Technologies for Manufacturing as a Service Ecosystems

Deliverable 3.7

Tec4MaaSEs governance services v1

WP3: DT Modelling, Operation and Governance for resilient value networks

Editor:	Eleni Gkinou
Lead beneficiary:	MAG
Version:	1.0
Status:	Final
Delivery date:	30/06/2025
Dissemination level:	PU (Public)

This project has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101138517.

Deliverable Factsheet

Grant Agreement No.	101138517
Project Acronym	Tec4MaaSEs
Project Title	Technologies for Manufacturing as a Service Ecosystems
Start date	01/01/2024
Duration	36 months

Deliverable Name	D3.7 Tec4MaaSEs governance services v1
Related WP	WP3 DT Modelling, Operation and Governance for resilient value networks
Due Date	30/06/2025

Author	Eleni Gkinou (MAG)
Contributor(s)	Kostas Kalaboukas, Eleni Kartsonaki, Dimitis Verlekis (MAG), Gonzalo Gil, Ignacio Lazaro, Maite Fernandez, Patricia Casla (TEK)
Reviewer(s)	Sadra Shoarinejad (ARC), İsa Aksu (KAREL)
Approved by	All partners

Document History

Version	Date	Author(s)	Organisation	Description
0.1	09/05/2025	Eleni Gkinou	MAG	ToC
0.2	13/06/2025	Gonzalo Gil, Ignacio Lazaro, Maite Fernandez, Patricia Casla	Tekniker	Contributions from partner
0.3	25/06/2025	Eleni Gkinou, Kostas Kalaboukas, Eleni Kartsonaki, Dimitris Verlekis	MAG	Complete draft ready for internal review
0.4	26/06/2025	Sadra Shoarinejad, İsa Aksu	ARC, KAREL	Peer review
0.5	26/06/2025	Eleni Gkinou	MAG	Integration of comments from internal review
0.6	30/06/2025	Armend Duzha	MAG	Quality check
1.0	30/06/2025	Eleni Gkinou	MAG	Final version ready for submission

Executive Summary

In the context of a MaaS, there are various things that might affect the whole request, negotiation and agreement phases. Although a MaaS platform can provide the necessary functionalities for (auto-) search, optimized pairs of consumers/providers, there are aspects that unless they have been properly addressed may risk the whole process.

A pre-defined set of conditions, rules and flows on how each partner can support a MaaS operation need to be considered alongside different particularities per case. This refers to the necessary business arrangements for the exchange of information and multi-step discussions on different aspects of such a collaboration. This is referred to as business governance services, i.e. the necessary functionalities to support different forms of negotiation and exchange of information.

On the level of information, we need also data governance services in line with the IDSA specifications that will ensure a secure and trust environment, in which information flows among different actors.

Last in a MaaS context there are algorithms/ models that propose new partners, make recommendations and provide some suggestions. It is very important for the end users to have trust in those models, and this can be done only if there is a clear explanation of how the algorithm works, under which conditions have been trained and under which assumptions any recommendation was made. Therefore, the role of AI models' governance is important to build trust in the platform and to create the best value for the consumers/providers.

D3.7 tackles the above governance pillars. It provides a set of services for business negotiations, data spaces and AI governance and how they are incorporated into the whole Tec4MaaSEs solution. For each of the services, it describes the approach used and presents the current implementation results.

It must be noted that the Tec4MaaSEs has an iterative approach in the implementation and this deliverable presents the developments made in the first iteration (till M18). A second version will follow (D3.8) with the final services descriptions and the additions during the second iteration (M19-M26)

Table of Contents

List of Figures

List of Tables

Acronyms and Abbreviations

Acronym	Description
AAS	Asset Administration Shell
DCAT-AP	Data Catalog Vocabulary – Application Profile
DGF	Data Governance Framework
DSC	Data Space Connector
DSP	Dataspace Protocol
DSSC	Data Spaces Support Centre
EDC	Eclipse Dataspace Connector
IDSA	International Data Spaces Association
MaaS	Manufacturing-as-a-Service
TCK	Technology Compatibility Kit
TDC	Tekniker Dataspace Connector
T4M	Tec4MaaSEs
UI	User interface

1 Introduction

1.1 Purpose and Scope

The purpose of D3.7 is to deliver the necessary services that implements the governance framework described in deliverable D2.3.

More particularly, the governance services consist of three main pillars:

- The business governance services: those services that support the negotiation and agreement of consumers/providers in the context of a MaaS.
- Data governance services: data spaces implementation responsible for trusted and secure exchange of data between different T4M components and collaborators.
- AI governance services: Services that provide the necessary explainability to the end users about the use of AI services developed within the T4M project/ platform.

D3.7 described the first iteration of the services implementation, which is a work in progress. The final version (D3.8) with the full services description will be delivered on M26.

1.2 Relation with other deliverables

D3.7 takes as input the governance framework defined in D2.3. It is also linked with the architecture deliverable (D2.5) with regards to the positioning in the whole system and the services' expected functionalities.

Last it encapsulates all business and operational concepts defined in the reference cases/scenarios described in D2.1.

All services will be finally integrated within the T4M solution under WP4, mainly in deliverables D4.2 and D4.3

1.3 Structure of the document

The document is structured as follows:

- **Section 2** describes the concept of business governance, presenting the problem context, the proposed approach, the ongoing implementation within the MIRA platform, and the challenges and lessons learned.
- **Section 3** introduces the Data Governance, describing the approach used to manage and share data consistently and securely. It also outlines progress in implementing these solutions across key components.
- **Section 4** presents the proposed approach for managing AI model documentation in the MIRA platform, the implementation of the AI Model feature, and key challenges and lessons learned.

2 Business Governance

2.1 Introduction (Problem Overview)

Business governance in a Manufacturing as a Service (MaaS) context involves the processes and actions needed for organizations to reach an agreement within a value network. This includes how resources are published in a marketplace, how collaboration is initiated, and how terms are negotiated and agreed on between the involved parties.

In practice, providers and consumers must not only discover potential partners but also evaluate whether their offerings and requirements align. This involves structured negotiations, the exchange of service requests and offers, configuration of access policies, and sharing of supporting documents, or other operational details. These steps require coordination and ongoing updates to ensure mutual understanding as the negotiation progresses.

One of the key challenges is achieving consistency in how these interactions are handled across different organizations, each of which may operate with different systems, procedures, and terminology. Without a clear and shared governance framework, collaboration efforts can become inefficient, reducing trust and slowing down the overall process.

This section presents the business governance functionalities developed within the project. While the full methodology is provided in Deliverable D3.1, key components are included also here to offer a complete view of how the approach supports governance workflows tailored to the different pilots and value networks within the project.

2.2 Proposed Approach

The proposed business governance approach builds upon the foundational concepts developed in the Mira platform, enhancing them to support structured interactions across diverse value networks. While the core elements for modelling resources and supporting collaboration have been retained, new features have been introduced to align with industry standards and meet the specific needs of negotiation processes within MaaS environments.

The first part of this approach is a messaging system integrated into the collaboration environment. This system enables users to communicate through messages, attach documents, and exchange structured templates that contain fields selected from AAS submodels. This system supports the different negotiation loops by enabling structured exchanges of information, such as service requests, offers, and clarifications, while ensuring consistency throughout the negotiation process.

To further support coordination, the approach includes a customizable negotiation status feature. This allows users to define and manage their own set of statuses that reflect the different phases of interaction, from initial negotiation and agreement to service execution and delivery. These statuses provide all involved parties with a consistent, real-time overview of progress at every stage, ensuring alignment throughout the entire collaboration process.

The combination of a messaging system and customizable status tracking enhance the efficiency and coordination of negotiation and MaaS service delivery processes.

2.3 Implementation

As part of the Tec4MaaSEs project, a key governance functionality has been implemented in the MIRA platform to support structured negotiation processes between organizations. This implementation focuses on the Messaging system, which plays a central role in enabling clear, traceable, and flexible communication across different stages of collaboration in a Manufacturing-as-a-Service (MaaS) environment.

The Messaging system is designed to facilitate the exchange of service requests, offers, and supporting information during the negotiation phase and potentially in post-agreement interactions. The initial implementation is accessible through each collaborator's profile page, where users can initiate communication by creating and sending messages to selected collaborators. As shown in Figure 1, the process begins with the user defining key information for the message, such as assigning a title and optionally linking it to a specific asset (resource). Once this setup is completed, the user is directed to the message composition interface, where the content of the message can be written and sent to the selected collaborator, as illustrated in Figure 2.

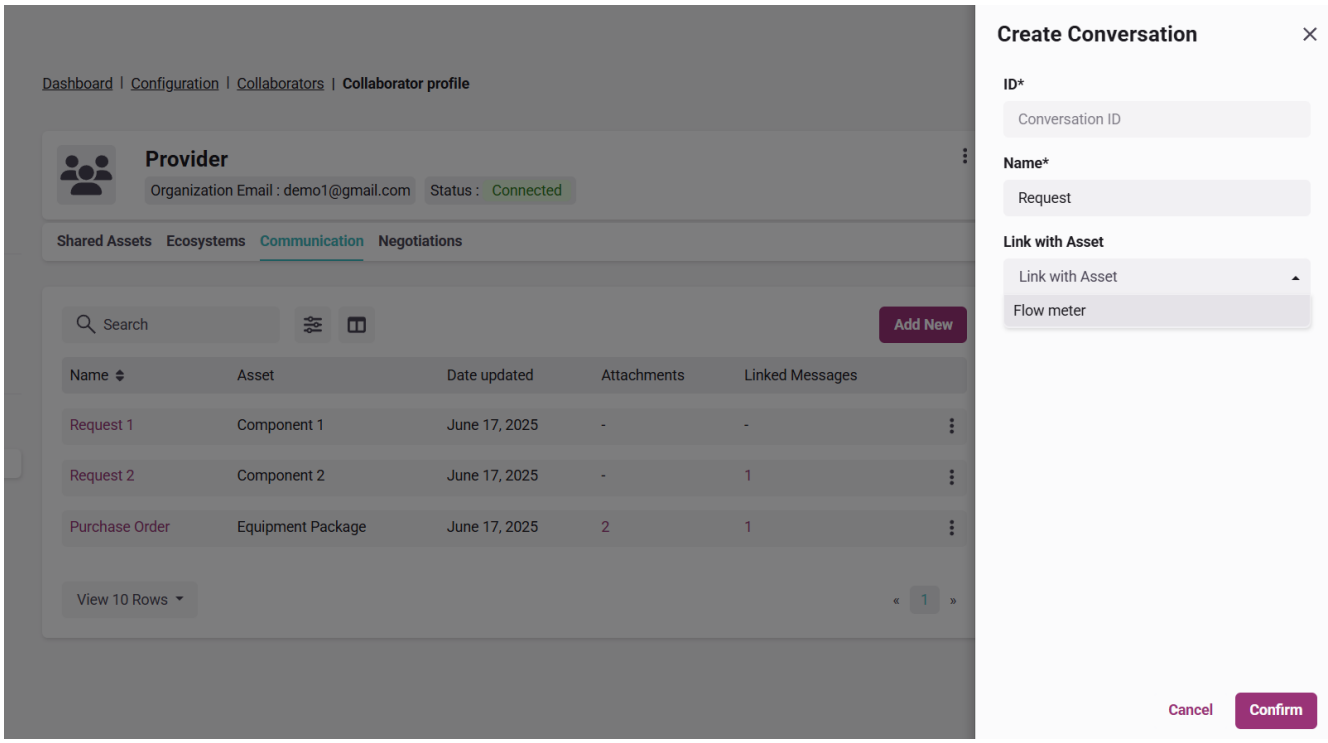


Figure 1: Message Creation

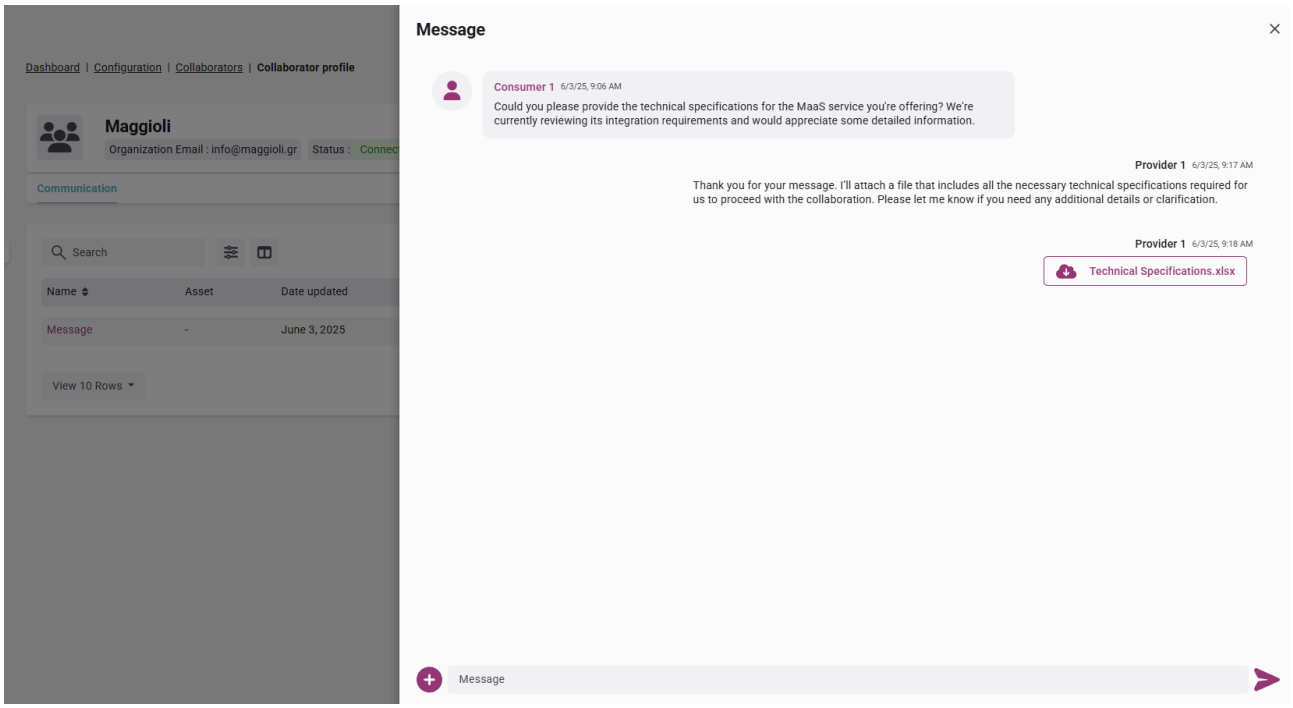


Figure 2: Interface for Composing and Sending a New Message

In addition to messages, users can send **predefined templates** that include structured fields derived from AAS submodels, such as the Purchase Request Notification Submodel. These templates standardize the information exchanged and ensure clarity during negotiations. In the first version, a Purchase Request Notification template has been included. This template allows users to enter values for predefined fields such as order quantity unit, price, and special treatment class, and send the completed request directly to the selected collaborator (see Figure 3 and Figure 4).

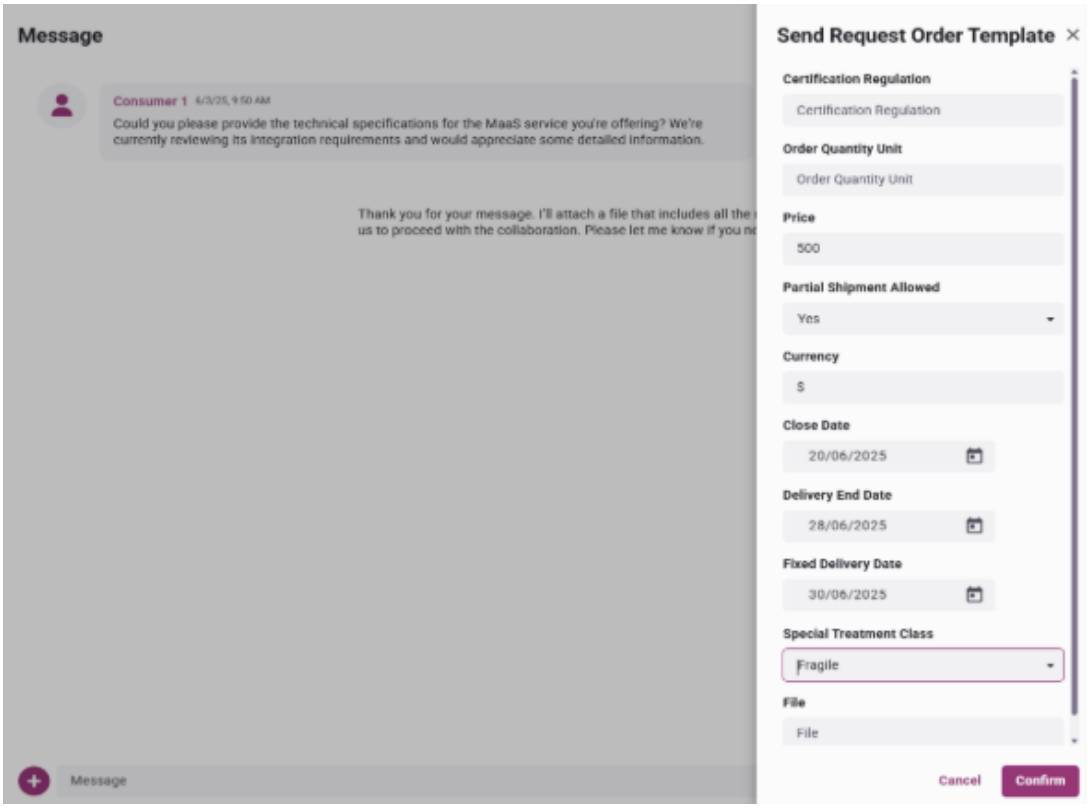


Figure 3: Creating the Request Order Template

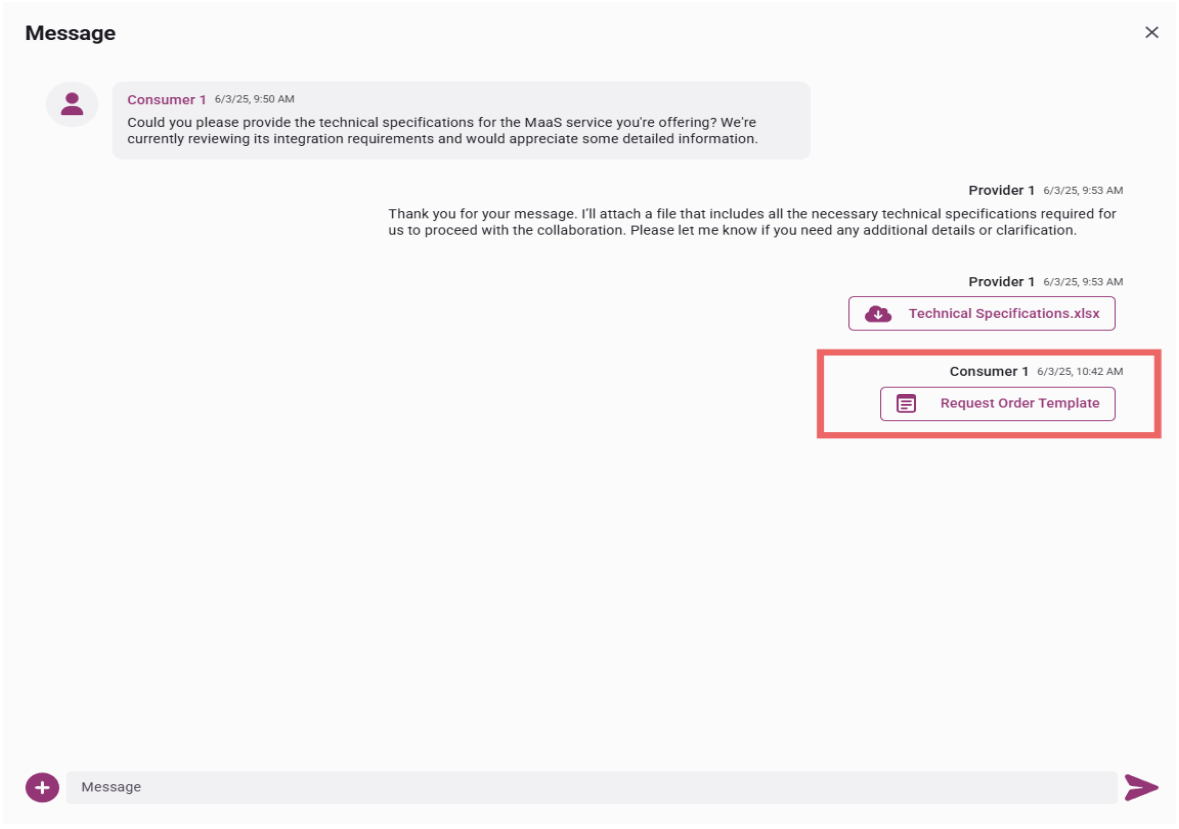


Figure 4: Template Message Exchange

As illustrated in Figure 5, after the message or template is sent, a notification appears to the user confirming that the message was successfully delivered. This notification not only informs the user of the successful action but also provides a direct link to view the message. By clicking it, users are redirected to the corresponding page and form where the sent message is displayed.

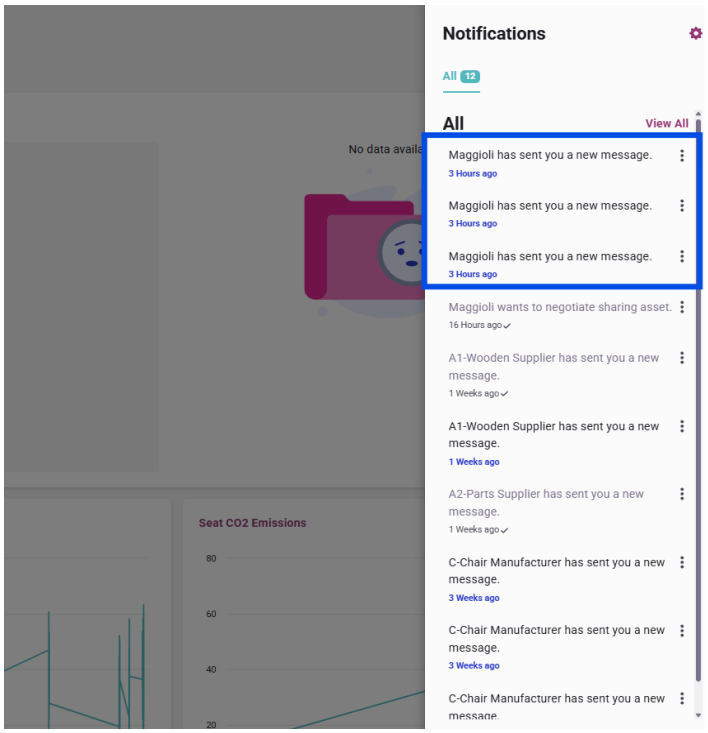


Figure 5: Message Notification

The messaging system was also created to support the attachment of files and documents to any message and to allow users to link related messages within the same conversation thread. These linked messages and attached documents are displayed in a dedicated column next to each message entry in the message list, improving context tracking and helping users maintain a clear and organized history of negotiations (see Figure 6).

