Towards interconnected dataspaces: implementing decentralised identity management via DAPS

Panos Protopapas[1]

[1]Inlecom Innovation, Tatoiou 11, 1st floor, Kifissia, 14561 Athens, Greece

Abstract

Data sovereignty is a fundamental principle of the dataspace framework, supported by both centralized and decentralized identity management mechanisms.[1] However, nearly all existing open-source dataspace connector implementations exclusively support centralized identity management[2], presenting significant challenges for cross-dataspace interoperability. This limitation forces organizations to either isolate their dataspaces or compromise their security by relying on an external authority controlling the single Dynamic Attribute Provisioning Service (DAPS). The sole exception, the Eclipse Dataspace Connector (EDC), supports a decentralized approach[3] but serves as a toolbox for building customized distributions rather than a ready-to-use connector implementation.[4] While discussions about addressing this issue through Decentralized Data-Sharing (DDS) in *Dataspaces 4.0* are underway, adoption remains in its early stages.[5]

This paper presents a proof-of-concept extension to the Sovity Connector, addressing the rigid identity management limitations present in all open-source dataspace connector implementations. We propose a federated dataspace architecture that enables connectors to operate across multiple DAPS without compromising the principle of data sovereignty, while -most importantly- exclusively utilizing currently available software. Our approach extends the Sovity Connector to dynamically trust multiple DAPS, facilitate interoperability across independently managed dataspaces, and enforce strict issuer verification and identity-claim integrity, enabling seamless cross-dataspace collaboration.

Keywords

federated dataspaces, federated identity management, sovity connector

This paper addresses (Issue #11) of the W3C Dataspaces Community Group.

1. Introduction

The dataspace framework is built upon the core principle of data sovereignty, allowing participants full control over their data assets by enabling them to choose who they would like to share their assets with. For the needs of the DISCO project, a Horizon Europe-funded initiative (grant agreement No 101103954) aiming to address the fragmented landscape of data sharing within the urban freight industry, enabling seamless, secure, and sovereign data exchange among the various stakeholders, the dataspace framework was quickly identified as the means to achieve this. To implement this, the Sovity Connector was chosen as the foundational technology for the Urban Freight Data Space (UFDS), due to being built upon the Eclipse Dataspace Connector (EDC), having an open-source tier under an Apache 2.0 license, and being mature and straightforward to deploy and use.

Unfortunately, a critical limitation of the Sovity Connector related to data sovereignty soon became evident; it only supports centralised identity management forcing all participants to authenticate using a single Dynamic Attribute Provisioning Service (DAPS) in charge of the whole dataspace. This limitation, shared among all open-source and ready-to-use dataspace connector implementations presented a substantial challenge, (i) limit communications to connectors of a single dataspace, or (ii) substantially increase our development effort and build a customised connector distribution based on EDC which supports decentrilised identifiers (DIDs).

Due to effort constrains, we were unable to build a customised connector. Consequently, some of our partners in the DISCO project who had already been operating, or were in the process of establishing

The Third International Workshop on Semantics in Dataspaces, co-located with the Extended Semantic Web Conference, June 01, 2025, Portorož, Slovenia

panosprotopapas84@gmail.com (P. Protopapas)

0000-0002-4959-4183 (P. Protopapas)

their own dataspaces, spanning various of their departments or partners of them outside our project, were now facing a dilemma: (i) either treat the UFDS as a completely independent and isolated dataspace from the rest of their dataspace infrastructure or, (ii) to achieve interoperability, unify them under a common DAPS with all other UFDS participants. Reasonably, the latter approach raised substantial security concerns, as it required entrusting identity management to an external organization, thereby substantially increasing the risk of a data breach.

In this paper we present our solution to the above issue, a proof-of-concept extension to the Sovity Connector aimed at allowing for identity decentralisation by allowing connectors to operate on a dataspace architecture comprising of multiple DAPS without compromises on their data sovereignty. Essentially, our approach allows for the connection between different dataspaces and the creation of a dataspace federation, and -most importantly- achieving this with software that is open-source and available today.

The remainder of this paper is structured as follows. Section 2, elaborates on the problem, detailing the architectural limitations of the Sovity Connector. Section 3, presents the design of our proposed solution, explaining how it addresses the identified limitations. Finally, Section 4 concludes the paper.

2. Limitations of the Sovity Connector

Sovity provides an easy-to-deploy software suite, under an Apache 2.0 license, making it easy to deploy a dataspace. Specifically, (i) the Sovity Connector community edition serves as the core component for enabling data exchange within a dataspace, and (ii) *sovity-daps*, a Keycloak-based implementation of DAPS. When deployed together, DAPS manages dataspace participants and enforces fine-grained, participant-based data sovereignty. To put it simply, this allows each connector in the dataspace to maintain its own access control list (ACL) over their data assets.

