOOS partners can adopt to connect their local installations to the central platform while preserving control over their operational data. The approach remains fully compliant with the NGSI-LD standard and leverages existing FIWARE Generic Enablers, ensuring openness, interoperability, and long-term sustainability.

The federation mechanism will be further validated and extended across pilots, open calls and tools:

- More factories will be onboarded using the federation pattern,
- Additional entity types and use cases will be synchronised, and
- The integration with higher-level services (such as GRETA, SCDT, SCOPT and marketplace logic) will be refined and documented in subsequent technical and demonstration deliverables.

Through this work, CIRCULOOS progresses towards a distributed, secure and collaborative data space for circular manufacturing, where local autonomy and cross-pilot interoperability co-exist within a single, coherent NGSI-LD–based architecture.

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101092295. The herewith information reflects only the author's view. The European Commission is not responsible for any use that may be made of the information herewith included.