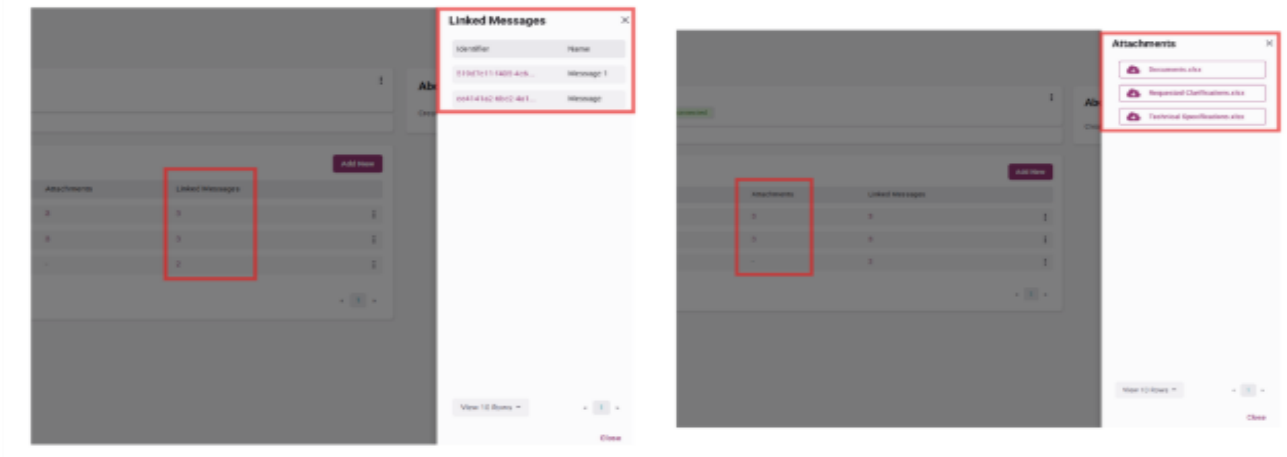


Figure 6: Attachments & Linked Messages in Messaging System

In future iterations, this functionality will be extended to allow message creation directly from the Shared Asset profile page, offering an alternative entry point for a resource or a service-specific communication. Users will also be able to choose from a wider selection of predefined templates based on various AAS submodels or create their own custom templates by selecting specific fields. This implementation will support a wider range of negotiation scenarios across different value networks.

2.4 Challenges and Lessons Learned

During the initial development of business governance functionalities, several challenges emerged. A major challenge was designing a negotiation process that could flexibly adapt to the diverse needs and requirements of each pilot scenario. Since each pilot had unique processes and interaction models, creating a flexible solution that could work across all use cases proved difficult.

Another key challenge involved the inconsistency in statuses used by various organizations. For instance, while one organization might use the term “order closed,” another might refer to the same stage as “order delivered.” These variations highlighted the need for a negotiation approach that could be both generic and adaptable. To address this, customizable statuses will be incorporated within the negotiation workflows, enabling organizations to utilize terminology that aligns with their internal practices, while maintaining consistency throughout the value networks.

An additional challenge was designing a template system that could handle many different negotiation needs without becoming too complex or too rigid. Each use case may require different information, depending on the context and industry. To address this, the next version of the system will include a broader set of predefined templates based on Asset Administration Shell (AAS) submodels, while also allowing users to create their own templates by selecting the fields they need. The main challenge here is balancing standardization with flexibility—making sure the templates are structured and consistent, but still adaptable enough to support a wide range of use cases.

Several lessons were drawn from these experiences, with a strong focus on flexibility and reusability. Supporting different negotiation needs, terminology, and workflows across pilots required a system design that could easily adapt to a variety of contexts. This led to an iterative development approach for the MIRA platform, starting with simple, generic features such as free-text conversations. Over time, based on user feedback, more advanced capabilities were introduced, including customizable status labels, creation of additional tabs in system’s entities, and structured templates. These enhancements allow each value network to tailor the system to its specific needs, while still maintaining a consistent overall framework.

3 Data Governance

3.1 Introduction

3.1.1 Problem Overview

In a Manufacturing as a Service (MaaS) data space, data governance plays a vital role in enabling trust, collaboration, and efficiency among stakeholders who share sensitive industrial information. Since this ecosystem depends on exchanging valuable data—like product designs, production capabilities, and operational details—clear governance ensures that this exchange is secure, fair, and legally compliant.

A decentralized governance model is key, as it allows all participants to maintain control over their own data while still working together. This model supports open negotiation and collaboration without compromising the autonomy or trust of any party.

One of the most important principles is data sovereignty. Manufacturers and service providers must retain control over their proprietary information. A strong governance framework gives them the right to decide how, when, and by whom their data is accessed or used. Transparency and accountability are also critical. Participants need to know their data will be used responsibly, under clear policies. This assurance encourages more organizations to participate and share valuable data. Because much of the information exchanged is sensitive or protected by law (e.g., intellectual property or trade data) security and privacy must be maintained. Governance ensures all data sharing follows legal standards like or industry-specific regulations, protecting participants from misuse or unauthorized access. Another core aspect is fair value exchange. Since data in a MaaS space holds economic value governance ensures usage agreements are fair and transparent, promoting equity across the ecosystem. Lastly, interoperability is essential. Stakeholders operate on different platforms and systems, so governance must support smooth data integration and compatibility, making collaboration seamless.

In this environment, three main stakeholders interact:

1. **MaaS Consumers** are companies that need manufacturing services. They define product requirements, provide designs and forecasts, and depend on MaaS Providers to produce goods or parts.

2. **MaaS Providers** offer the actual manufacturing services and resources. They supply capabilities, labor, equipment, and flexible production options tailored to consumer needs.

3. **Platform Providers** operate the digital platforms that connect consumers and providers. They offer the infrastructure for data sharing, enforce governance rules, ensure security and legal compliance, and support functions like matchmaking and service optimization.

The Tec4MaaSEs Data Governance Framework (DGF) is structured into four key layers, each playing a vital role in building a secure, interoperable, and trusted environment for data exchange in a decentralized MaaS ecosystem. These layers are designed to flow from top-level governance principles down to operational data usage, creating a coherent system that supports collaboration and compliance.

- **Data Space Layer**: sets the core rules and principles, ensuring participants control their data and comply with frameworks like the International Data Spaces Association (IDSA)1 and Gaia-X2 through a strict onboarding process.
- **Use Case Layer**: defines specific data usage policies that control what actions can be performed on shared data, protecting sensitive information.
- **Data Offering Layer:** allows participants to share datasets under clear terms, emphasizing interoperability for smooth collaboration.
- **Data Usage Layer**: manages contract negotiations and enforces usage policies during data exchanges, with customized agreements based on participants' needs.

The Data Space Connector (DSC) serves as the central technical component linking all four layers of the Tec4MaaSEs Data Governance Framework by enforcing policies and managing data exchange throughout the MaaS ecosystem. At the Data Space Layer, it controls access by verifying participant identities and ensuring compliance with core rules. Moving to the Use Case Layer, the DSC enforces data usage policies that regulate which actions can be performed on data based on specific agreements. In the Data Offering Layer, it allows participants to securely publish and share datasets, managing both access and usage conditions. Finally, at the Data Usage Layer, the DSC oversees contract negotiations, enforces agreed terms during data transfers, and continuously monitors compliance.

The increasing adoption of Asset Administration Shells (AAS) as standardized digital representations of physical assets in Industry 4.0 has created a critical demand for secure, scalable, and interoperable data sharing mechanisms across organizations and systems while preserving data sovereignty and trust.

The primary efforts so far have focused on the Data Offering and Data Usage Layers. In the Data Offering Layer, the Tekniker Data Connector (TDC)[3] has been extended to simplify the creation of data offerings based on a given AAS model representing factory-level digital twins, and to enable secure communication with the MaaS platform. In the Data Usage Layer, work has focused on ensuring interoperability between the TDC and the creation of data offerings based on a given AAS model representing factory-level digital twins, and to enable secure communication with the MaaS platform. In the Data Usage Layer, work has focused on ensuring interoperability between the TDC and the Eclipse Dataspace Connector (EDC)[4], enabling seamless negotiation between different connectors.

3.1.2 State of the art

In February 2024, IDSA defined the Dataspace Protocol (DSP)[5], a set of protocols and schemes to facilitate data sharing between organizations governed by usage control. Based on the DSP 2024-1, currently, the Eclipse Foundation is working on CEN/CELEC and ISO standardization of the DSP[6]. This involves the definition of a DSP specification document, the development of a referent solution that implements it and the

[1] https://internationaldataspaces.org/
[2] https://gaia-x.eu/
[3] Tekniker Data Connector
[4] Eclipse Dataspace Components | projects.eclipse.org
[5] Dataspace Protocol 2024-1 | IDS Knowledge Base
[6] Dataspace Protocol 2025-1-RC2

development of a Technology Compatibility Kit (TCK) that will allow future solutions to validate their compatibility with the DSP.

Through the implementation of the DSP, Tekniker already developed the TDC, a reference implementation published in the Data Spaces Support Centre (DSSC) toolbox[7] that enables any company to publish datasets, negotiate policies and access agreed dataset in a standardised way.

The TDC will be used as the reference tool that any participant within the Tec4MaaSEs Data Space must use to share data from digital twins to different consumer APIs in an interoperable and sovereign manner. However, to do so, two limitations should be addressed: the expressiveness of datasets in the Data Offering Layer and the compatibility with the DSP in the Data Usage Layer.

On the one hand, Tec4MaaSEs proposes a two-fold approach to manage data from digital twins. While the AAS allows the management of the data from digital twins at a factory level, the TDC allows to manage how this data will be shared across the Tec4MaaSEs Data Space between factories and the MaaS platform. To enable the discovery of data from digital twins at factory level to the Tec4MaaSEs Data Space, it is of utmost importance to provide with the tools to easily describe AAS models as datasets in the TDC. Recent works [1] already propose an approach to automatically generate datasets from AAS models in an EDC. However, the lack of expressiveness of the EDC limited the information published in the datasets from the AAS models and, therefore, their discovery.

On the other hand, once data from digital twins is published from AAS as datasets in the TDC, it becomes discoverable, the usage policies can then be negotiated and if agreed, data can be accessed in a sovereign manner. To standardize these processes, the TDC must correctly implement the DSP. In this regard, although there is already a stable DSP specification document and a reference implementation such as EDC, there is no TCK published so far to validate compliance of emerging solutions such as the TDC and, therefore, the Tec4MaaSEs Data Space with the DSP.

3.2 Proposed Approach

Figure 7 represents the proposed approach to share data from digital twins to different consumer APIs in an interoperable and sovereign manner.

[7] https://toolbox.dssc.eu/

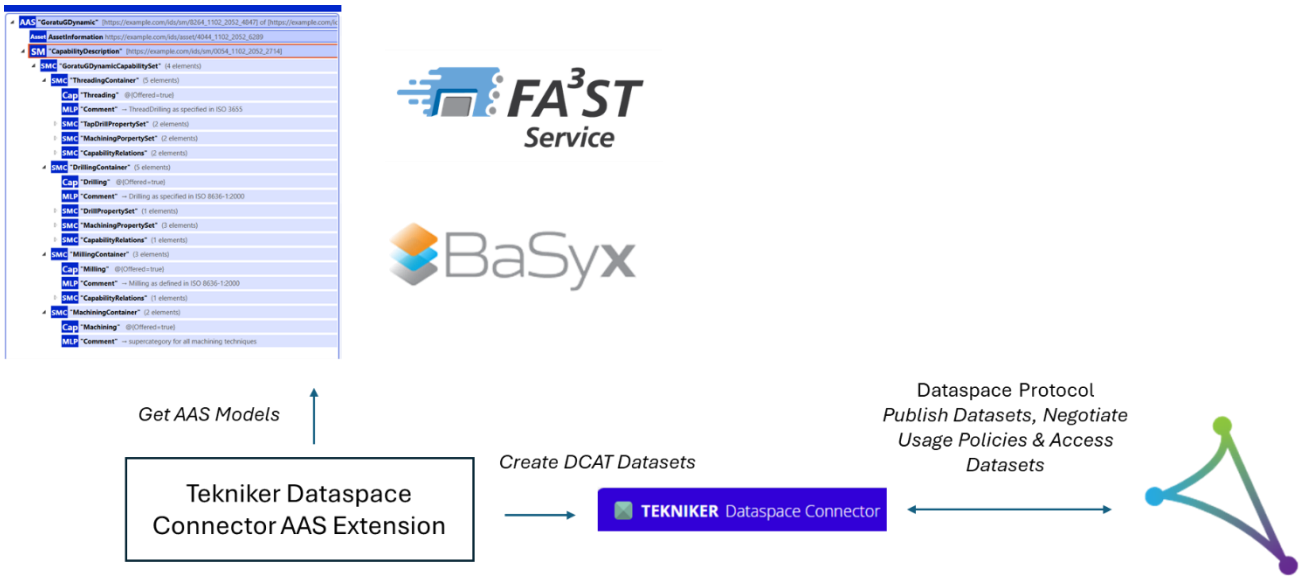


Figure 7: Proposed Approach for Interoperable and Sovereign Digital Twin Data Sharing – AAS extension for TDC

This approach involves four components: AAS Server, the TDC AAS Extension, the TDC and the EDC. They are described below:

- AAS Server: It represents the central hub for managing digital twins of physical or logical assets in an Industry 4.0 environment. It implements requests in compliance with the Specification of the Asset Administration Shell Part 2: Application Programming Interfaces[8] that should ensure compatibility with any AAS server adhering to this specification such as FA³ST Service[9] or Eclipse BaSyx[10].
- TDC AAS Extension: it is a tool developed in Tec4MaaSEs to automatically generate Data Space datasets from AAS models.
- TDC: it is a background component already developed by Tekniker to enable data management and data sharing based on the DSP.
- EDC: it is an open-source already framework developed by the Eclipse Community for sovereign, inter-organizational data sharing considered as the reference for the implementation of the DSP.

As previously stated in the state of the art, two limitations must be addressed to enable data sharing from digital twins to different consumer APIs through the Tec4MaaSEs Data Space: the expressiveness of datasets in the Data Offering Layer and the compatibility with the DSP in the Data Usage Layer. While we propose to address the former through the development of the Tekniker Dataspace Connector AAS Extension based on existing approaches, we believe that testing the interoperability between the TDC and the EDC is the best approach to ensure compatibility of the TDC with the DSP. The TDC AAS Extension and the TDC EDC compatibility test are further described below.

[8] https://industrialdigitaltwin.org/en/wp-content/uploads/sites/2/2025/06/IDTA_01002-25-01_AAS-Specification_Part2_API.pdf
[9] https://www.iosb.fraunhofer.de/en/projects-and-products/faaast-tools-digital-twins-asset-administration-shell-industrie40.html
[10] https://eclipse.dev/basyx/

3.2.1 Data Offering Layer: TDC AAS Extension

There is already work on the automatic generation of datasets from AAS models to DSC [1]. However, the current implementation of DSC only supports the default properties of a dataset, which in many cases is insufficient for providing a meaningful description. To help Data Consumers better understand what a dataset represents and the type of data it shares, existing implementations should be extended to offer greater expressiveness.

The DSP relies on the DCAT-AP information model to describe datasets and associated metadata between DSC. DCAT AP provides RDF classes and properties to allow datasets and data services to be described and included in a catalog as represented in Figure 8.

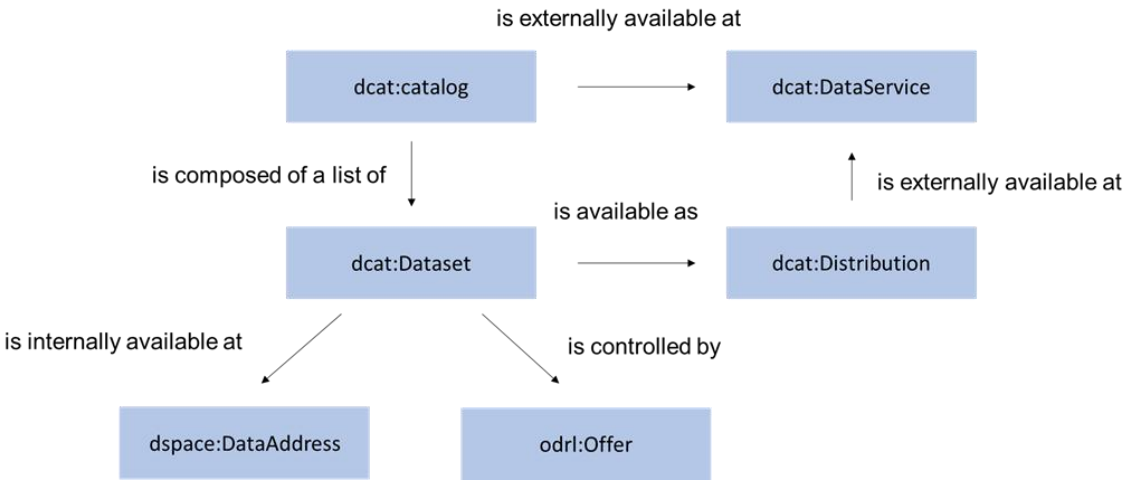


Figure 8: DCAT AP Catalog representation

To enhance expressiveness and improve the user experience when defining data offerings and sharing AAS-based digital twins, the first step was to analyse the *Specification of the Asset Administration Shell (AAS) Part 1*: *Metamodel*[11] represented in Figure 9 in order to identify which metadata from AAS shells, submodels, and submodel elements should be included in the catalog.

[11] https://industrialdigitaltwin.org/en/wp-content/uploads/sites/2/2025/06/IDTA_01001-25-01_AAS-Specification_Part1_Metamodel.pdf

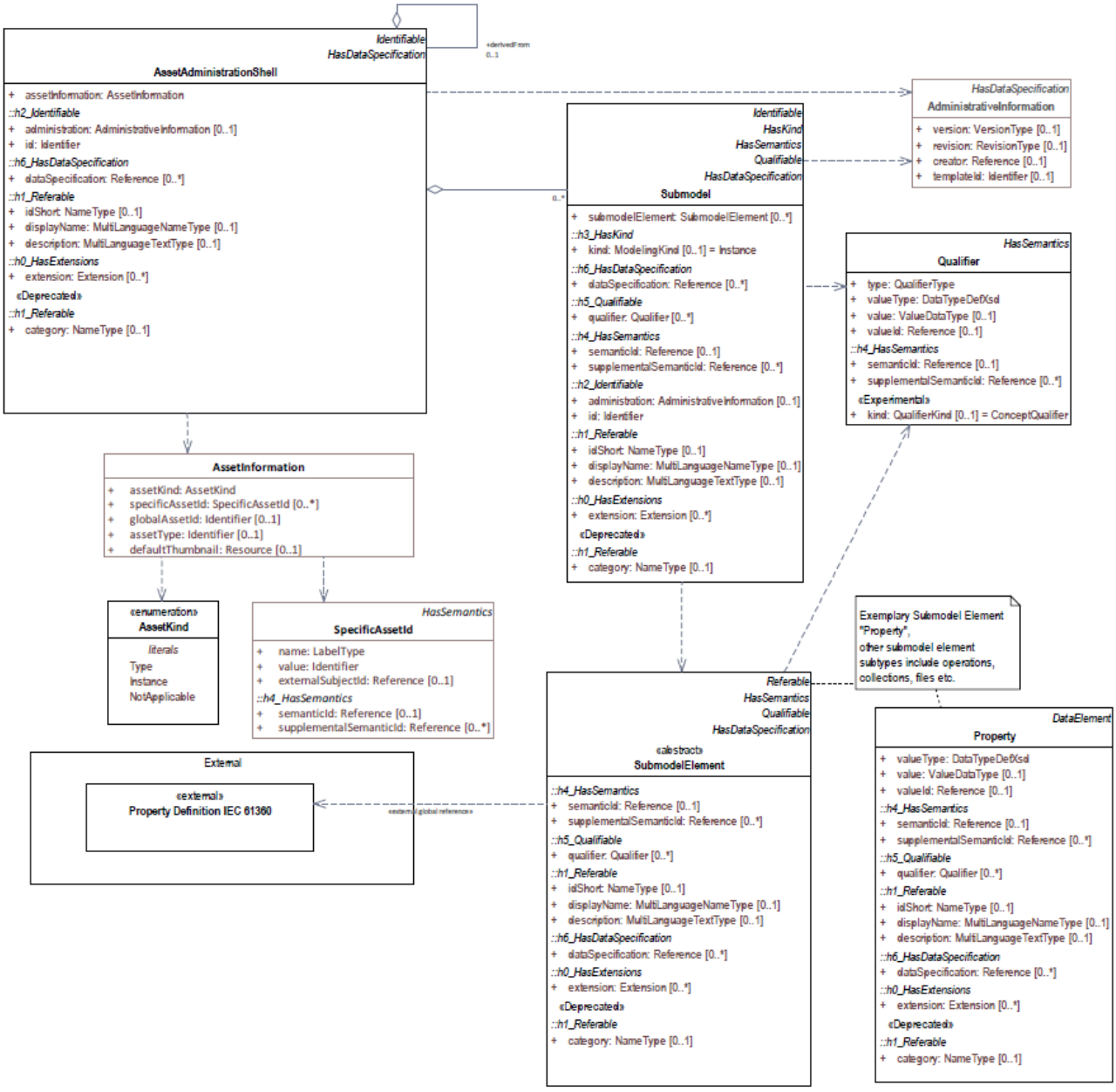


Figure 9: AAS Metamodel

Figure 10, Figure 11 and Figure 12 represent the key metadata identified at the AAS Shell, Submodel, and SubmodelElement levels, respectively.

PropertyAssetInformation	Range	Card	Definition
id	Identifier	1	The globally unique identifier of the element Typically, URLs will be used. IRI (URL)
idShort	NameType	0..1	In case of identifiables, this attribute is a short name of the element. In case of a referable, this ID is an identifying string of the element within its name space.
semanticID	Reference	0..1	Identifier of the semantic definition of the element called semantic ID or also main semantic ID of the element
description	MultiLanguageNameType	0..1	Description or comments on the element The description can be provided in several languages. If no description is defined, the definition of the concept description that defines the semantics of the element is used. Additional information can be provided, e.g. if the element is qualified and which qualifier types can be expected in which context or which additional data specification templates.

Asset is related to Submodel

Figure 10: Asset key metadata

Property	Range	Card	Definition
id	Identifier	1	The globally unique identifier of the element Typically, URLs will be used. IRI (URL)
idShort	NameType	0..1	In case of identifiables, this attribute is a short name of the element. In case of a referable, this ID is an identifying string of the element within its name space.
semanticId	Reference	0..1	Identifier of the semantic definition of the element called semantic ID or also main semantic ID of the element
description	MultiLanguageNameType	0..1	Description or comments on the element The description can be provided in several languages. If no description is defined, the definition of the concept description that defines the semantics of the element is used. Additional information can be provided, e.g. if the element is qualified and which qualifier types can be expected in which context or which additional data specification templates.

Submodel is related to Asset Administration Shell, SubmodelElement

Figure 11: Submodel key metadata

Property	Range	Card	Definition
isShort	NameType	0..1	In case of identifiables, this attribute is a short name of the element. In case of a referable, this ID is an identifying string of the element within its name space.
semanticId	Reference	0..1	Identifier of the semantic definition of the element called semantic ID or also main semantic ID of the element
description	MultiLanguageNameType	0..1	Description or comments on the element The description can be provided in several languages. If no description is defined, the definition of the concept description that defines the semantics of the element is used. Additional information can be provided, e.g. if the element is qualified and which qualifier types can be expected in which context or which additional data specification templates.