We proceed by presenting, in broad terms, the procedure establishing trusted connections between Sovity connectors, based on DAPS issued *JSON Web Tokens*[1] (JWT) and verified via the *JSON Web Key Set URI*[2] (jwks-uri). Let C_x and C_y be two arbitrary connectors within a dataspace.

1. During startup, both C_x and C_y contact the DAPS's jwks-uri to obtain the public keys needed for verifying DAPS-issued JWT tokens.
2. Before initiating interaction with C_y, C_x authenticates with DAPS and receives a JWT token certifying the token bearer to be C_x via the `client_id` claim.
3. C_x initiates a request to C_y including this token in the headers.
4. Upon receiving the token, C_y verifies its signature matches with one of the public keys acquired from the DAPS during startup.
5. If verified, C_y proceeds with the request, trusting the requester to be C_x as indicated by the `client_id` claim in the received token.

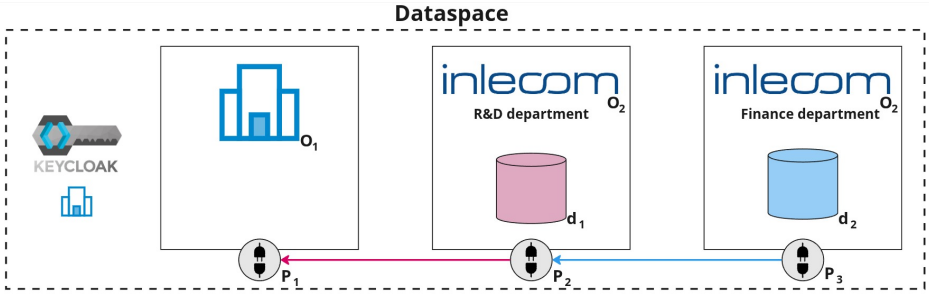

Figure 1: Single dataspace architecture's data sovereignty issues.

[1]JSON formatted tokens commonly used for authentication and authorization to securely transmit information.
[2]URL endpoint providing a set of public keys, allowing clients to verify signatures of JWTs issued by the DAPS.

As mentioned in Section 1, the Sovity Connector only supports a *single* dataspace architecture, that is, one where all connectors rely on the same DAPS for authentication and communication. While this architecture is suitable for situations where all participants fully trust the participant operating the DAPS, it introduces considerable security concerns in situations were this is not the case.

Figure 1 illustrates a minimal example of such a situation. Specifically, it presents a dataspace with three participants (P_1, P_2, P_3). P_1 represents Organisation 1, O_1, a *Data Consumer* of the dataspace and also acting as the DAPS operator. P_2 and P_3 represent different divisions of Organisation 2, O_2, both acting as *Data Producers* by providing a single data asset each. Specifically, P_2, owning asset d_1, allows P_1 access to it. On the other hand, P_3, owning asset d_2, allows P_2 access to it but not P_1.

The security concern in this architecture arises from the fact that under a centralised single DAPS setup, O_2 must trust O_1 to safeguard d_2, while O_1, by being in control of the DAPS, can have P_1 impersonate P_2 through a falsified JWT token and gain unauthorized access to d_2.

3. Proposed federated dataspace architecture

Briefly put, the solution we propose allows the Sovity Connector to dynamically trust multiple DAPS without compromising data sovereignty works as follows. All connectors participating in the federation can trust one or more DAPS. Specifically, each connector designates exactly one DAPS as *primary* from which it requests JWT tokens before interacting with other connectors. Furthermore, any number of DAPS can be set as *secondary*, implying the connector will trust JWT tokens signed by them. Finally, a connector's identity within the federation is partly determined by its primary DAPS.

A high-level illustration of how the issues discussed in Section 2 and illustrated in Figure 1 can be addressed by our proposed architecture is shown in Figure 2, where (i) in addition to the DAPS controlled by O_1, Data Space DAPS 1 (DSD_1), O_2 has deployed and is controlling another DAPS, Data Space DAPS 2 (DSD_2); (ii) the identities of all 3 participants partly depend on their primary DAPS (DSD_1:P_1, DSD_2:P_2, DSD_2:P_3), and (iii) DSD_1:P_1 (respectively, DSD_2:P_2) in addition to setting DSD_1 (respectively, DSD_2) as their primary DAPS have also designated DSD_2 (respectively, DSD_1) as a secondary DAPS.