SubmodelElement is related to Submodel

Figure 12: SubmodelElement key metadata

Based on this initial analysis, the DCAT-AP information model, particularly the Dataset class, was examined to identify suitable concepts for representing the identified metadata and to determine any necessary extensions.

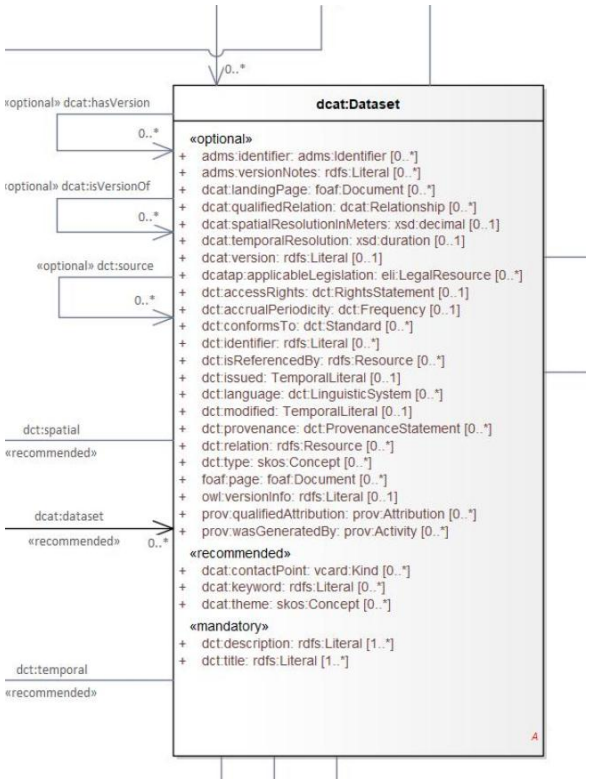


Figure 13: Dataset class in DCTA AP

A set of candidate terms was identified based on the analysis. Table 1 summarizes the results, detailing the selected DCAT-AP properties used to map each concept derived from the AAS metamodel.

Table 1: Mapping AAS metamodel and DCAT AP information model

AAS property	AAS definition	DCAT AP property	DCAT AP definition
id	The globally unique identifier of the AAS element. IRI (URL). Required for AAS API request	dct:identifier	A unique identifier of the resource being described or cataloged
idShort	In case of identifiable, this attribute is a short name of the element. In the case of a referable, this ID is an identifying string of the element within its name space.	dct:title	A name given to the resource.
semanticId	Identifier of the semantic definition of the element called semantic ID or also main semantic ID of the element	dcat:themeTaxonomy	A knowledge organization system (KOS) used to classify the resources documented in the catalog (e.g., datasets and services).
Submodel parent	Parent Submodel corresponds to a Submodel that has other	dcterms:isPartOf	A related resource in which the described resource is

AAS property	AAS definition	DCAT AP property	DCAT AP definition
	Submodels or SubmodelElements defined in a hierarchy level under it		physically or logically included.
Submodel childs	List of Submodel or SubmodelElements that are defined under another parent Submodel	dcterms:hasPart	A related resource that is included either physically or logically in the described resource.

This additional metadata serves as the basis to extend the DCAT model in the TDC and to develop the Tekniker AAS Extension for an improved data offering discovery based on the AAS metamodel.

3.2.2 Data Usage Layer: TDC EDC Compatibility Test

The DSP is currently in the process of standardization. The specification document is almost finalized with some details remaining. However, although there is already a reference implementation, the EDC, no official TCK has been published to validate the correct implementation of the DSP in other solutions such as the TDC.

In this context, as there is no TCK already available, to validate secure and interoperable communications between Dataspace Connectors within the Tec4MaaSEs Data Space, some compatibility checks have been carried out between the TDC and the EDC.

Compatibility has been validated considering the TDC as the Data Provider and EDC as the Data Consumer and vice versa. In turn, tests have been carried out for the three complementary protocols that compose the DSP: the Catalog Protocol for dataset discovery, the Contract Negotiation Protocol for policy negotiation and the Transfer Process Protocol for the access of agreed datasets.

3.3 Implementation

The following subsections describe the implementation of the two key contributions developed in the proposed approach to enable interoperable and sovereign data sharing from AAS based digital twins and the TDC. These contributions are: the TDC AAS Extension and the TDC–EDC Compatibility Test.

3.3.1 Data Offering Layer: TDC AAS Extension

The integration of AAS data into the TDC ecosystem requires a structured and standardized approach to ensure semantic consistency, interoperability, and compliance with dataspace protocols. This section provides a comprehensive overview of the technical components, workflows, and architectural elements involved in enabling this integration.

3.3.1.1 Extending the DCAT-model library for AAS Integration

TDC includes, as part of their own java libraries, a specific library that implements the DCAT-model (TDC DCAT library) to ensure that all services of TDC share the same model. The TDC AAS extension also integrates this DCAT-model library. The previous mapping of the AAS metamodel to the DCAT-AP information model revealed the need to enhance the TDC DCAT Java library to accommodate additional AAS-specific parameters. To address this, a new version of the library was developed, incorporating these extended properties.

An excerpt of the code implementation showcasing the new properties is presented below.

```
@JsonProperty("dcat:themeTaxonomy")
@JsonAlias({"dcat:themeTaxonomy", "theme"})
@Schema(description = "DataSet theme", example = "endpointOWL")
private String theme;

@JsonProperty("dct:identifier")
@JsonAlias({"dct:identifier", "otherIdentifier"})
@Schema(description = "Dataset other identifier", example = "other id")
private String otherIdentifier;

@JsonProperty("dcterms:isPartOf")
@JsonAlias({"dcterms:isPartOf", "isPartOf"})
@Schema(description = "DataSet isPartOf", example =  "UUID")
private UUID isPartOf;

@JsonProperty("dcterms:hasPart")
@JsonAlias({"dcterms:hasPart", "hasPart"})
@Schema(description = "DataSet hasPart", example =  "[\"UUID1\",\"UUID2\"]")
private ArrayList<UUID> hasPart;

public DataSet() {}
```

Figure 14: Code excerpt illustrating the extended TDC DCAT- model library with new properties

3.3.1.2 TDC AAS Extension Workflow

TDC AAS integration workflow enables the transformation of AAS Shell, Submodel, and SubmodelElement into publishable datasets, ensuring they can be discovered, accessed, and governed via the TDC infrastructure.

To better understand the integration and workflow of the TDC AAS extension, Figure 15 provides an overview of the main components and their interactions throughout the process.

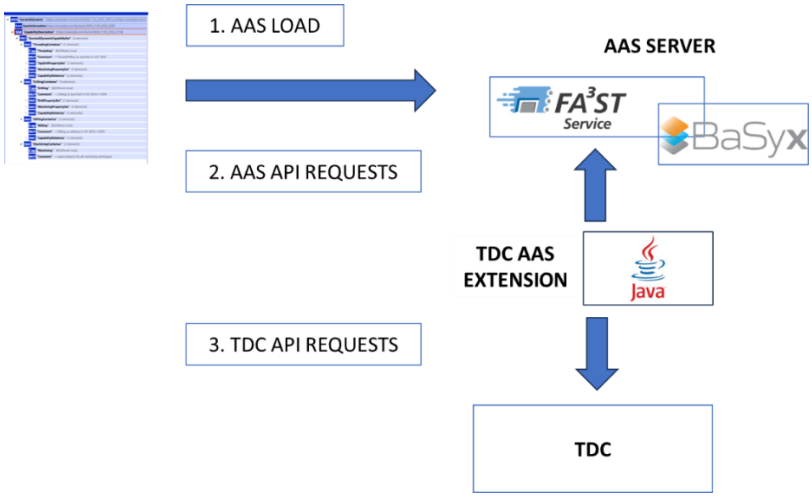


Figure 15: AAS – TDC Integration workflow

The process begins by loading an Asset Administration Shell (AAS) model, which includes Submodel, and SubmodelElement. This model provides the digital representation of an asset (i.e., digital twin). The digital twin contains structured information about the asset, such as technical properties, capabilities, capacities, and other relevant data.

Once the AAS is loaded, the next step is to interact with it through API requests. These requests are handled by an AAS server, such as FA³ST Service or Eclipse BaSyx. The AAS server exposes the AAS data and functionality via a RESTful API, allowing external applications to query or update the asset's digital representation.

The TDC AAS extension communicates with the AAS server by sending API requests to retrieve the AAS model data and to define datasets according to the DCAT-AP model.

Finally, the TDC AAS extension sends the necessary requests to the TDC API to create the catalog, datasets, and other required entities and relationships as defined by the Dataspace protocol. This ensures that data shared across organizational boundaries is managed with integrity and in compliance with agreed-upon policies.

Figure 16 shows the Swagger UI used to submit the publication request of the AAS to the TDC. The input JSON payload requires parameters such as the AAS URL, which allows the extension to interact with an accessible AAS server. Other parameters include metadata for catalog definition, such as title, keywords, and description.

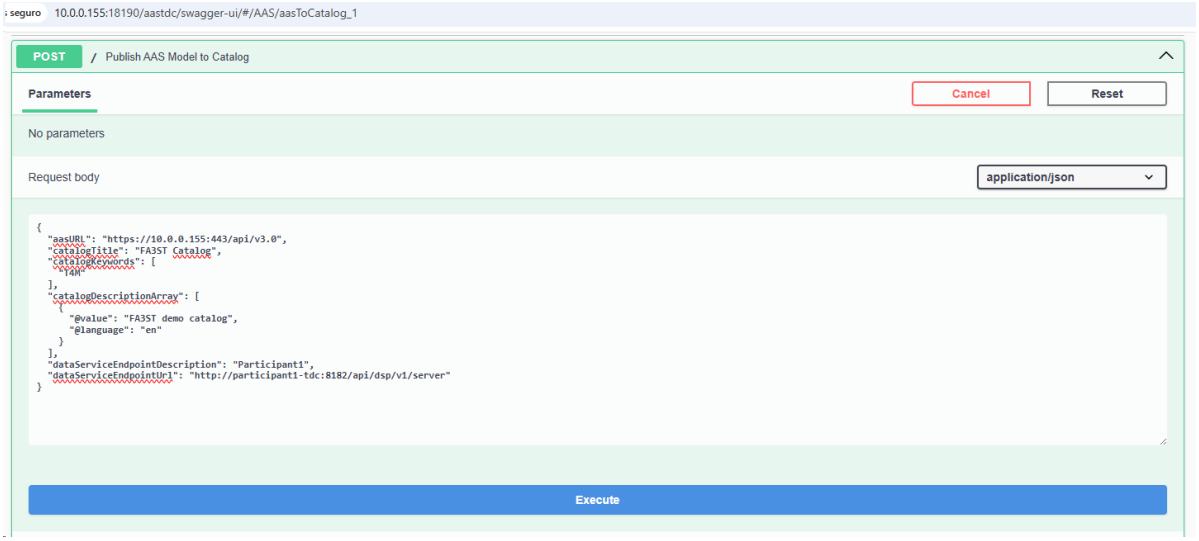


Figure 16: TDC AAS extension - Swagger UI

Once datasets are published to the data connector, the TDC UI enables users to explore and download Submodel, and SubmodelElement data directly from the interface as well as to assign the appropriate usage policies.

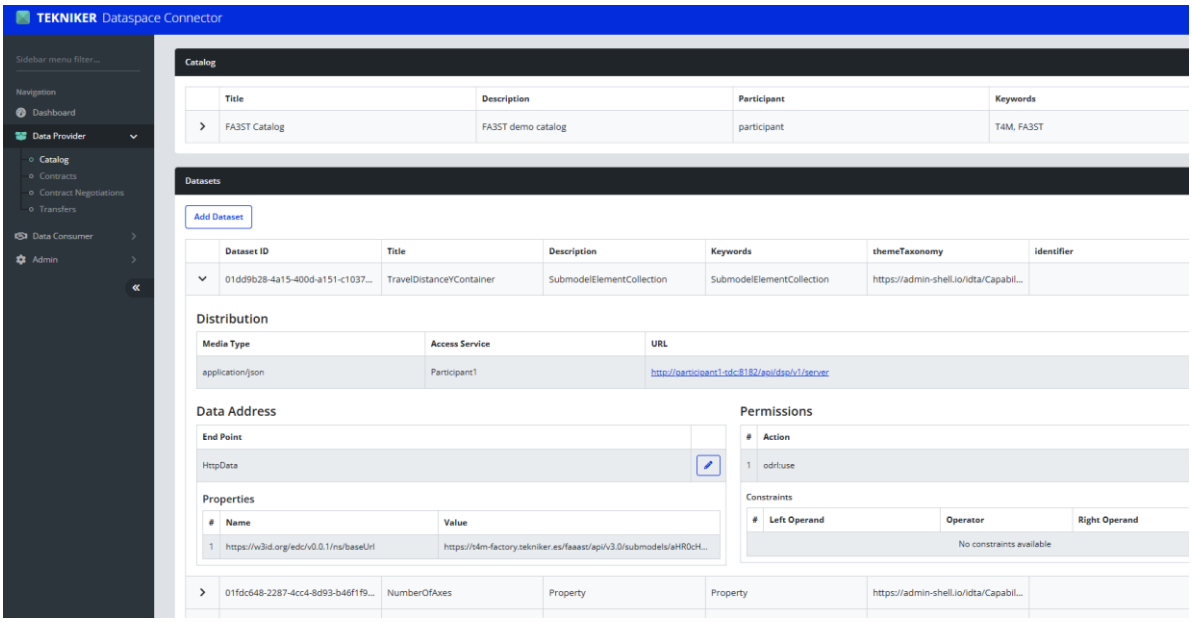


Figure 17: Excerpt from the TDC UI showing datasets and the assignment of usage policies

The TDC AAS extension is composed of a collection of projects designed to publish Submodel, and SubmodelElement as datasets in the TDC Catalog. These projects are developed in Java using the Maven framework.

The TDC AAS extension consists of three projects:

- **tdc.api.client:** A set of API requests to the TDC for creating the necessary entities to populate the catalog, including Catalog, DataAddress, DataService, DataSet, Distribution, PolicyOffer, and the required relationships among them.
- **aas.server:** The core project of the extension responsible for:
    1. Parsing AAS shells, Submodel, and SubmodelElement from the AAS server.
    2. Creating datasets and other required entities based on the DCAT AP model.
    3. Publishing datasets to the TDC using methods defined in the tdc.api.client project (TDC API Client).
- **aas.ws**: An API service for the AAS extension that exposes methods to handle requests for publishing AAS Shells, Submodel, and SubmodelElement as datasets in the TDC.

3.3.1.3 Initial results

Several tests were carried out to create datasets from different AAS models, including various submodels (e.g., Nameplate submodel, Hardcover submodel, Technical Data submodel, CapabilityDefinition submodel, etc.). These tests were successful in generating the expected datasets from the AAS models using the TDC AAS extension. Below is an example illustrating the results obtained when using a representative AAS model within the scope of Tec4MaaSES,

The selected AAS model represents a machining manufacturing resource, incorporating both a CapabilityDescription submodel and a TechnicalData submodel. These submodels collectively define key aspects of the resource, including operational capabilities, performance characteristics, and technical specifications.

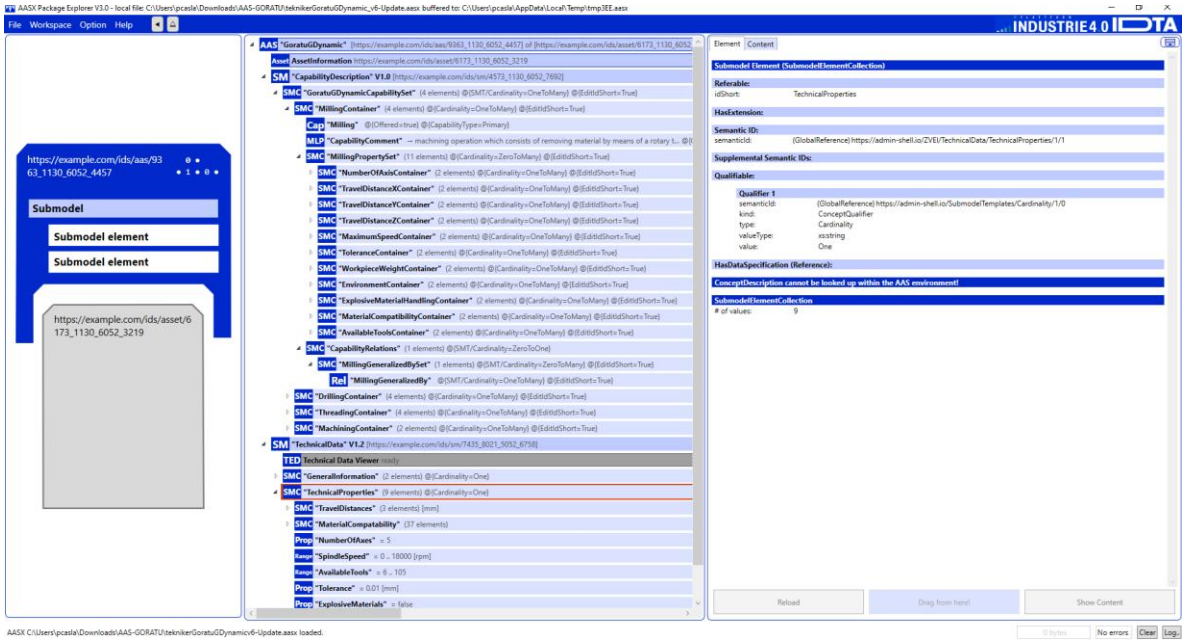


Figure 18: AAS model representing the capabilities and technical data of a machining resource

The process followed to create the datasets included the following steps:

- AAS Model Deployment: the AAS model representing the digital twin of physical manufacturing resource deployed on both FA³ST and BaSyx AAS servers.
- API Access: The AAS servers exposed the model via RESTful APIs, enabling standardized access for external tools.
- TDC AAS Extension Integration: The TDC AAS extension accessed the AAS model through these APIs, extracted metadata, and mapped it to the DCAT-AP standard to create structured datasets.
- Dataset Publication: The generated datasets were published to the TDC . This included creating catalogs and metadata, triggered via Swagger UI with a JSON payload specifying the AAS URL and descriptive fields.

Figure 19 shows the user interface of the TDC, where the datasets automatically generated through the AAS–TDC connector are displayed. It can be observed that extended attributes are included to enhance expressiveness and help users better understand the exposed datasets. Specifically, it includes the identification of the AAS metamodel's semantic ID via dcat:themeTaxonomy, and shows the relationships between Asset Shell, Submodel, and SubmodelElement using dcterms:isPartOf and dcterms:hasPart.

One of the main issues observed in the results is that, depending on the complexity of the AAS model, a large number of datasets can be generated. In this case, 144 datasets were created, as datasets were generated at the SubmodelElement level. Although expressiveness is improved, a tabular representation makes it difficult to understand the relationships between the different datasets and to navigate among them.

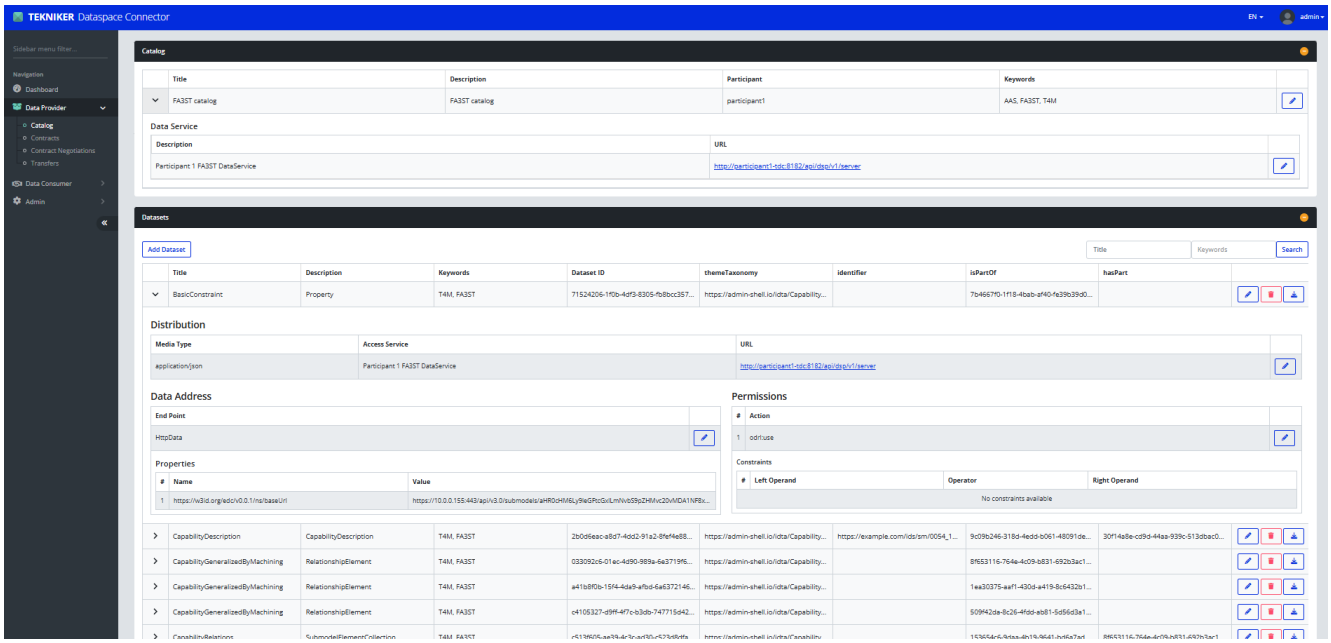
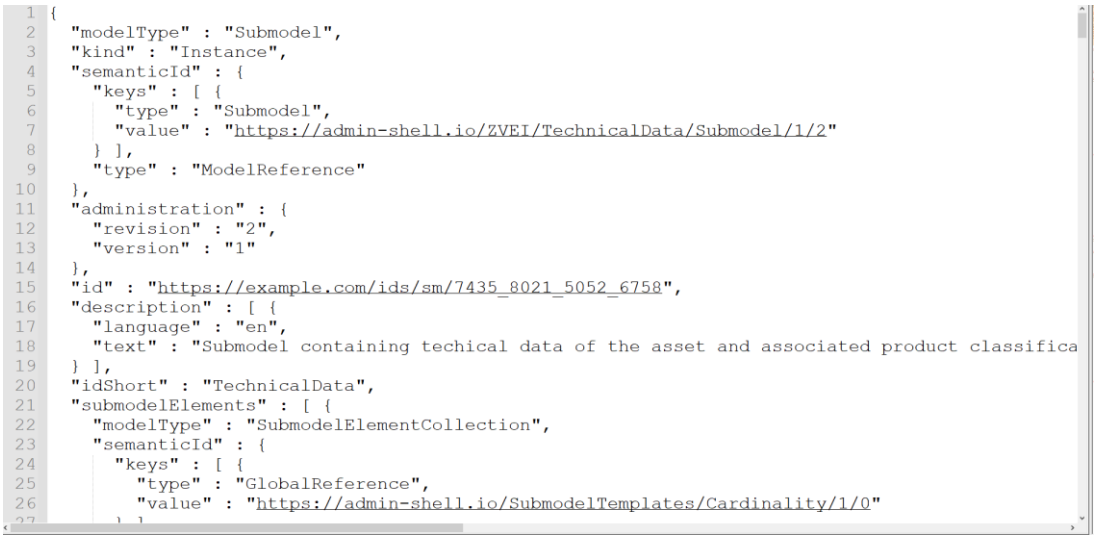


Figure 19: TDC UI showing the generated datasets including the DCAT extended (thme, isPartOf, hasPart)

Figure 20 shows an excerpt of what is obtained when selecting one of the datasets and accessing its data, specifically, the dataset corresponding to the TechnicalData Submodel.



```
{
  "modelType" : "Submodel",
  "kind" : "Instance",
  "semanticId" : {
    "keys" : [ {
      "type" : "Submodel",
      "value" : "https://admin-shell.io/ZVEI/TechnicalData/Submodel/1/2"
    } ],
    "type" : "ModelReference"
  },
  "administration" : {
    "revision" : "2",
    "version" : "1"
  },
  "id" : "https://example.com/ids/sm/7435_8021_5052_6758",
  "description" : [ {
    "language" : "en",
    "text" : "Submodel containing techical data of the asset and associated product classifica
  } ],
  "idShort" : "TechnicalData",
  "submodelElements" : [ {
    "modelType" : "SubmodelElementCollection",
    "semanticId" : {
      "keys" : [ {
        "type" : "GlobalReference",
        "value" : "https://admin-shell.io/SubmodelTemplates/Cardinality/1/0"
```

Figure 20: JSON excerpt

Compatibility with both FA³ST and BaSyx was validated by testing the TDC AAS extension on each AAS server using the same AAS models. The results confirmed full interoperability: in both cases, identical datasets were successfully generated in the TDC catalog in all the test cases.

3.3.2 Data Usage Layer: TDC EDC Compatibility Test

The compatibility between the TDC and the EDC has been split into two different scenarios. While the first one addresses the scenario in which the TDC acts as the Data Provider and the EDC as the Data Consumer,

the second one addresses the scenario in which the EDC acts as the Data Provider and the TDC as the Data Consumer. The tests performed in both scenarios are described below.

3.3.2.1 TDC as Data Provider and EDC as Data Consumer

The tests have been carried out for the three protocols that compose the DSP: Catalog Protocol, Contract Negotiation Protocol and Transfer Process Protocol. The three tests are detailed below.

3.3.2.1.1 Catalog Protocol test

To test the Catalog Protocol, a catalog of datasets has been defined in the TDC. Figure 21 shows the catalog through the TDC UI.

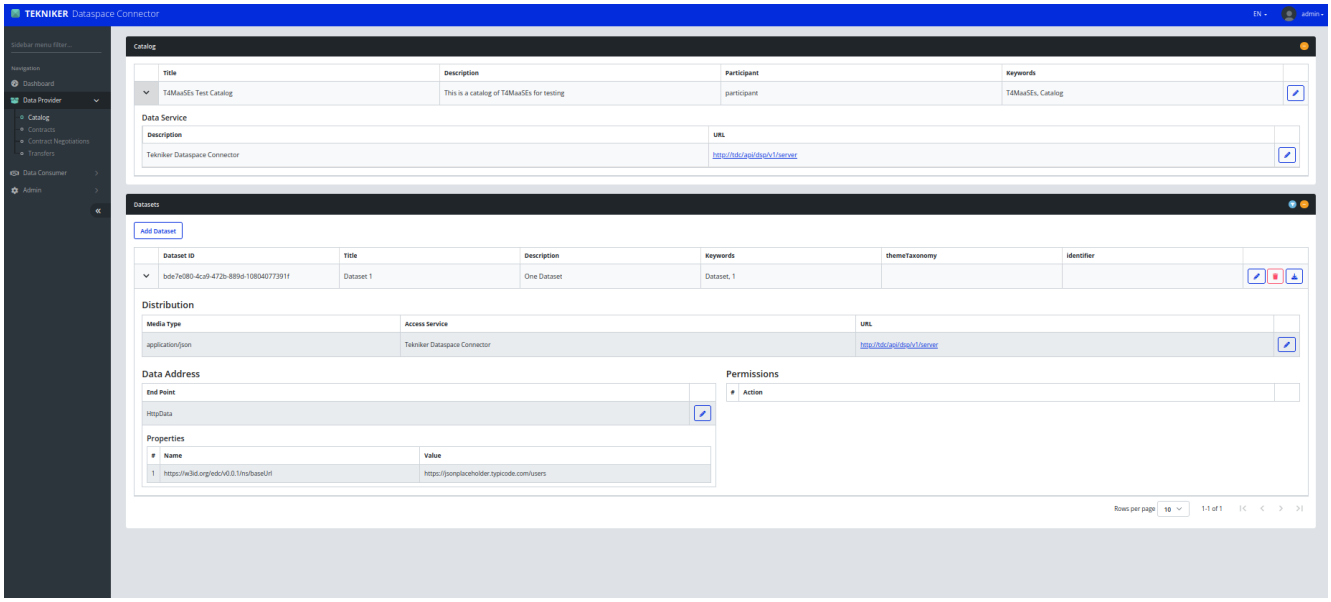


Figure 21: TDC UI - Catalog

Afterwards, the catalog has been requested through the EDC. Without a UI for the EDC, Figure 22 shows the request through Postman as well as the catalog received from the TDC.

Figure 22

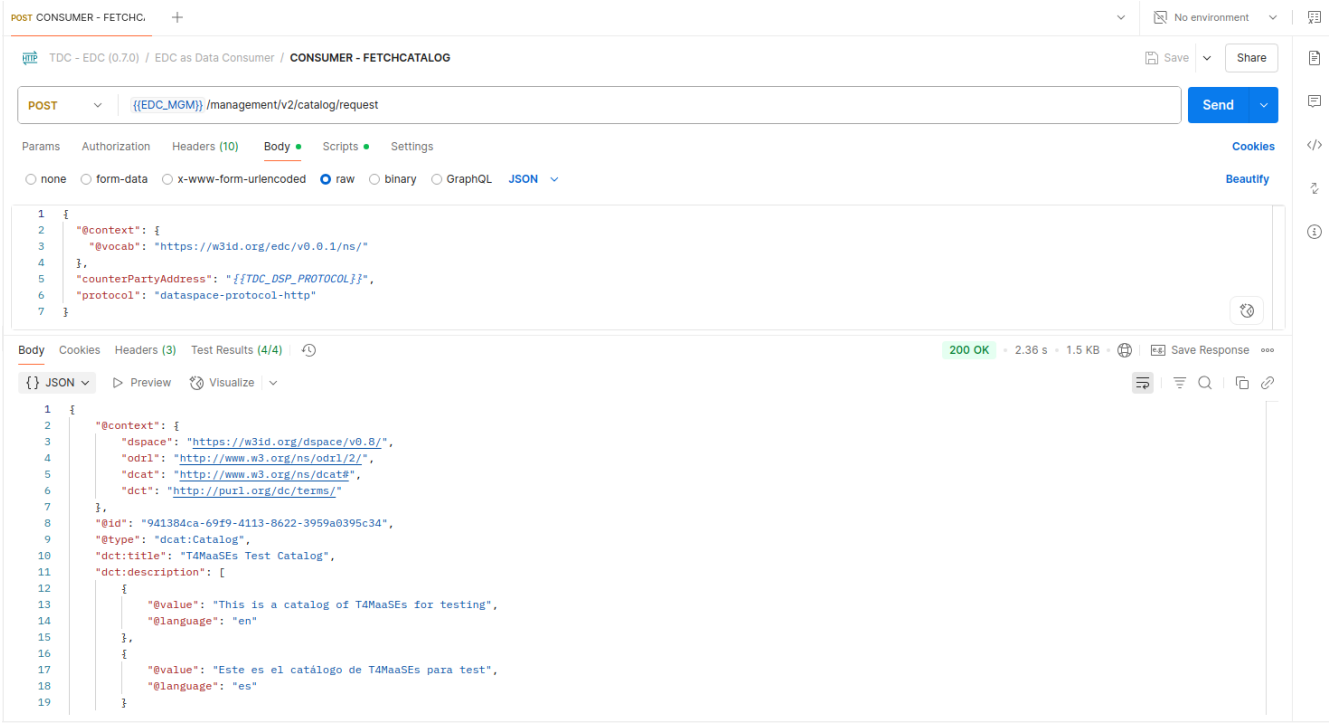


Figure 22: EDC Postman Request - Catalog request

Thus, validating the compatibility of the Catalog Protocol between the TDC acting as a Data Provider and the EDC acting as the Data Consumer.

3.3.2.1.2 Contract Negotiation Protocol test

To test the Contract Negotiation Protocol, it should be noted that the EDC only implements a simple contract negotiation process in which the policies from a dataset of the Data Provider are accepted. In this regard, Figure 23 shows the request accepting the policies for a dataset from a Data Provider through the Postman as well as the response received from the TDC accepting the start of the contract negotiation process.

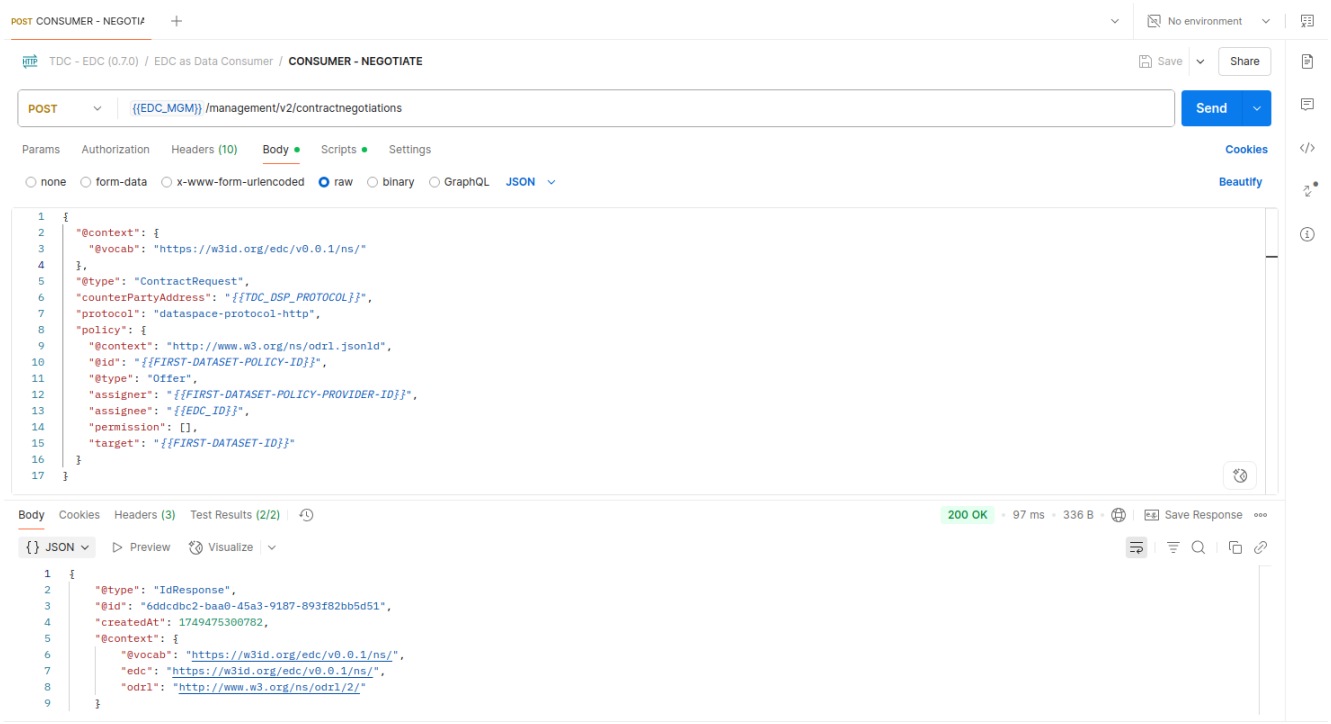


Figure 23: EDC Postman Request – Contract Negotiation request

As the contract negotiation process takes place in background, both the EDC and the TDC can monitor the state of the contract negotiation process. While a FINALIZED state will mean a successful agreement, a TERMINATED state will mean an unsuccessful contract negotiation process. Figure 24 and Figure 25 show, respectively, the request to the EDC through Postman and to the TDC through the UI for the contract negotiation state and the result.

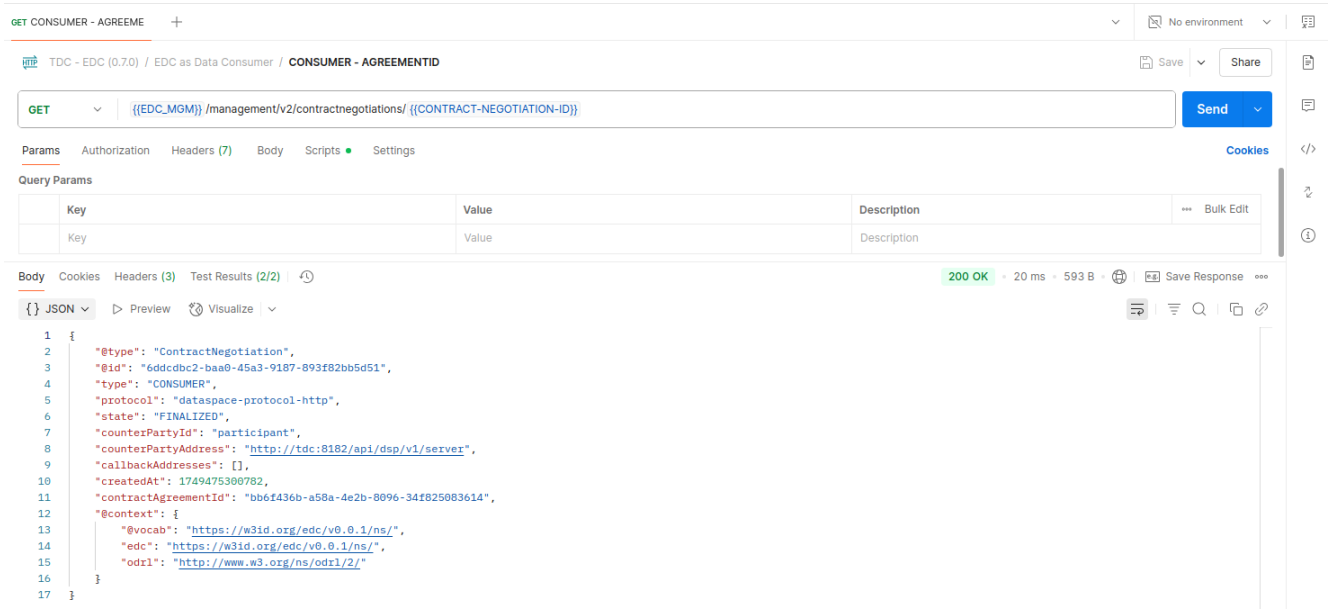


Figure 24: EDC Postman Request – Contract Negotiation state request

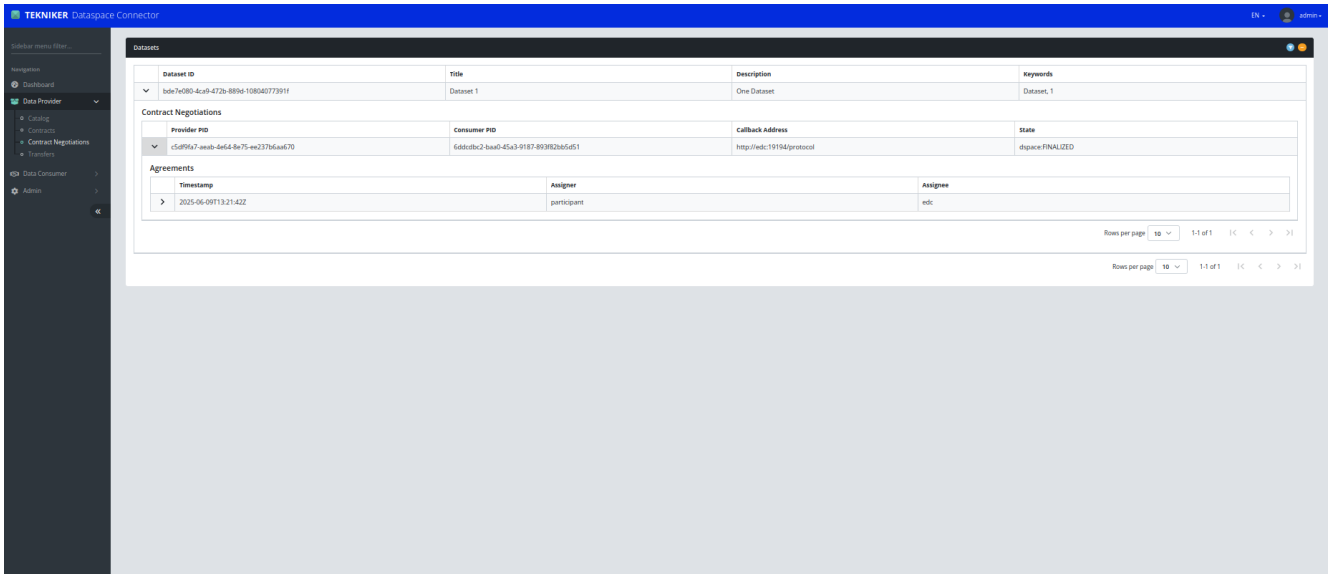


Figure 25: TDC- UI: Contract Negotiation state request

Both demonstrates that the contract negotiation has successfully FINALIZED. In turn, Figure 14 represents the resulting agreement. Thus, validating the compatibility of the Contract Negotiation Protocol between the TDC acting as a Data Provider and the EDC acting as the Data Consumer.

3.3.2.1.3 Transfer Process Protocol test

To test the Transfer Process Protocol, it should be noted that data can be shared following a pull and push approach. The tests performed for both approaches are described below.

Pull approach

To test the Transfer Process Protocol following the pull approach, a request to the EDC should be performed to get from the TDC a temporary access to the agreed Dataset. Figure 26 shows this request through Postman as well as the response from the TDC accepting the transfer process.

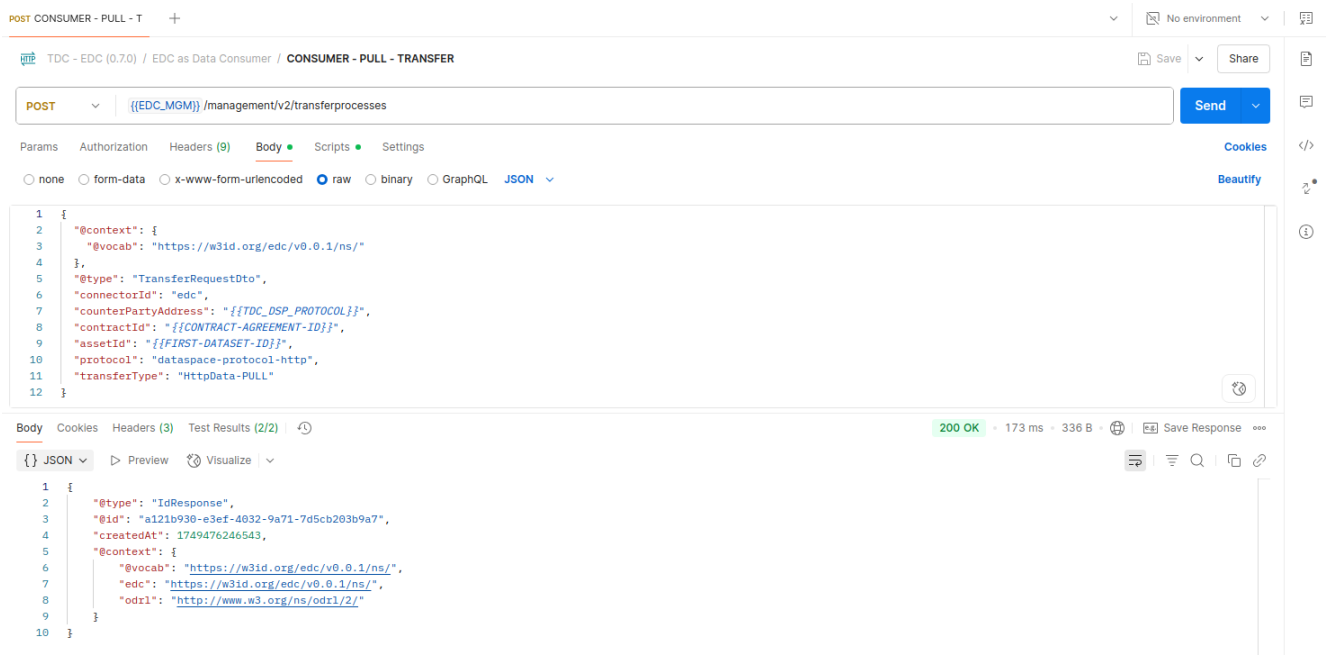


Figure 26: EDC Postman Request – PULL Transfer Process request

Once requested, the transfer process can be monitored. Figure 27 and Figure 28 respectively show the request to the EDC through Postman and to the TDC through the UI for the transfer process state.