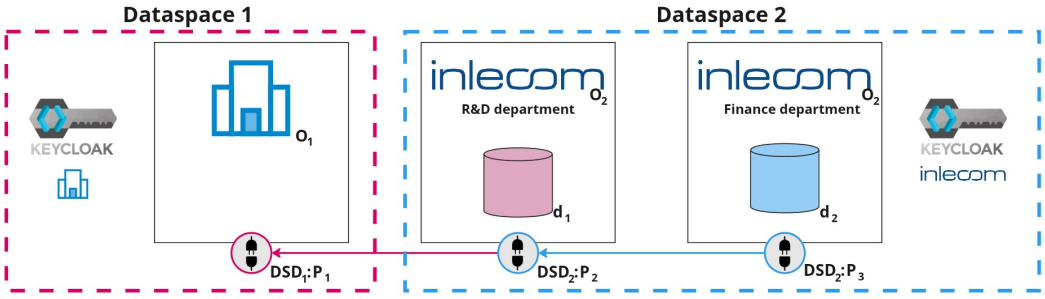

Figure 2: Proposed federated dataspace architecture.

To implement our proposed architecture, we utilised two main components: (i) an *API Gateway*, implemented via Apache APISIX, intercepting incoming requests targeting Sovity Connector *data space protocols's API URL* (DSP-URL) endpoint and verifying them through (ii) the *Federation Plugin*, implemented via a Python-FastAPI web-server, being also responsible for managing the delivery of public keys to connectors from all DAPS on the federated dataspace.

Figure 3 illustrates the two modifications made to the connector's operational procedure: the initialisation process (steps 1 and 2), and the handling of incoming requests from external networks at the DSP-URL (steps 3-6). These modifications, discussed below, form the core of our technical contribution and are essential in enabling the proposed federated dataspace architecture.

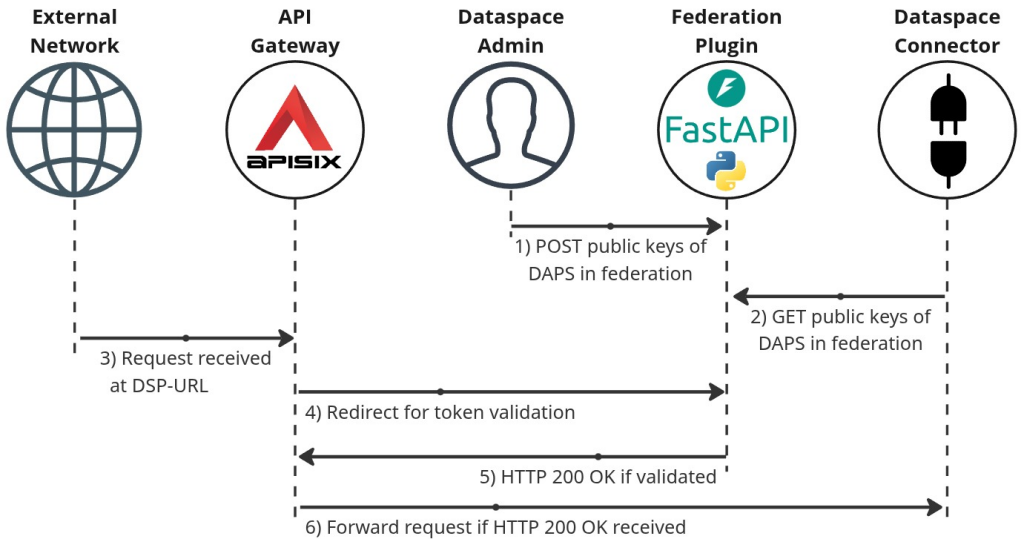

Figure 3: Modified connector operation procedure for federated dataspace architecture.

3.1. Initialization Process

The first modification concerns the handling of public keys received and stored by Sovity's connector during its startup process, modifying Step 1 of the procedure described in Section 2.

Instead of directly contacting the DAPS's jwks-uri to acquire the public keys when starting up, the connector is set to contact the *get-keys* endpoint of the Federation Plugin[3], which must be preloaded by an administrator with the public keys of all DAPS, using its *load-keys* endpoint, that the connectors using the plugin can set as their primary and secondary DAPS. During the preloading process, the administrator specifies for each loaded public key a *key identifier*. This identifier matches the public key's corresponding *DAPS realm URI*, that is, the value used to populate the `iss` claim in the tokens issued by this DAPS.

Furthermore, it should be noted that the Federation Plugin also allows for selective retrieval of public keys through query parameters, enabling connectors to obtain only the keys relevant to their operation.

3.2. Incoming Request Validation Process

The second modification concerns the handling of incoming requests targeting the connector's DSP-URL. These requests are intercepted by the API Gateway and forwarded to the Federation Plugin for validation; only validated requests are forwarded by the API Gateway to the connector.