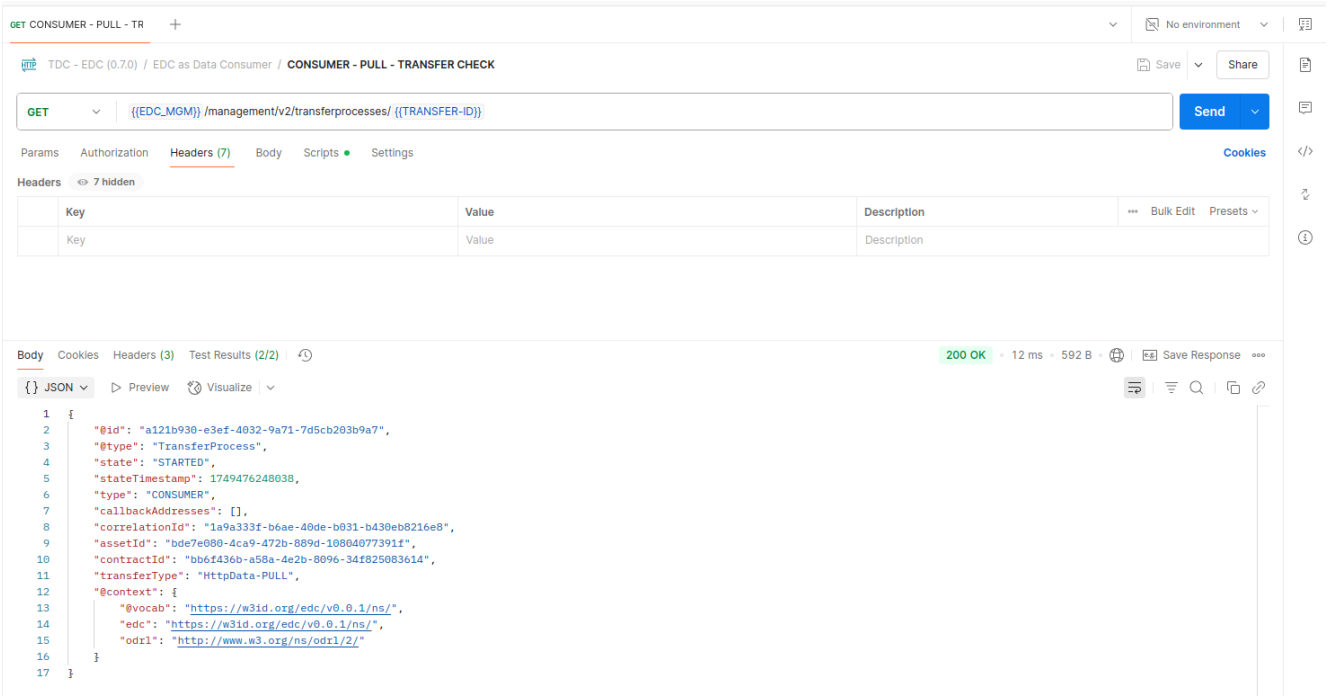


Figure 27: EDC Postman Request – PULL Transfer Process state request

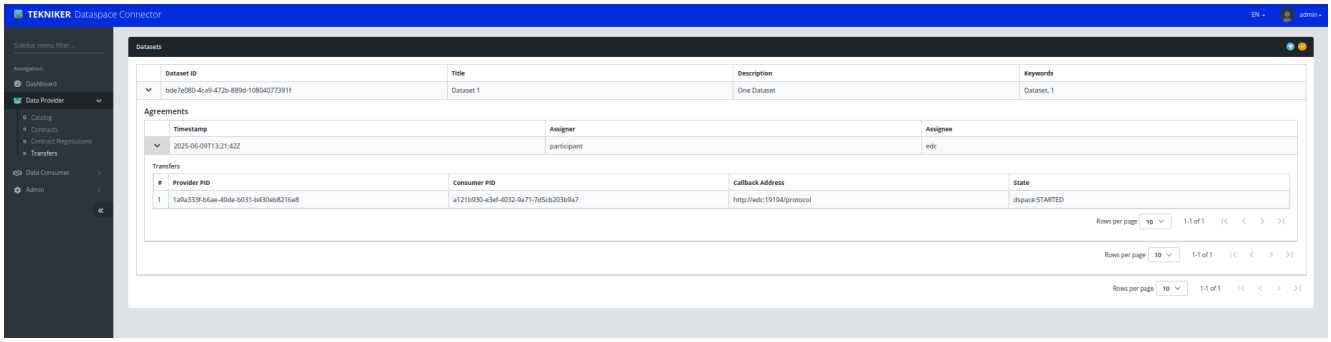


Figure 28: TDC UI – PULL Transfer Process state request

If a STARTED state is reached, the dataset will be temporarily accessible. Figure 29 and Figure 30 respectively shows the request of the data source to the EDC and the data itself to the TDC through Postman as well as the data retrieved.

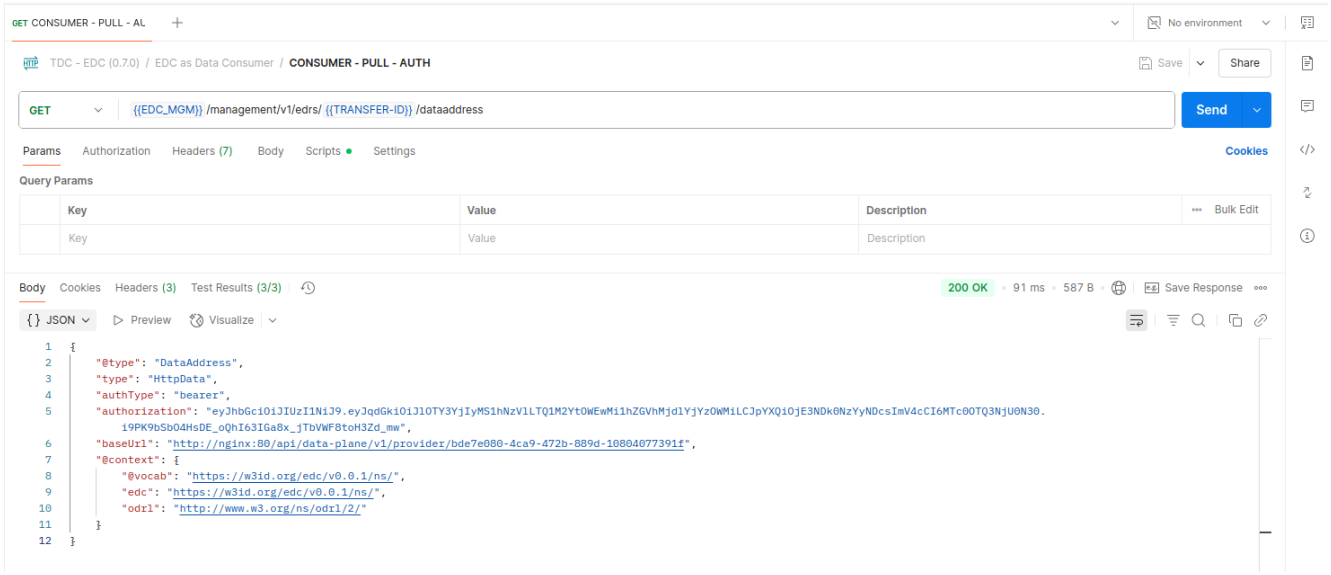


Figure 29: EDC Postman Request – PULL Data Source request

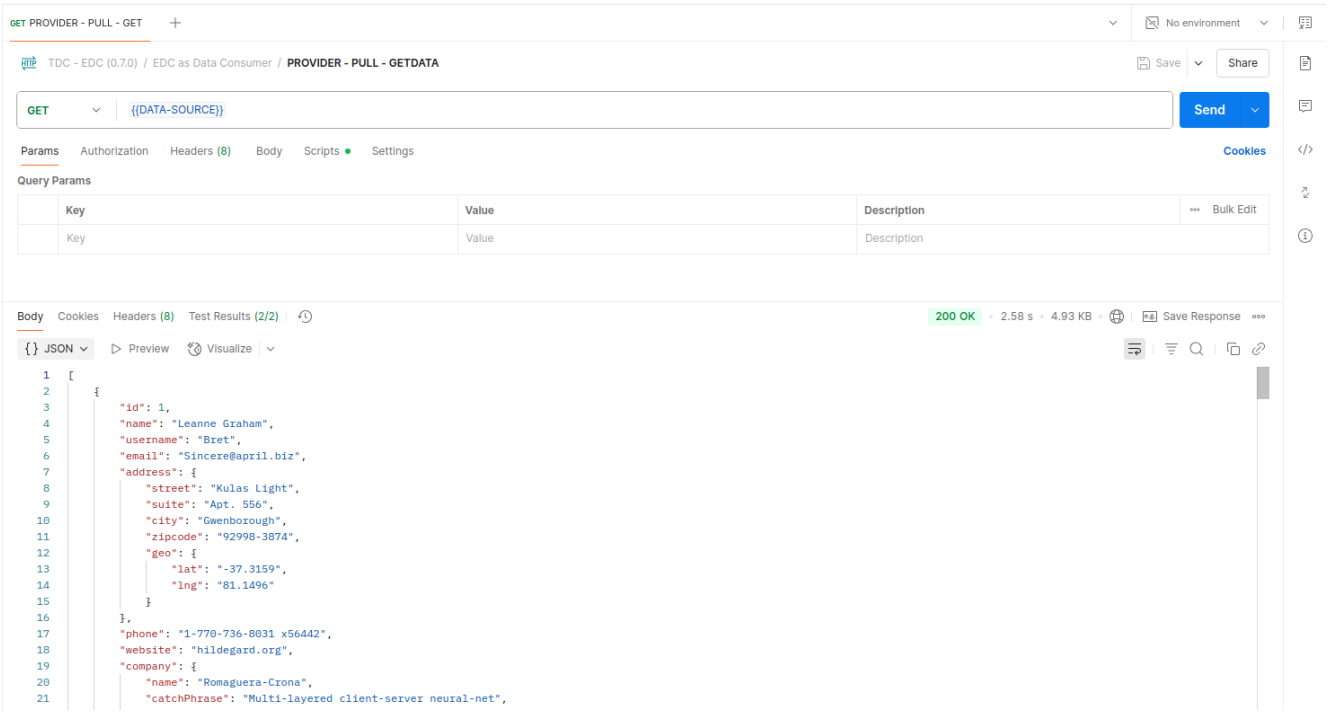


Figure 30: EDC Postman Request – PULL Data request

Thus, validating the compatibility of the Transfer Process Protocol between the TDC acting as a Data Provider and the EDC acting as the Data Consumer for the pull approach.

Push approach

To test the Transfer Process Protocol following the push approach, a request to the EDC should be performed to trigger data sharing from the TDC to a determined processing tool. Figure 31 shows this request through Postman as well as the response from the TDC accepting the transfer process.

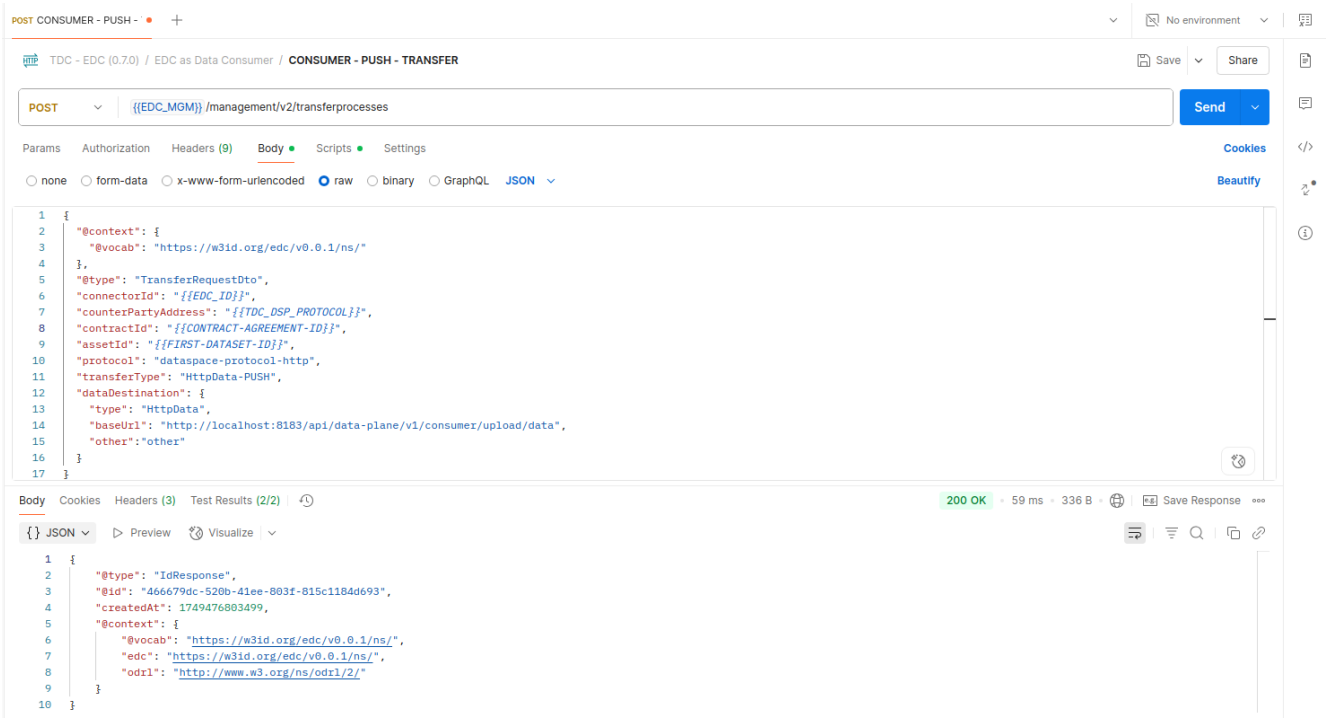


Figure 31: EDC Postman Request – PUSH Transfer Process request

Once requested, the transfer process can be monitored. Figure 32 and Figure 33 respectively show the request to the EDC through Postman and to the TDC through the UI for the transfer process state.

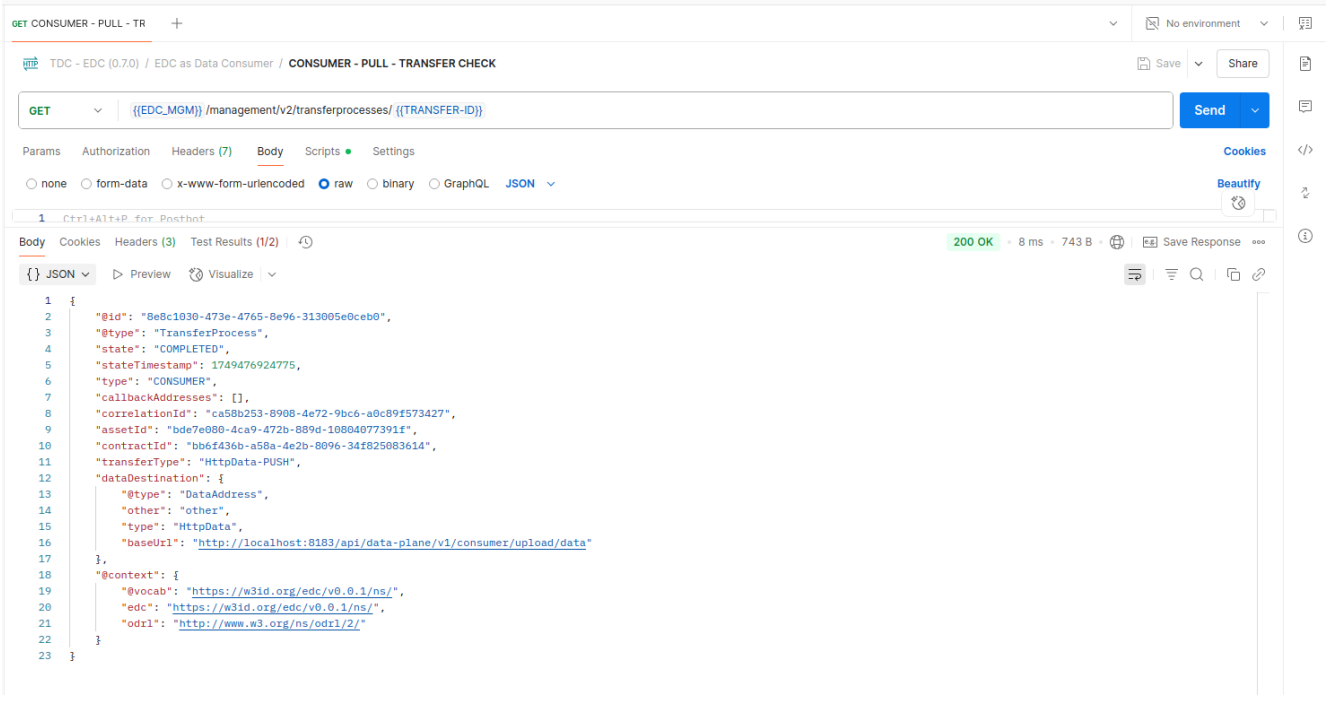


Figure 32: EDC Postman Request – PUSH Transfer Process state request

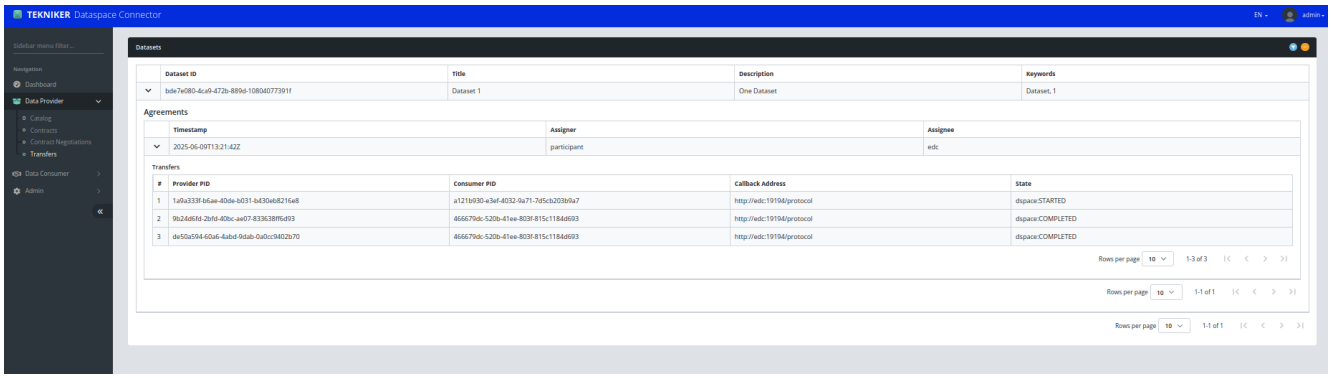


Figure 33: TDC UI – Transfer Process state request

If a COMPLETED state is reached, the Dataset has been sent. Thus, validating the compatibility of the Transfer Process Protocol between the TDC acting as a Data Provider and the EDC acting as the Data Consumer for the push approach.

3.3.2.2 EDC as Data Provider and TDC as Data Consumer

The tests have been carried out for the three protocols that compose the DSP: Catalog Protocol, Contract Negotiation Protocol and Transfer Process Protocol. The three tests are detailed below.

3.3.2.2.1 Catalog Protocol test

To test the Catalog Protocol, a catalog of datasets has been defined in the EDC. Figure 34, Figure 35 and Figure 36 show the process of creating a catalog in the EDC, i.e., add a dataset, add a policy and set the policy to the dataset.

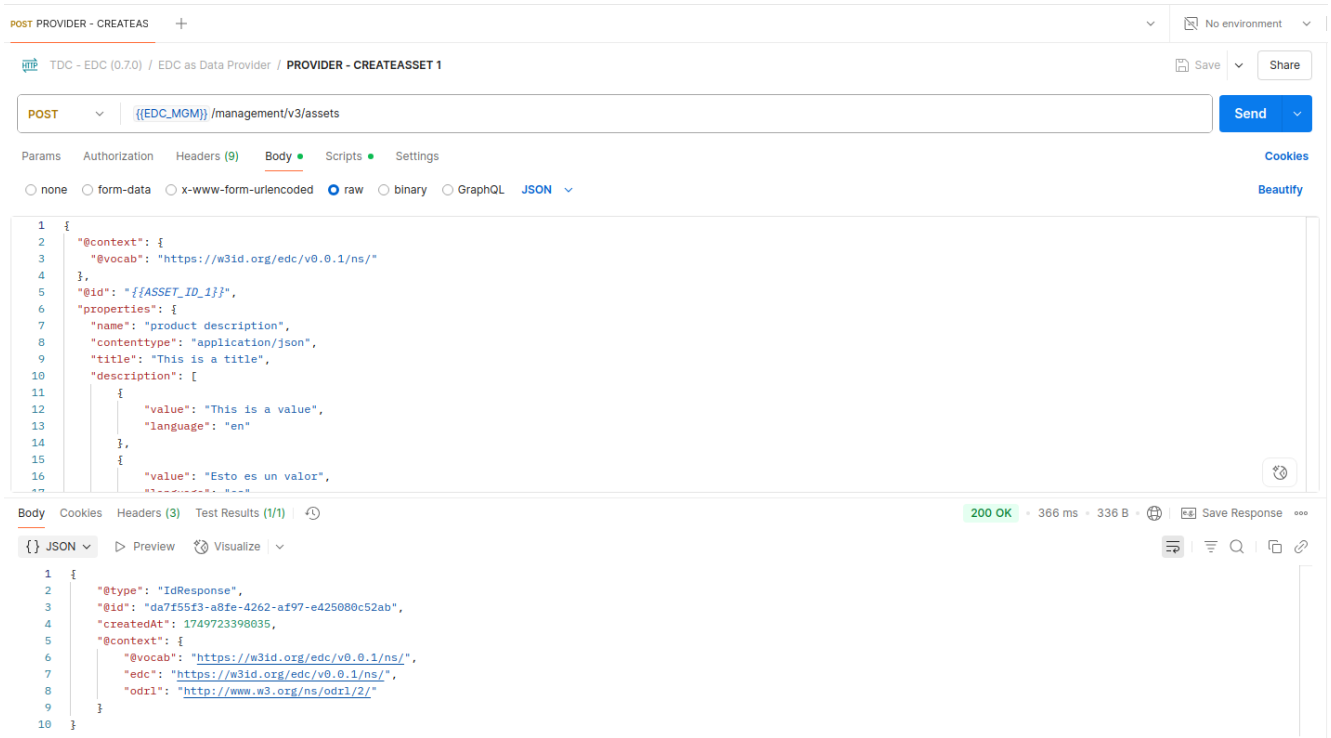


Figure 34: EDC Postman Request – Add a dataset

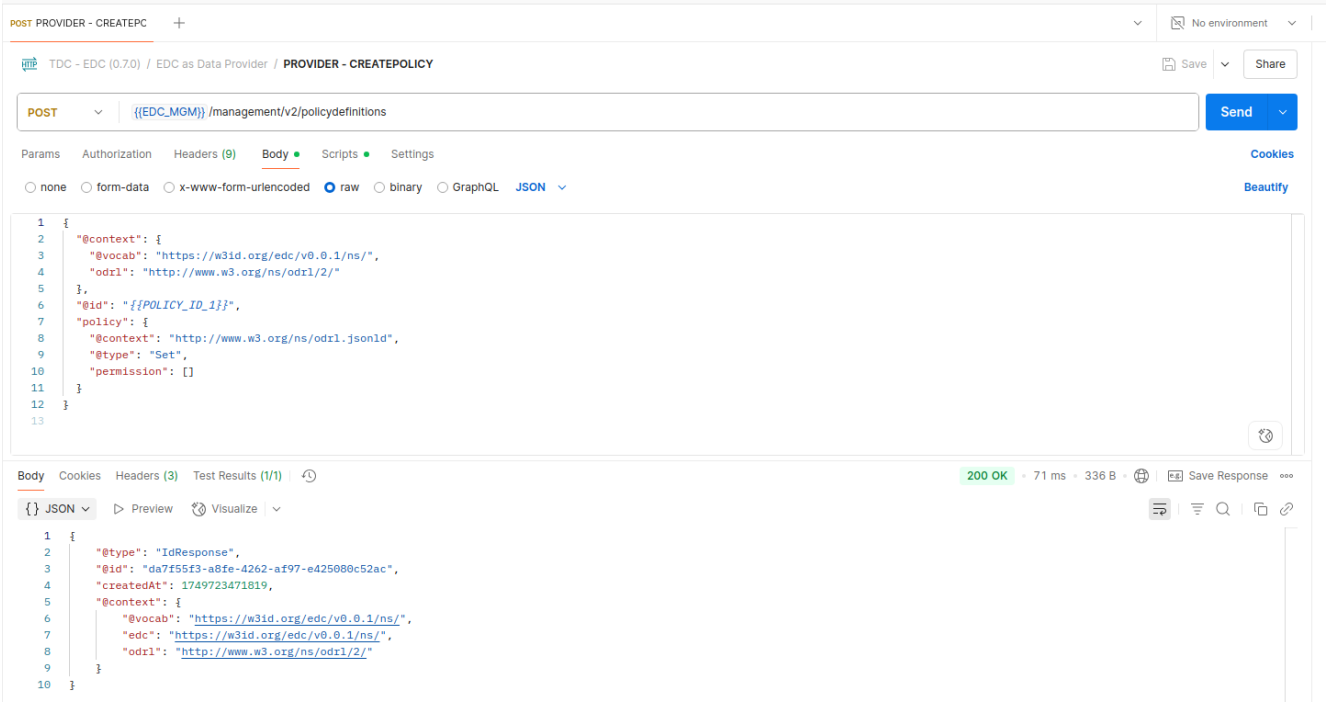


Figure 35: EDC Postman Request – Add a policy

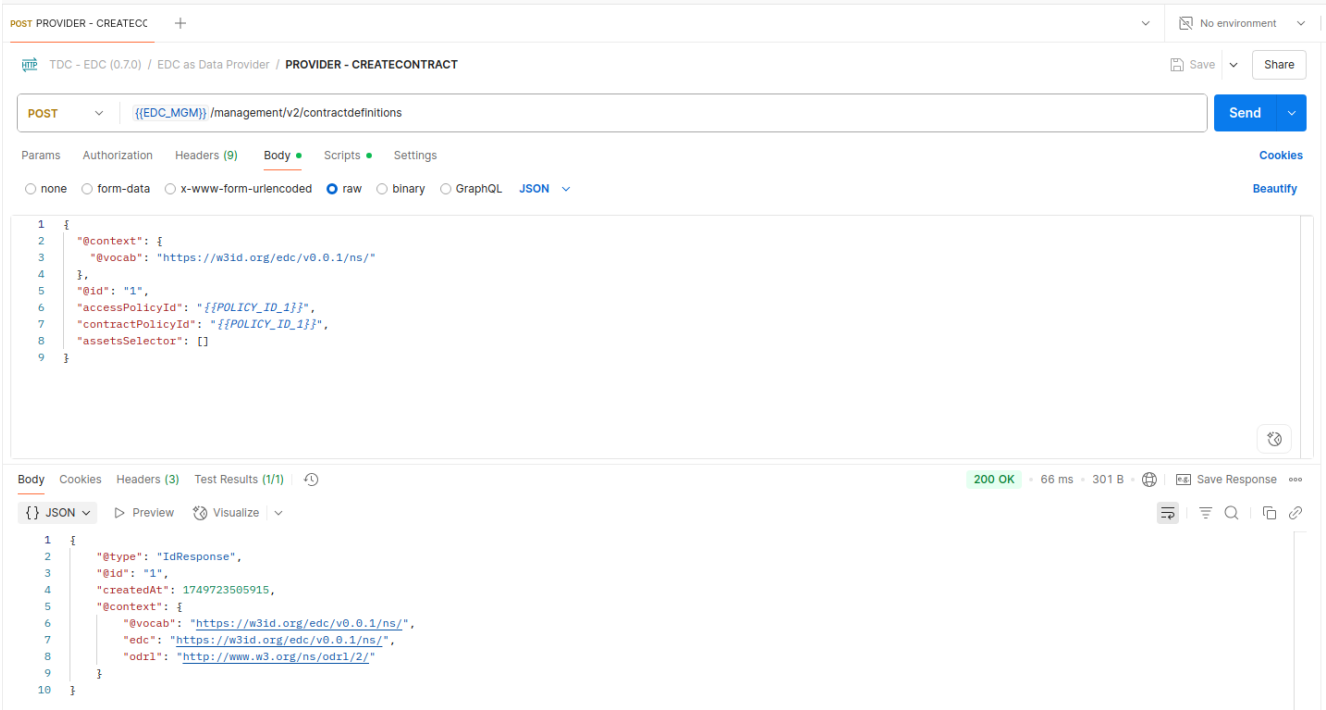


Figure 36: EDC Postman Request – Set the policy to the dataset

Then, the catalog has been requested through the TDC UI. Figure 37 shows the request as well as the catalog received from the EDC.

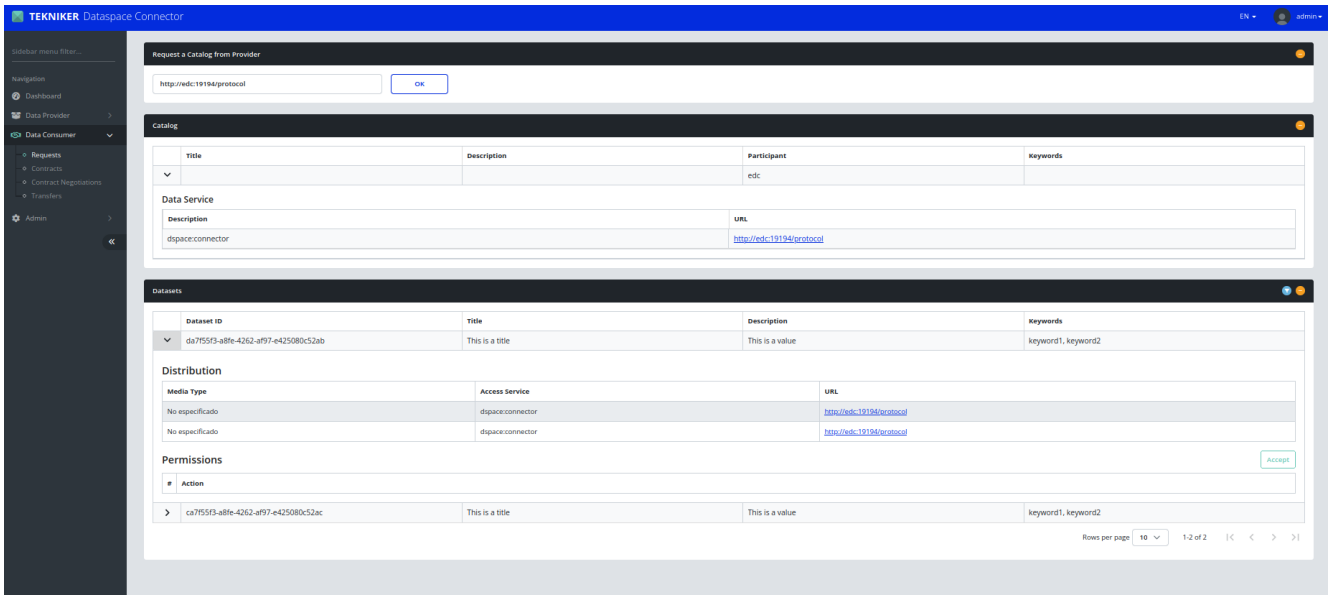


Figure 37: TDC UI - Catalog Request

Thus, validating the compatibility of the Catalog Protocol between the EDC acting as a Data Provider and the TDC acting as the Data Consumer.

3.3.2.2.2 Contract Negotiation Protocol test

To test the Contract Negotiation Protocol, the TDC UI includes an "Accept" button for each dataset from the EDC so as to accept the policies. Once accepted, the contract negotiation process takes place in background. Figure 38 and Figure 39 respectively shows in the TDC UI the contract negotiation initialization and the resulting finalized state and the agreement reached for a dataset from the EDC.

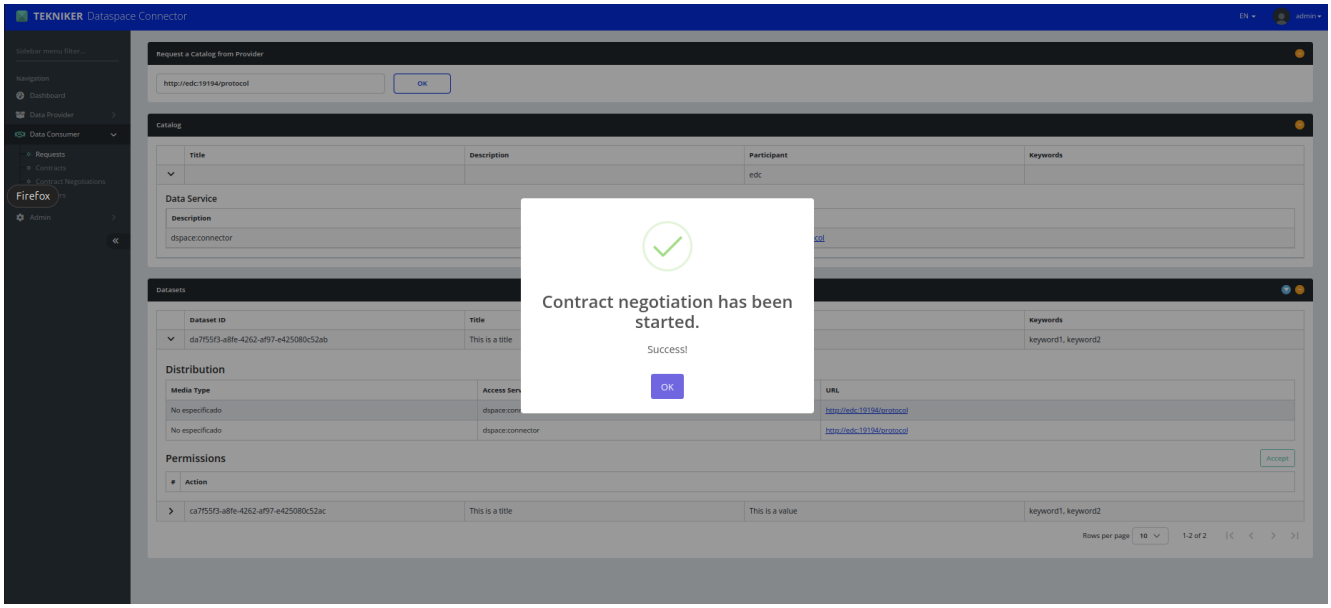


Figure 38: TDC UI - Contract Negotiation request

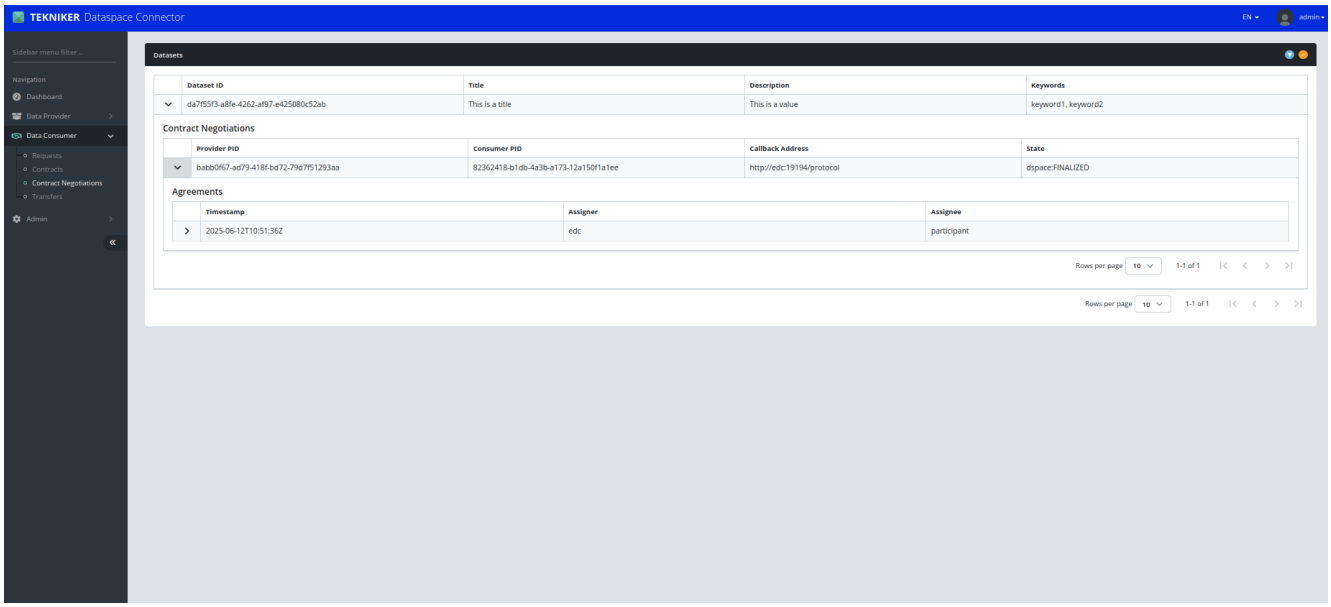


Figure 39: TDC UI – Contract Negotiation state request

It demonstrates that the contract negotiation has successfully FINALIZED with an agreement. Thus, validating the compatibility of the Contract Negotiation Protocol between the TDC acting as a Data Provider and the EDC acting as the Data Consumer

3.3.2.2.3 Transfer Process Protocol test

To test the Transfer Process Protocol, it should be noted that data can be shared following a pull and push approach. The tests performed for both approaches are described below.

Pull approach

To test the Transfer Process Protocol following the pull approach, a request to the TDC should be performed to get from the EDC a temporary access to the agreed dataset. Figure 40 and Figure 41 show this request through the TDC UI.

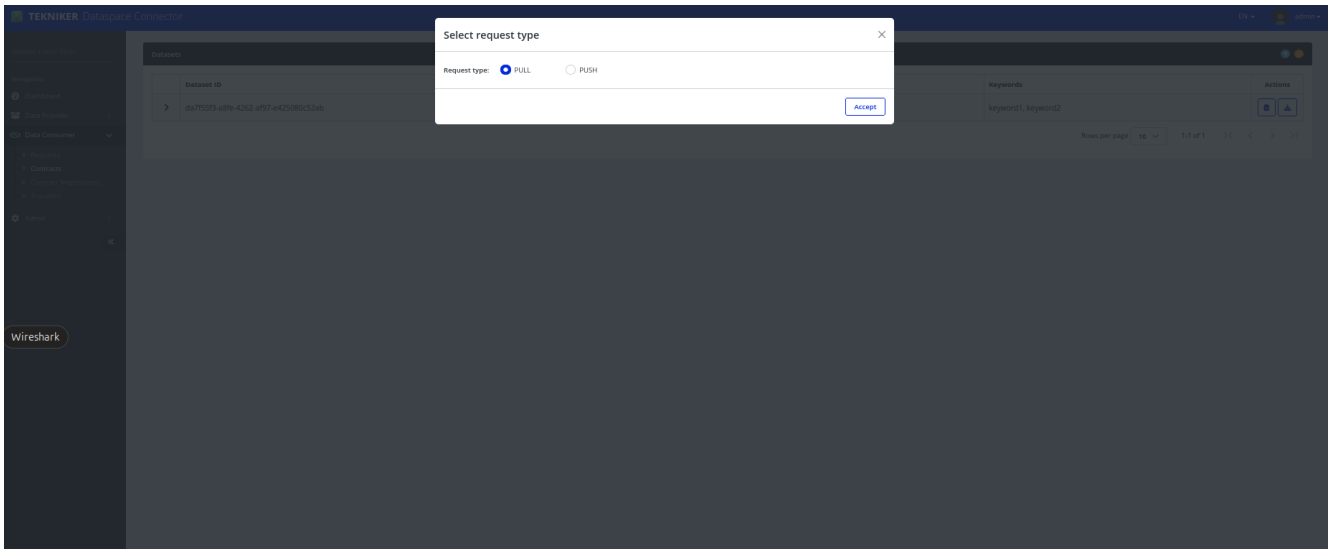


Figure 40: TDC UI – PULL Transfer Process request (1)

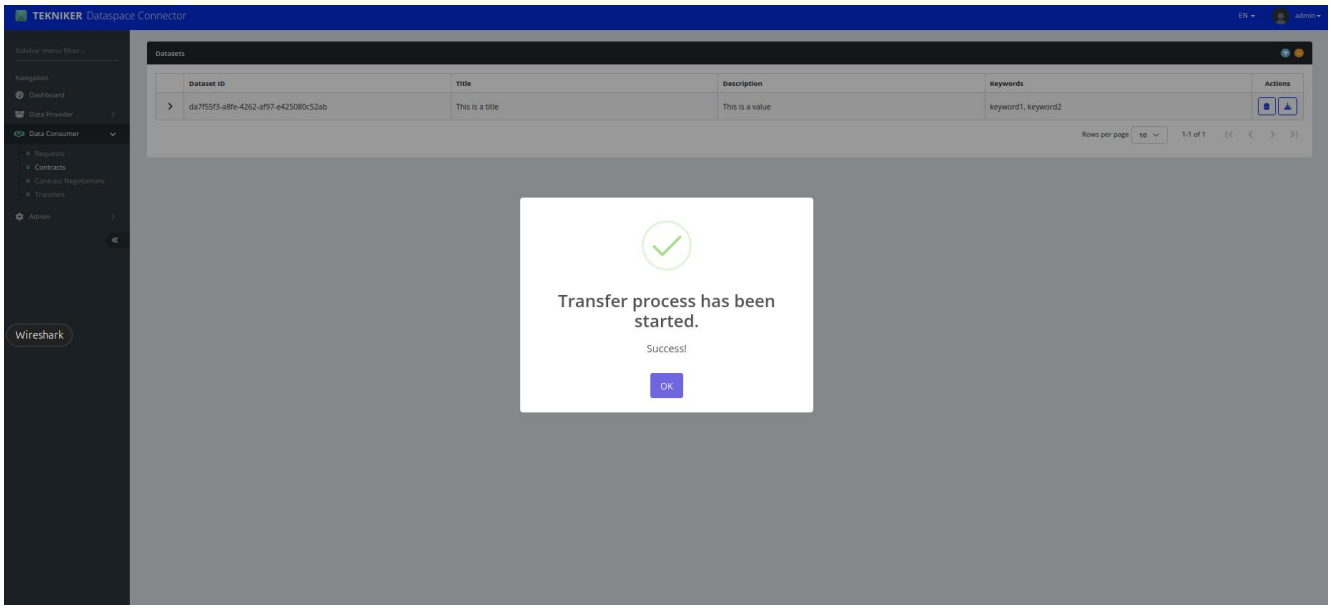


Figure 41: TDC UI – PULL Transfer Process request (2)

Once requested, the transfer process can be monitored. Figure 42 shows this through the TDC UI.

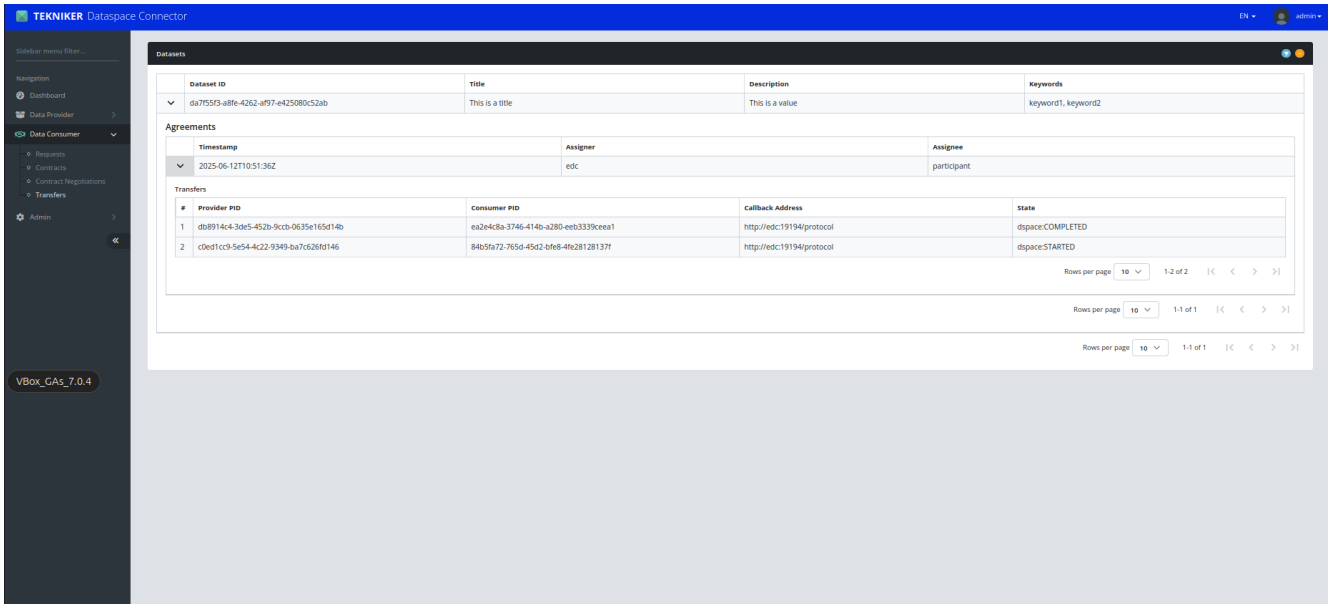


Figure 42: TDC UI – PULL Transfer Process state request

If a STARTED state is reached, data can be downloaded through the TDC UI. Thus, validating the compatibility of the Transfer Process Protocol between the EDC acting as a Data Provider and the TDC acting as the Data Consumer for the pull approach.

Push approach

To test the Transfer Process Protocol following the PUSH approach, a request to the EDC should be performed to trigger data sharing from the EDC to a determined processing tool. Figure 43 and Figure 44 show this request through the TDC UI.