The validation process conducted by the Federation Plugin is more comprehensive than the one described in the procedure of Section 2. Specifically, the process ensuring that no DAPS x within the federation can engage in token manipulation and falsely represent a connector with primary DAPS y is as follows:

1. The received token's signature is verified against the public keys stored by the Federation Plugin (i.e., those previously loaded via the *load-keys* endpoint).
2. If the token is verified, the key identifier of the public key used for verification is compared with the value of the `iss` claim in the received token. This ensures a malicious DAPS in the federation cannot impersonate another DAPS.
3. If the issuer is verified, the token's `client_id` is checked to ensure it follows the naming convention, requiring the format `<iss-claim-value>[SEP]<id-not-containing-[SEP]>`. This ensures a malicious DAPS in the federation cannot provide a claim impersonating a connector with a different primary DAPS.

[3]Achieved by simply changing the connector's EDC_OAUTH_PROVIDER_JWKS_URL environment variable's value.

4. If the `client_id` claim's format is valid, an `HTTP 200 OK` is returned (to the API Gateway).

It should be noted that this enforced naming convention is only a workaround for the Sovity Connector's limitation to track which connector "belongs" to which DAPS, in order to prevent federation-wide JWT manipulation. Moreover, while this enforced naming convention could be relaxed through the use of a configurable dictionary or a similar mechanism, we believe that the appropriate next step in the development process, should be a production-ready solution implementing this functionality as an EDC extension on a Sovity Connector fork. Such an approach would not only address the limitation more robustly but also result in a cleaner and more easily deployable solution.

On a second note, although a malicious DAPS operator could still manipulate JWTs and successfully impersonate a connector for which it acts as the primary DAPS, this security concern is not fundamentally different from the risk present in a single dataspace architecture with centralized identity management. In the former case, participants who choose to share data with connectors associated with a given DAPS implicitly accept the risk that the DAPS operator may be malicious, while in the latter case all participants accept the security risk of the (single) DAPS operator being malicious. Notably, in the former case, if the malicious DAPS is a secondary DAPS for a connector, the connector's administrator can mitigate the threat by removing the DAPS's public keys from the Federation Plugin and restarting the connector. In contrast, in a centralized identity management architecture, of if the malicious DAPS is the primary one, the only option is to shut down the connector entirely.

3.3. Further remarks

First, it is evident that trust among participants must be reciprocal. That is, for connector C_x with primary DAPS DS_1 to interact with connector C_y with primary DAPS DS_2, C_x must designate DS_2 as a secondary DAPS, and vice versa.

Furthermore and as mentioned previously, the Federation Plugin must be preloaded with the public keys of the DAPS that connectors utilising this plugin can set as primary or secondary DAPS. We assume this process to be performed manually by a dataspace administrator who retrieves and loads these keys via the API, and updates them when changes occur. Clearly, a more automated solution can be implemented assuming all DAPS' jwks-uri endpoints are publicly accessible for the public keys to be retrievable.

Finally, the codebase of the Federation Plugin has been open-sourced and is available at https://gitlab.com/disco-horizon-europe/open-software-repository/data-space-federation.

4. Conclusion

This paper addresses a critical architectural limitation present in the Sovity Connector and all other readily available open-source connector implementations. The proposed solution enables the deployment of dataspace architectures with multiple DAPS without compromising data sovereignty. Most importantly, it leverages existing open-source software to achieve a setup that aligns, to a large extent, with the benefits of Decentralized Data-Sharing advocated by Dataspaces 4.0 and DID-based decentralized identity management supported by EDC.

Acknowledgments

This work was supported by the European Union's Horizon Europe research and innovation programme under grant agreement No. 101103954.

Declaration on Generative AI

During the writing of this paper, the author used GPT-4o in order to enhance the clarity and readability of the text. After using this tool, the author reviewed and edited the content as needed to achieve the

final version, taking full responsibility for the publication's content.

References

[1] V. Siska, V. Karagiannis, M. Drobics, et al., Building a dataspace: Technical overview, Gaia-X Hub Austria (2023).

[2] T. Dam, L. D. Klausner, S. Neumaier, T. Priebe, A survey of dataspace connector implementations, arXiv preprint arXiv:2309.11282 (2023).

[3] P. Koen, Eclipse dataspace connector - overview deck, 2022. URL: https://github.com/eclipse-edc/Collateral/blob/e61262008772f2a2aaaf303cf6944e3921be3230/Latest%20Presentations/2022-04-26%20Eclipse%20Dataspace%20Connector%20-%20Overview%20Deck.pdf.

[4] E. D. C. Contributors, Edc base documentation, 2025. URL: https://eclipse-edc.github.io/documentation/#what-edc-is-not.

[5] S. H. Alsamhi, A. Hawbani, S. Kumar, M. Timilsina, M. Al-Qatf, R. Haque, F. M. A. Nashwan, L. Zhao, E. Curry, Empowering dataspace 4.0: Unveiling promise of decentralized data-sharing, IEEE Access 12 (2024) 112637–112658. doi:10.1109/ACCESS.2024.3442968.