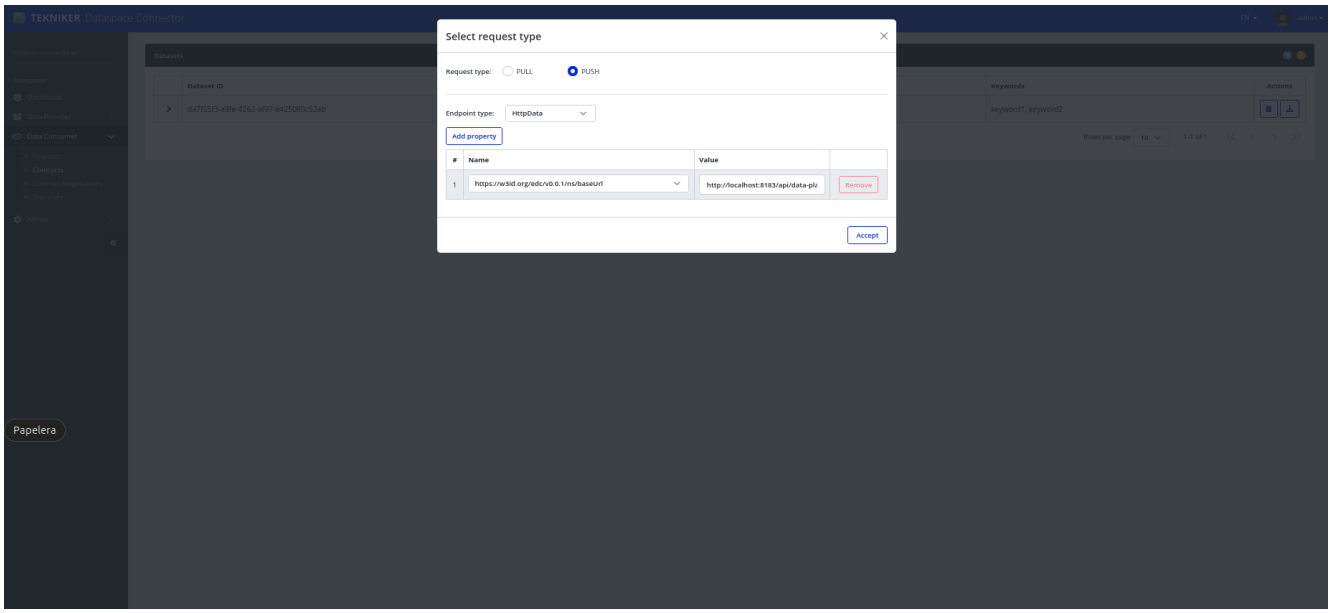


Figure 43: TDC UI – PUSH Transfer Process request (1)

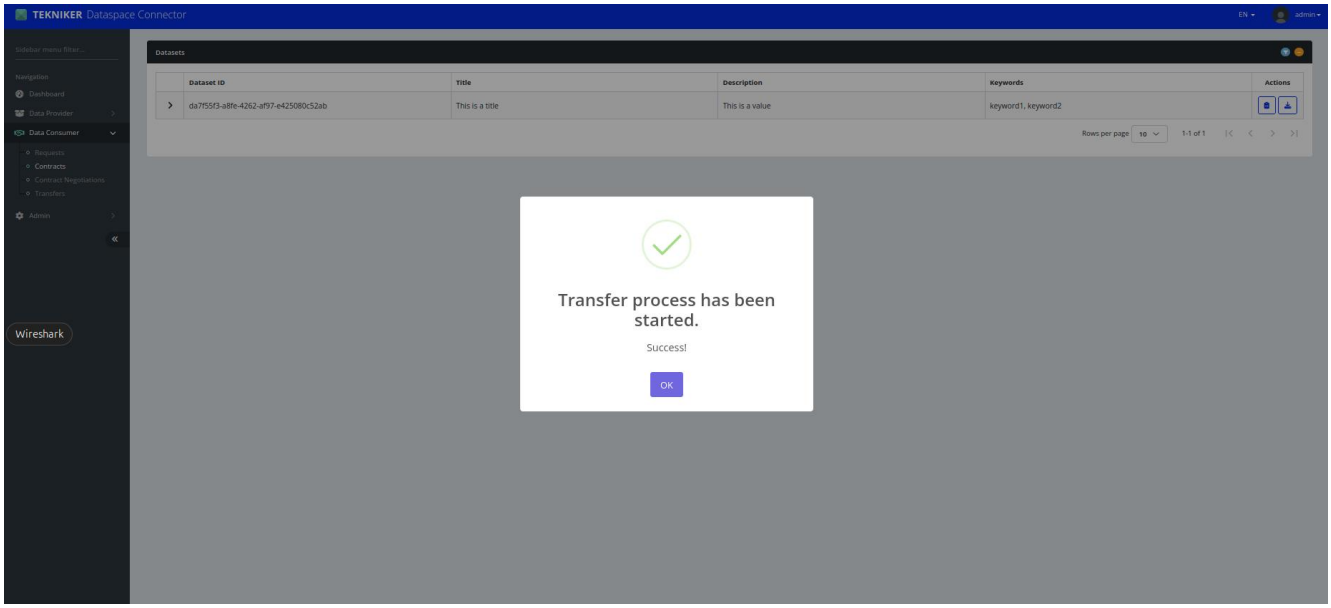


Figure 44: TDC UI – PUSH Transfer Process request (2)

Once requested, the transfer process can be monitored. Figure 45 shows this through the TDC UI.

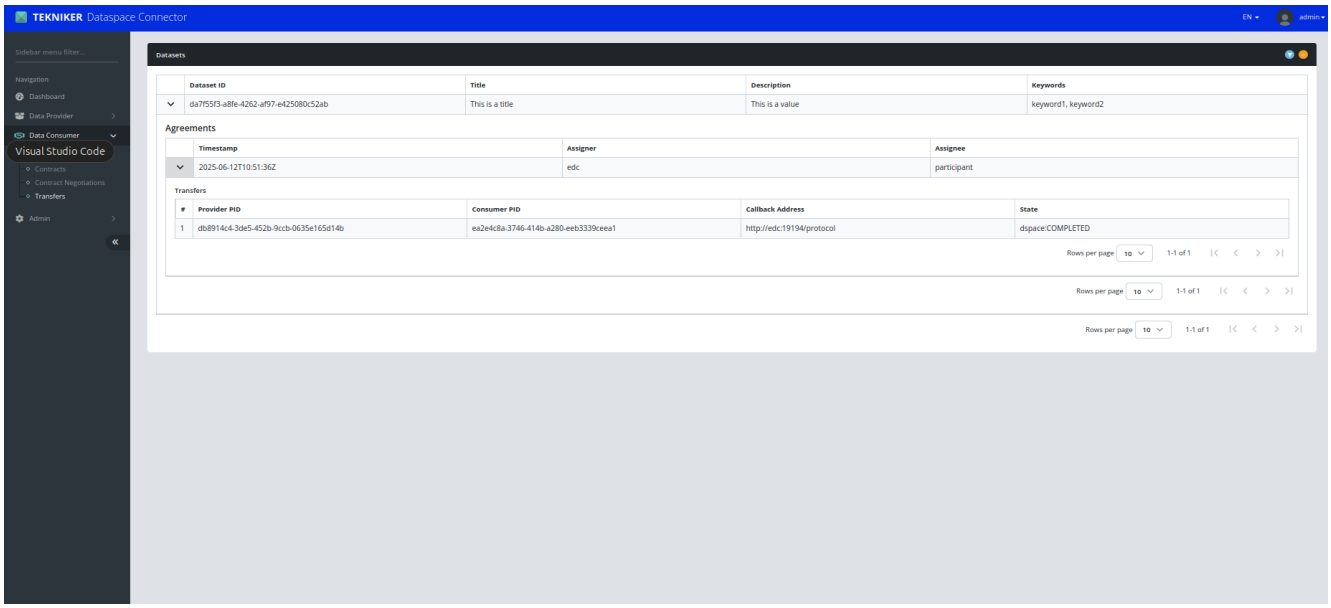


Figure 45: TDC UI – PUSH Transfer Process state request

A COMPLETED state means that data has been sent. Thus, validating the compatibility of the Transfer Process Protocol between the EDC acting as a Data Provider and the TDC acting as the Data Consumer for the push approach.

3.4 Challenges and Lessons Learned

A lesson emerged from the interoperability testing across AAS servers. Ensuring that the TDC AAS extension worked seamlessly with both FA³ST and Eclipse BaSyx servers required strict adherence to the AAS API specification. The successful outcome of these tests confirmed the importance of standards compliance in achieving cross-platform compatibility.

A significant technical hurdle was the lack of a TCK for the DSP, which is still under development. Without a formal validation tool, the team relied on manual testing against the EDC, the reference implementation. These tests, covering catalog discovery, contract negotiation, and data transfer protocols, were essential in verifying the TDC's compatibility with DSP. However, the absence of a TCK remains a limitation for full certification and long-term assurance. Once the TCK becomes available, the TDC will be validated against it to ensure full compliance with the DSP.

From a development perspective, extending the TDC's internal DCAT Java library to support AAS-specific metadata required careful architectural planning. The modular design, implemented using Maven, proved effective in managing complexity and ensuring maintainability. This approach facilitated the integration of new features without disrupting existing functionalities.

Finally, while the technical capabilities of the TDC and its extensions were validated, the usability of the system to navigate among the datasets emerged as a critical area for improvement. Future iterations should prioritize user interface enhancements and user feedback mechanisms to ensure that the tools developed are not only powerful but also accessible and intuitive.

4 AI Models Governance

4.1 Introduction (Problem Overview)

In recent years, the integration of Artificial Intelligence (AI) into digital Services has accelerated significantly. As services increasingly incorporate AI models into their execution, it becomes essential to ensure transparency, accountability, and governance over the AI components involved. Specifically, there is a need to document which are the AI Model Versions of the Service's AI Model, how they were configured, and which are the descriptive metadata of the datasets that supported their training and validation.

The absence of structured documentation and lifecycle management for AI models introduces several risks. These include the inability to reproduce service outcomes, difficulty in auditing or explaining decisions, and non-compliance with emerging regulatory requirements, such as those introduced by the EU AI Act. Without a clear framework for managing and tracing AI model usage, organizations face major challenges in ensuring responsible, trustworthy, and compliant AI deployments.no

4.2 Proposed Approach

At the first iteration of Tec4MaaSEs we have taken the concepts from Plooto [2] and Radioval [3] projects where relevant AI passports have been implemented. In T4M we have extended the use of current AI passport, through the incorporation of the AAS AI Model nameplate. In the second iteration we will create the AI passport as a service and integrate it in the marketplace platform.

The approach introduces a dedicated AI Model feature that enables Organisation Administrators to document and maintain detailed records of the AI models used within their digital services.

At the core of this approach is the integration of an AI Model management space within each Service Profile which is structured around key components that reflect the lifecycle of an AI model: Versions, Version History and Datasets. This ensures that every AI Model used during a Service's execution can be documented in a standardized and repeatable way. Each AI Model Version includes both system-defined fields (e.g., Version Name, Type, and Description) and customizable metadata defined by the Organisation Administrator, allowing flexibility while maintaining traceability.

Each AI Model Version can be linked to descriptive metadata about the Datasets used for training or validation or any other purpose defined by the Organisation Administrator. This association creates a clear record of how the AI Model Version was trained and validated, aligning with organizational governance needs and external regulatory expectations.

Version control is a key element of the proposed approach. Only one Version can be Active at a time, but multiple Versions with Status Draft can coexist. When a Version is activated, the previously active Version is automatically deactivated and stored in the Version History tab, ensuring clarity over which AI Model Version was active at any given time.

Looking forward, the approach is designed to be extensible to support future needs and evolving standards. Planned enhancements for upcoming iterations include runtime usage logging, which will record which AI Model Version was used during each Service execution, improving traceability and accountability. An

Evaluations tab will also be introduced, where AI Model Versions will be evaluated. In addition, a Documentation tab will be added, and the Organisation Administrator will be able to add technical, performance requirements and enrich this section by adding custom documentation fields. Another planned feature is the Declarations tab, where declarations related to the AI Model can be uploaded by the Organisation Administrator. A key enhancement in this roadmap is the introduction of AAS compliant AI Model Template. This template can be customized by the Organisations, allowing them to define reusable structures for AI Model Versions input, ensuring consistency in documentation and better alignment with AAS-compliant models and emerging regulatory frameworks. Collectively, these enhancements will strengthen the governance capabilities of the AI Model feature and support responsible, transparent AI adoption across the platform.

4.3 Implementation

The AI Model feature in the MIRA platform addresses the growing need for structured documentation, lifecycle management, and traceability of AI models used in Services execution. As AI becomes a central component of digital Services, Organisations must be able to track which AI Model Versions are active, how they were configured, metadata about datasets that supported them, and how they evolved over time. The AI Model feature provides a dedicated space within each Service Profile where Organisation Administrators can create, manage AI Model Versions, link them to descriptive metadata about the datasets used for each AI Model Version, and maintain an auditable Version History. This functionality supports transparency, accountability, and alignment with governance and compliance requirements.

The implementation of the AI Model can be accessed by navigating to Services, selecting the desired Service, navigating to the Service Profile and selecting the tab AI Model (see Figure 46, Figure 47 and Figure 48).

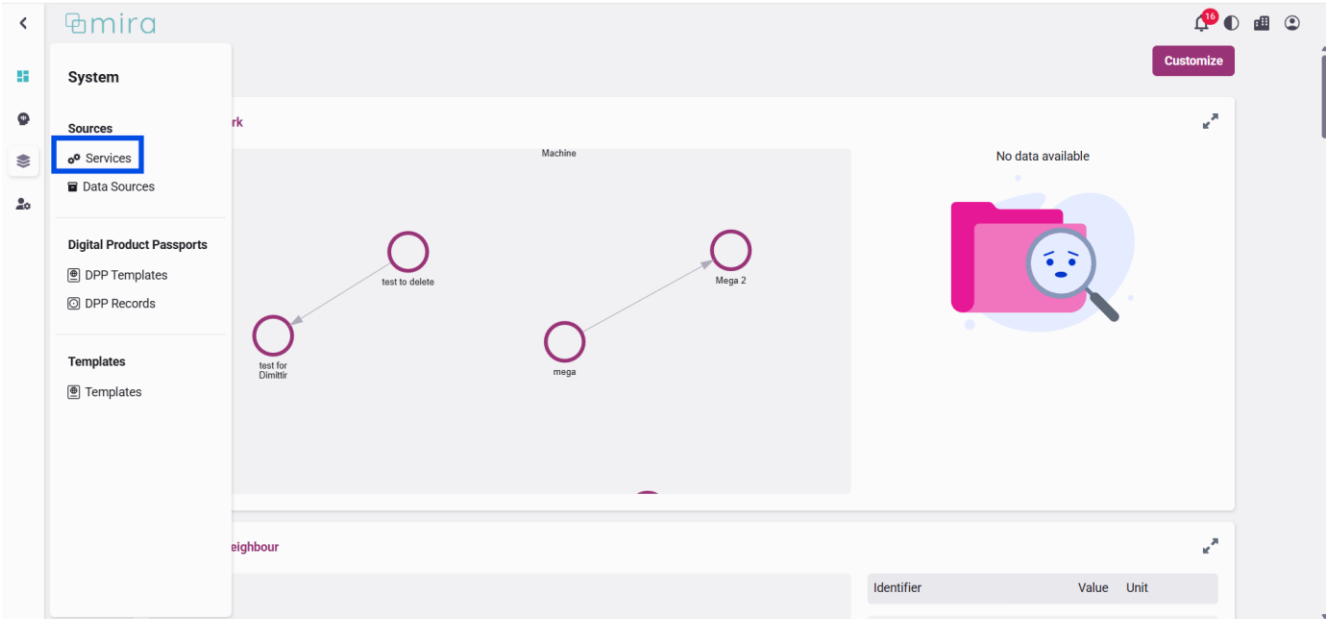


Figure 46: Accessing Services User Interface

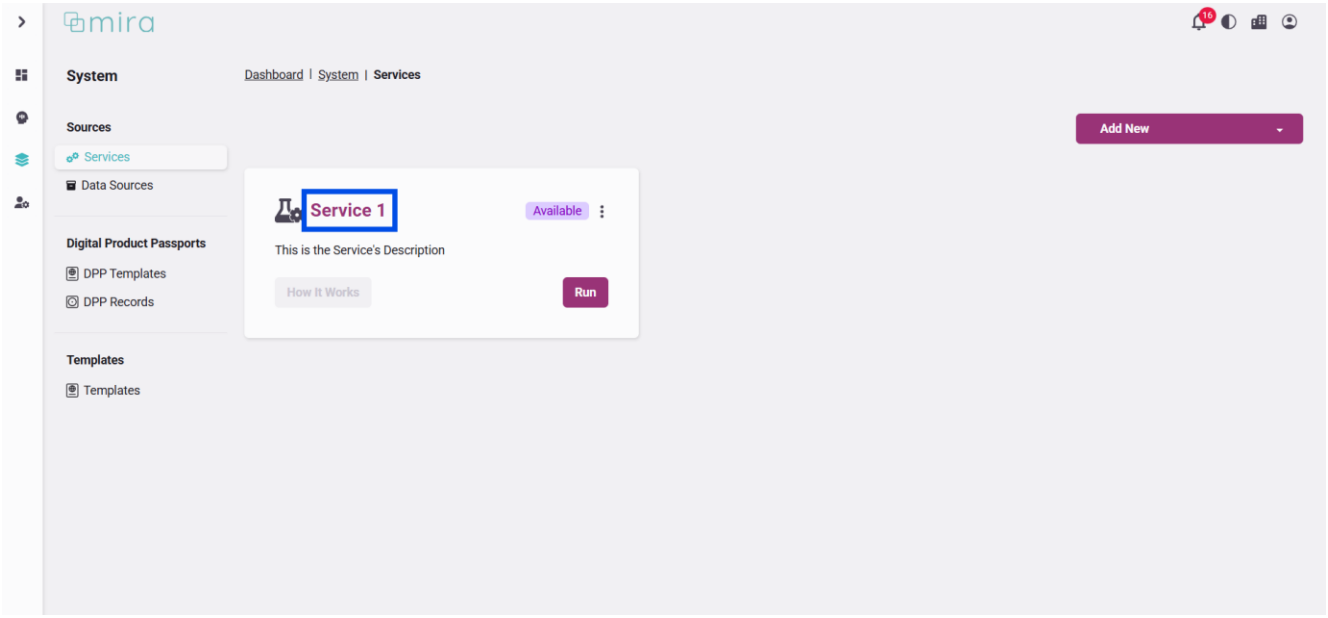


Figure 47: Available Services View

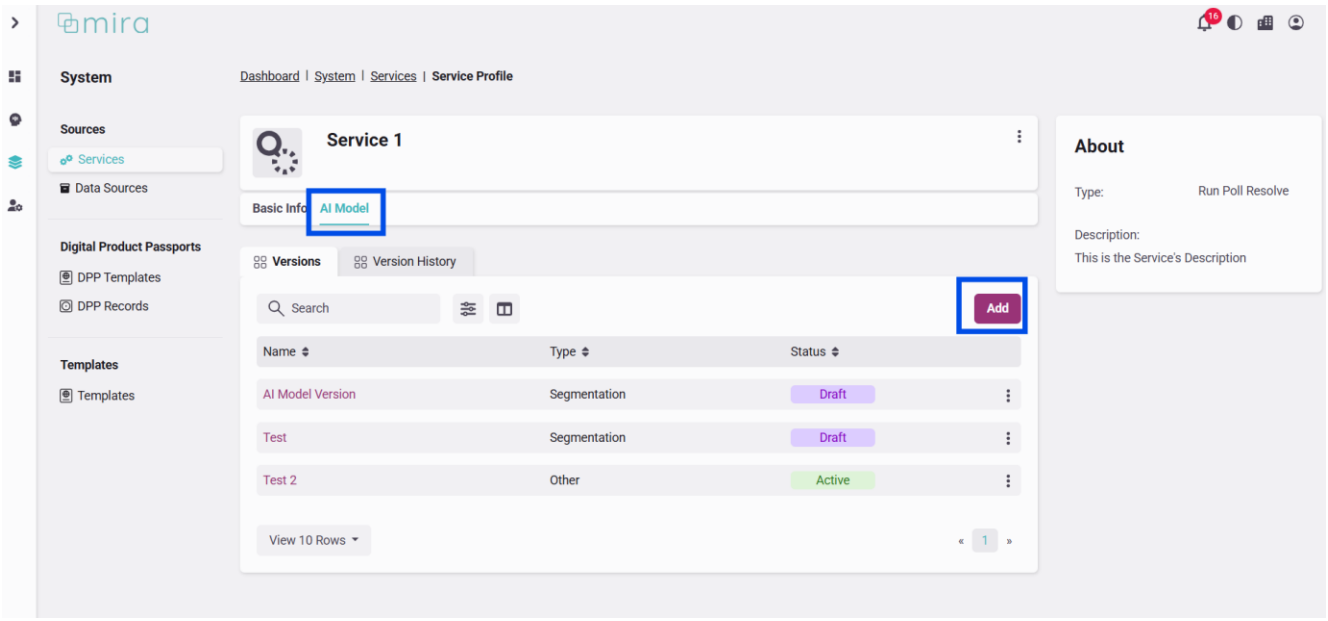


Figure 48: AI Model tab User Interface

As depicted in Figure 48, on the AI Model tab of the Service Profile, there is a table with all the available Versions of the AI Model with Status Active or Draft. Only one Version can be active at a time, but multiple Draft Versions can coexist. The Active Version can be edited, deactivated and deleted whereas the Draft Versions can be edited, activated and deleted.

A new AI Model Version can be instantiated by the Organization Administrator by selecting the Add button and entering Basic Information about the AI Model Version, including its Name, Version, Type and Description as shown in Figure 49. Upon clicking Next, the Org Admin can define Custom Fields to capture the Version's specific attributes (see Figure 50). Each Custom Field requires a Name, Type (string, numeric) and an Answer. Once the information is confirmed by clicking on the Confirm button, the Version is created with Status Draft.

All the fields that were filled in are displayed in the Overview tab of the AI Model Profile as shown in Figure 51.

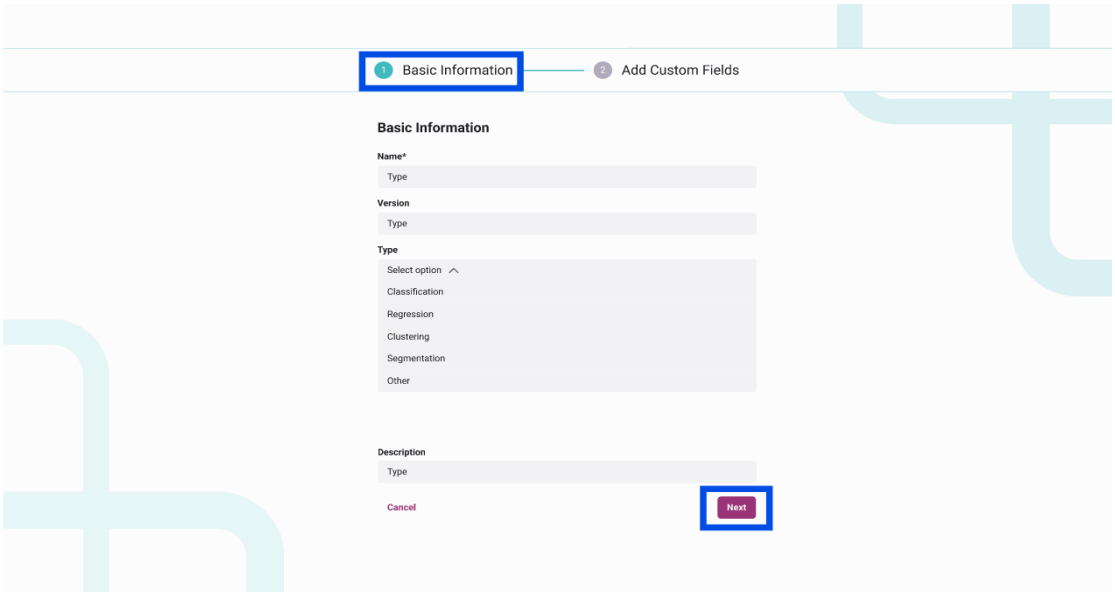


Figure 49: Filling in the Version's Basic Information

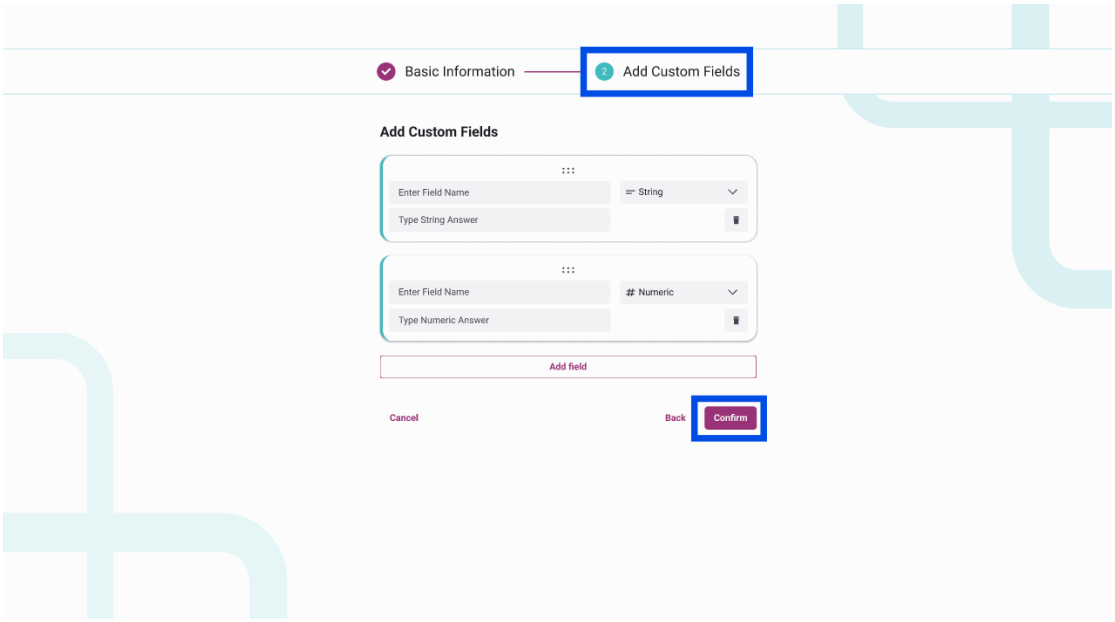


Figure 50: Filling in the Version's Custom Fields

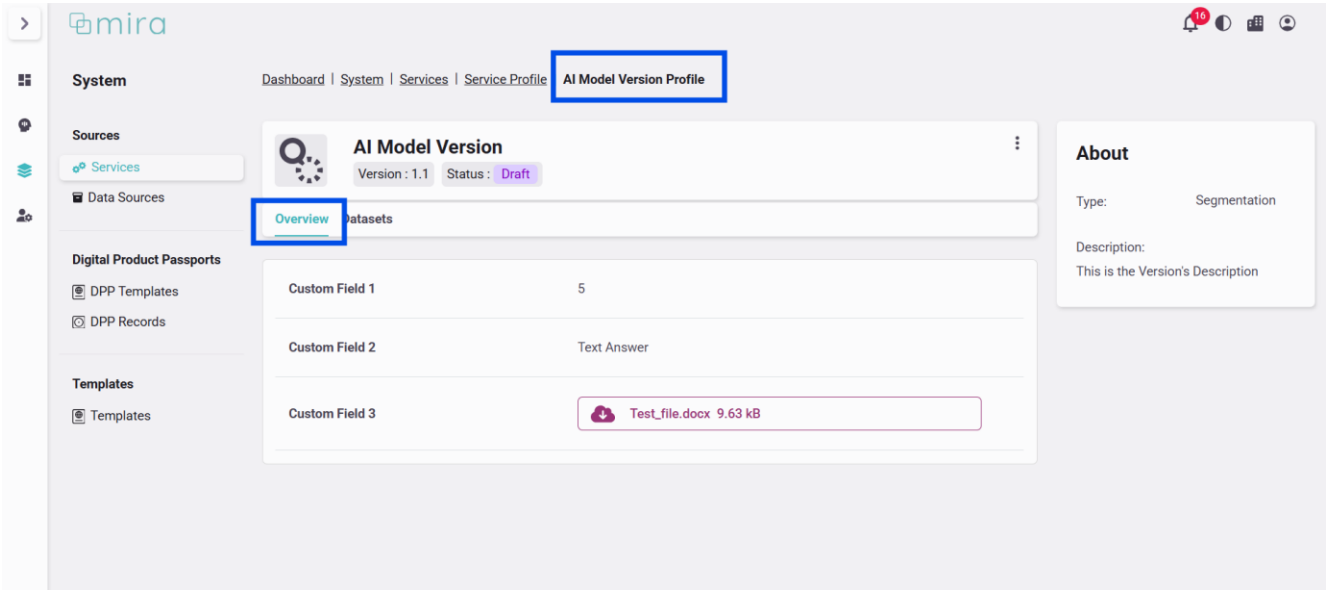


Figure 51: Overview tab of the AI Model Version Profile User Interface

The Version's Status can be set to Active by choosing Activate from the available actions (see Figure 52). When a Version is activated, the previously Active Version is automatically deactivated and transferred to the Version History subtab of the AI Model tab, as depicted in Figure 53. Deactivated Versions are read-only and can only be deleted.

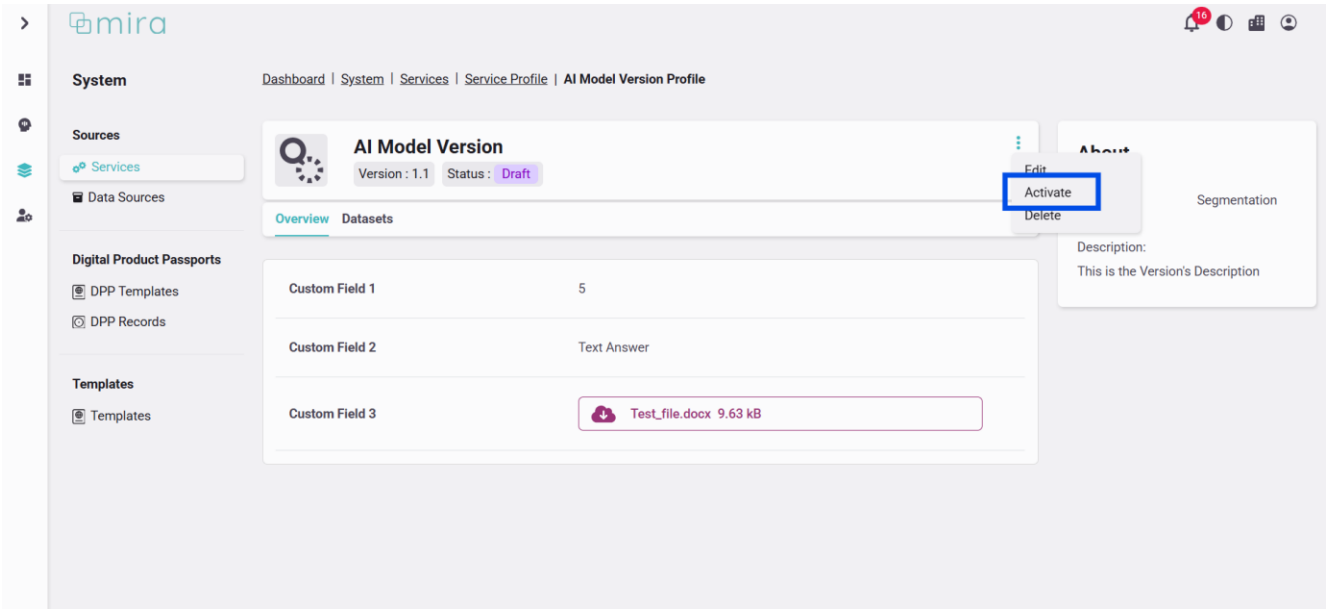


Figure 52: Activation of an AI Model Version

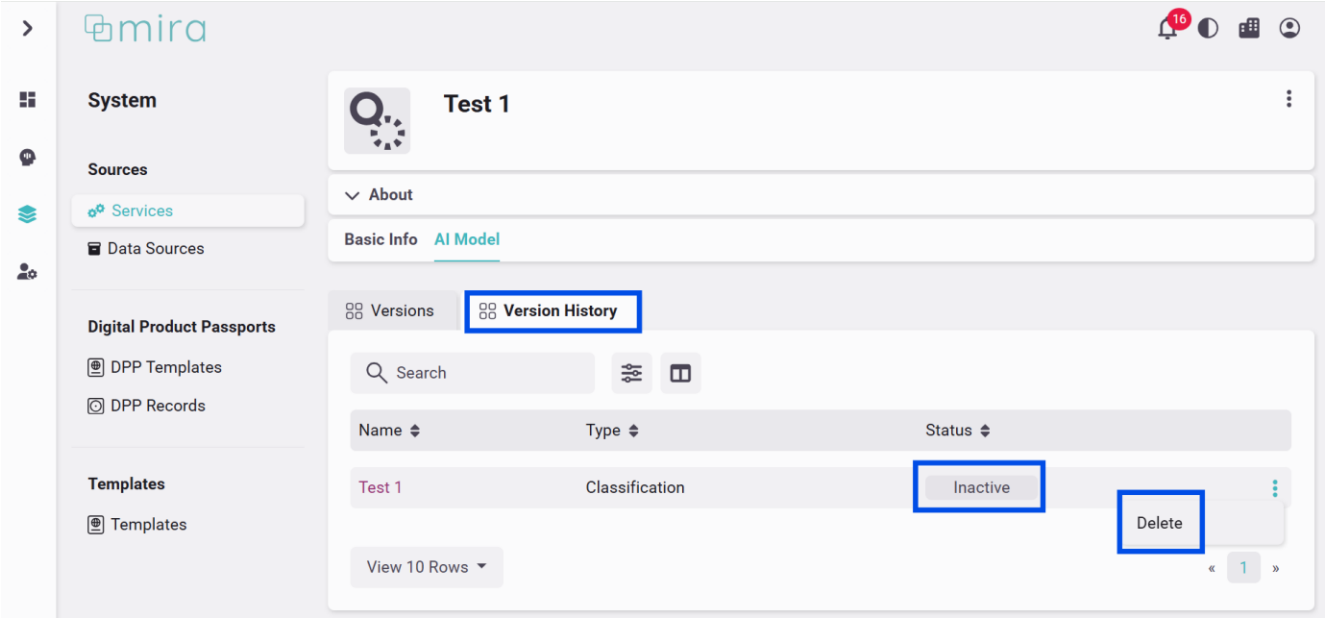


Figure 53: Version History User Interface

The Overview tab of the AI Model Version Profile, displays all the information that were entered during the creation of the Version, as shown in Figure 54.

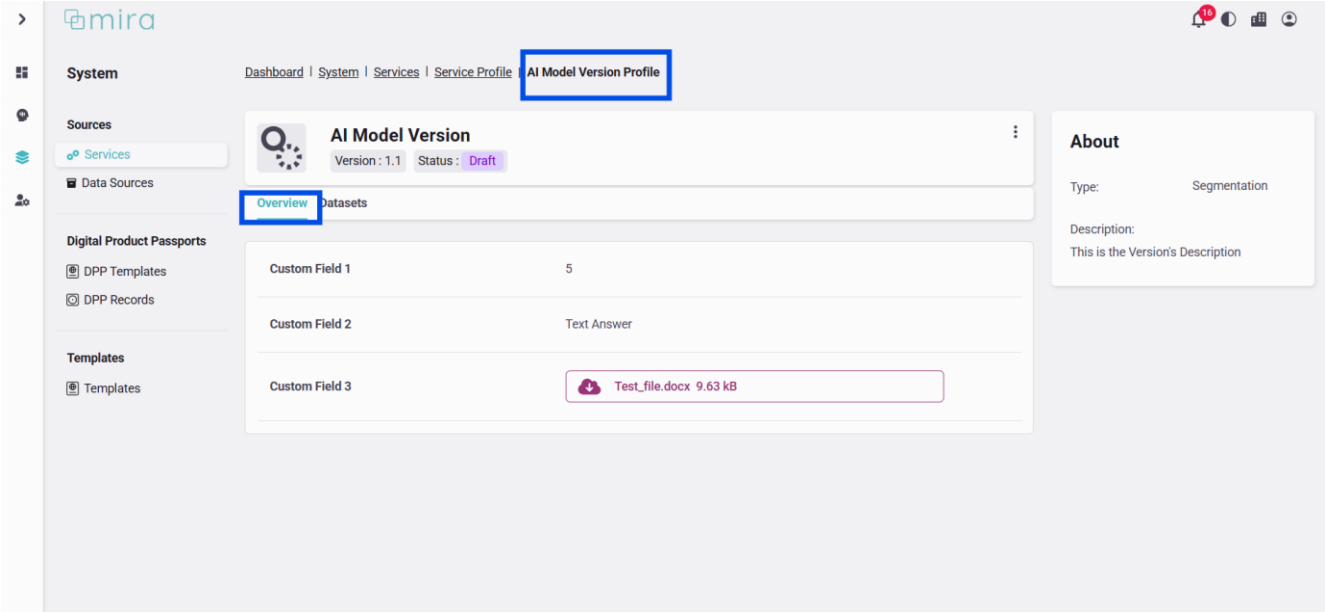


Figure 54: AI Model Version Profile's Overview Tab User Interface

In the Datasets tab, each entry on the table corresponds to descriptive metadata about the datasets associated with the Version, for training, validation or any other reason the Org Admin defines (see Figure 55).

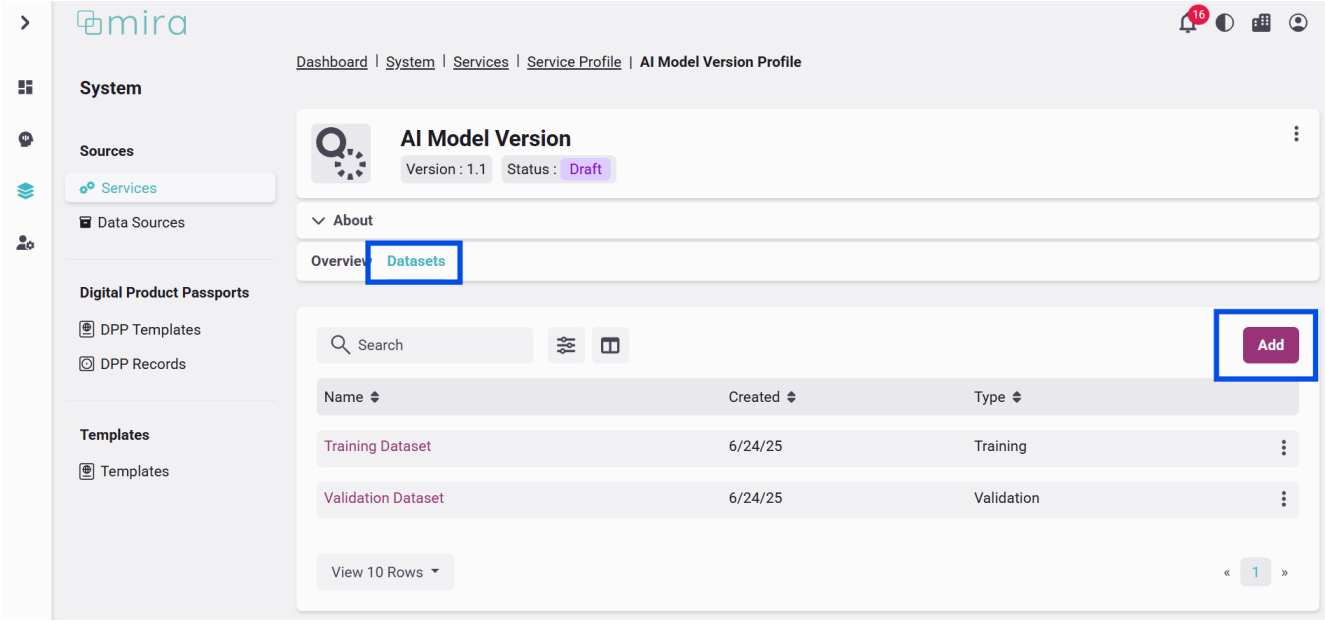


Figure 55: Datasets tab User Interface

A new Dataset entry can be instantiated by the Organization Administrator by selecting the Add button and filling in the fields and clicking the Confirm button as depicted in Figure 56.

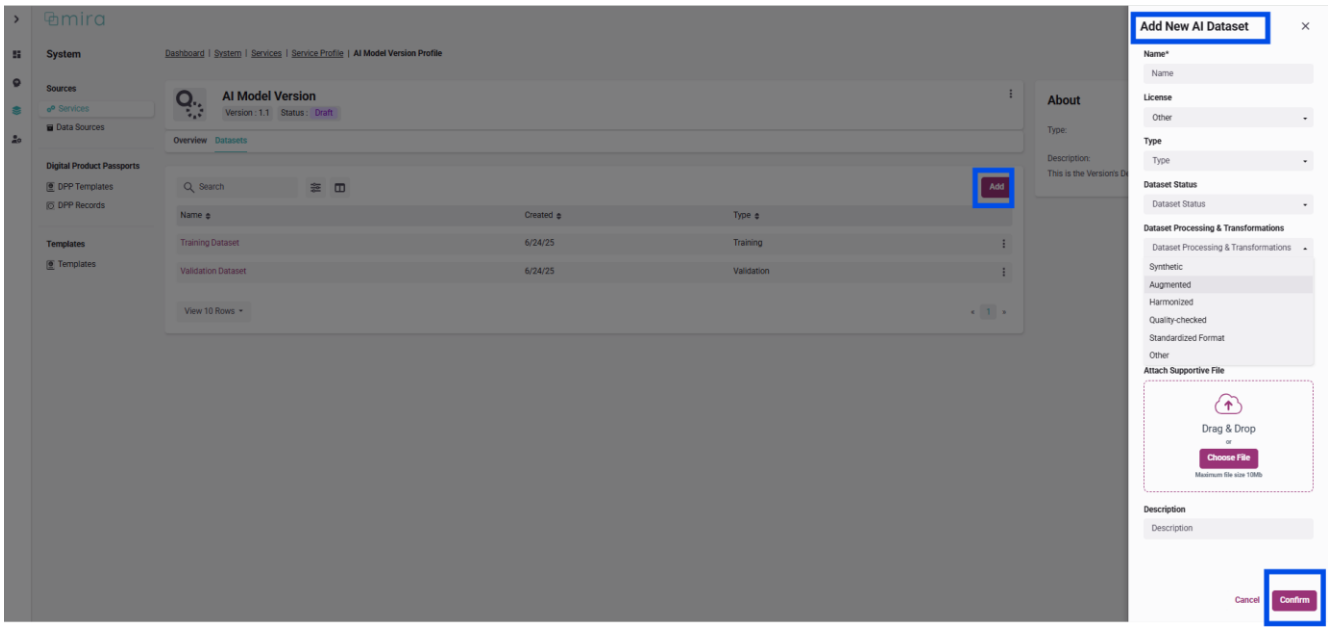


Figure 56: Adding Metadata Information about the Datasets used for the AI Model Version

When dataset entry is clicked on the datasets table, user is directed to the corresponding Dataset Profile, where all the dataset information is displayed (see Figure 57).

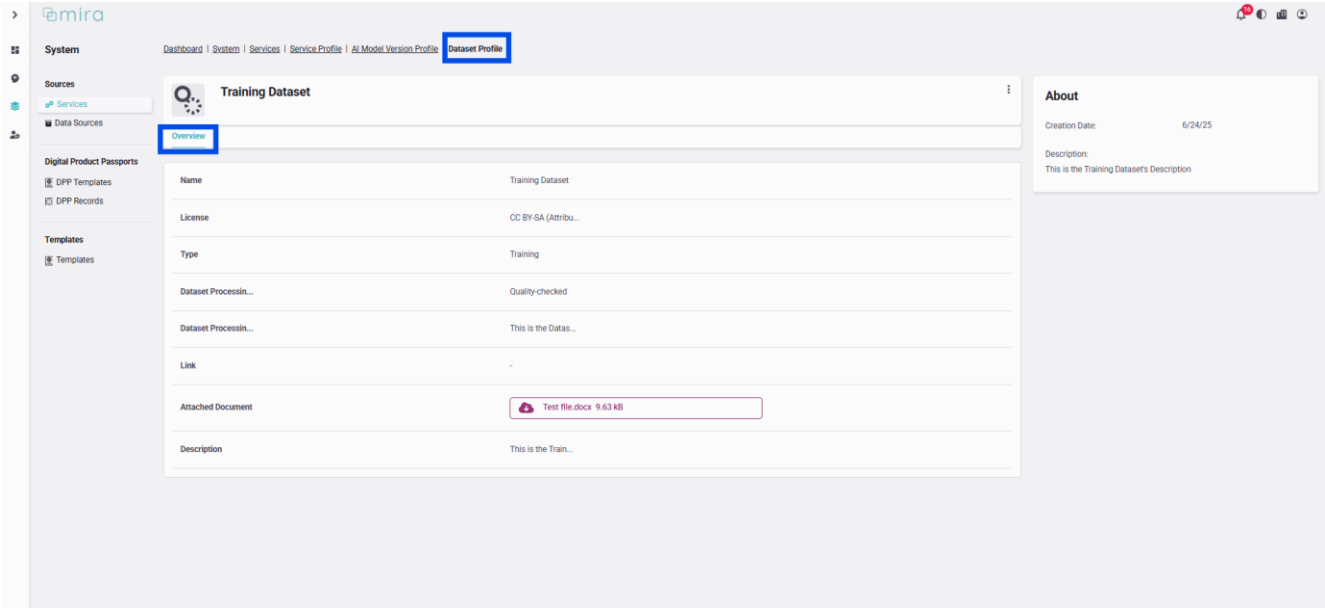


Figure 57: Dataset Profile User Interface

4.4 Challenges and Lessons Learned

The development of the AI Passport feature in MIRA introduced both technical and conceptual challenges that provided valuable insights. One of the primary challenges was designing a flexible approach that could accommodate varying documentation needs across different Services and AI Model Versions while remaining aligned with the principles of transparency and traceability.

Another challenge was the implementation of the Version's Statuses logic. Understanding the need to maintain a reliable history of deactivated Versions, having multiple with Status Draft and ensuring that only one AI Model Version could be active at any given time.

Also, a key challenge was ensuring that the user experience remained intuitive despite the complexity of the feature. The AI Model tab contains multiple sub-sections and will contain more in future iterations, each with its own logic and interaction patterns. Providing users, especially Organization Admins, with a clear understanding of how to manage AI Models without overwhelming them was essential. This required close collaboration between designers and developers to implement consistent UI patterns and ensure that critical workflows like adding a new Version or activating one were smooth and self-explanatory. These efforts emphasized the importance of user-centric design when implementing advanced functionality.

A key lesson learned was the need to apply the option of custom fields to indicate varied Service and AI Model Version needs. Also, the initiative highlighted the need to future-proof the system by anticipating emerging regulatory requirements, such as those outlined in the EU AI Act. Designing the AI Passport with extensibility in mind will help ensure that the platform remains compliant as AI governance standards evolve.

Conclusions

The concept of Governance in Manufacturing as a Service is thoroughly covered from different perspectives. From how the whole process of searching, negotiation, agreement is done, to the need of data control and gain trust on the applicable AI models/ services used in Tec4MaaSEs.

The project in its 1st iteration has developed prototypes for each service which will be tested by the pilots (1st pilot iteration) aiming to get feedback on their appropriateness and whether they cover the needs of the pilots.

The services will continue to be developed during and after the 1st pilot iteration according to the roadmap and including also feedback from the users during the initial evaluation phase. The project believes that governance should be placed on utmost importance because it deals with data, clarity and trust on the recommendation results and finally on automating the whole agreement process covering different needs and particularities defined by the pilots.

The governance services will be integrated within the whole Tec4MaaSEs solution and deployed to the pilots according to the deployment plan (WP5).

References

[1] Jacoby, Michael & Volz, Friedrich & Weißenbacher, Christian & Stojanovic, Ljiljana & Usländer, Thomas. (2021). An approach for Industrie 4.0-compliant and data-sovereign Digital Twins. at - Automatisierungstechnik. 69. 1051-1061. 10.1515/auto-2021-0074

[2] Plooto HorizonEurope Project. (n.d.). Retrieved from https://www.plooto-project.eu

[3] Radioval HorizonEurope Project. (n.d.). Retrieved from https://radioval.eu/