DELIVERABLE D4.1 – IDT TOOLKIT MODULES – DATA ACCESS - PHASE I

PROJECT ACRONYM:	DS2
PROJECT TITLE:	DataSpace, DataShare 2.0
GA NUMBER NO.	101135967
WEBSITE:	www.dataspace2.eu
DUE DATE OF DELIVERABLE:	2025-06-30
SUBMISSION DATE:	2025-06-27
LEAD BENEFICIARY:	IBM
LEAD AUTHORS:	Eliot Salant (IBM), Jens Schimmelpfennig (SWAG), Jarmo Kalaoja(VTT), Carlos Fernández Sánchez (INDRA), Alex McDonald (ICE)
REVIEWERS:	Tomaž Zadravec (ITC), Elias Dakos (ATC)
TYPE:	OTHER
DISSEMINATION LEVEL:	PUBLIC

DISCLAIMER

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or granting authority. Neither the European Union nor the granting authority can be held responsible for them.

STATEMENT OF ORIGINALITY

This Deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

DOCUMENT HISTORY

VERSION	DATE	DESCRIPTION	NAME	ORG
V0.1	2025-05-21	Table of contents	E. Salant	IBM
V0.1	2025-05-21	WPL Sections Initial	E. Salant	IBM
V0.1	2025-05-30	Module Added	Module Owners	Various
V0.2	2025-06-05	WPL Final sections and editing	E. Salant	IBM
V0.3	2025-06-15	Comments by Reviewer 1 Processed	E. Salant	IBM
V0.3	2025-06-24	Comments by Reviewer 2 Processed	E. Salant	IBM
V1.0	2025-06-26	Final Document	E. Salant	IBM

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This document describes the work performed in DS2 Work Package 4 until M18 of the project.

O4: Create the framework tools necessary to create and maintain the transfer of data across distributed, disparate data spaces.

- O4.1: To provide the tools to help orchestrate and govern the exchange of vast volumes of data between enterprises and sectors, across geographically distributed data stores and frameworks, meeting lifecycle, performance, sovereignty and security requirements
- O4.2. To detect different types of disruptions to data workflows through AI-driven anomaly detection
- O4.3. To develop the technologies required for interoperability of data across data stores via transformation tools to support data portability

WP4 concentrates on actions around the obtaining and utilization of data between a data space Consumer and Provider.

There are many facets to this challenge, including:

- Providing the mechanisms to implement and extend fundamental data space protocol, such as policy enforcement and a federated Catalogue
- Improving data consumability through:
  - Syntactic conversion to handle differences in source and target schemas.
  - Data quality inspection.
  - Data federation across disparate data sources.
- Infrastructural
  - Orchestration of services including data services and services from other modules
  - Simplifying the data retrieval process for consumers.

The formal WP deliverables are all software (Type: OTHER), so the software, possibly source code, documentation, detailed progress tracking are on the DS2 GitHub which will evolve and grow in content over time. Within this document an introduction to the WP is provided, followed by a brief description of the different modules developed. There is a summary of the progress for each module, including links to the DS2 repository. In addition to this some WP have other non-module and/or non-software activities which are also reported where applicable. This is then followed by a description of KPI's, Primary Risks, and Primary Issues for the WP, followed by a Conclusion section.

In summary, there are few significant risks and issues, and all software is on track.

1 INTRODUCTION

The WP4 suite of software provides a combination of modules, some of which are required for a minimally configured DS2 system and others which give extra value to a data space environment.

This includes modules for the below which are documented in Section 3:

- DS2 Orchestration Module ORC
- DS2 Data Retrieval Module RET
- DS2 Data Curation Module CUR
- DS2 Policy Agreement and Enforcement Module PAE
- DS2 Data Detection and Transformation Module DDT
- DS2 Model Development Toolkit MDT
- DS2 Catalogue CAT

The technical partners involve are IBM, INDRA, SWAG, DIGI, VTT and i4RI. Note that originally ICE had resources here but agreement was reached to swap some activity between ICE/i4RI and WP4/6.

1.1 Document Structure

- Section 1: Introduction
- Section 2: MModule Description and Overall Status
- Section 3: Software progressSoftware progress
- Section 4: Additional Activities
- Section 5: KPI, Risks, and Primary Issues
- Section 6: Conclusion

1.2 Glossary and Abbreviations

A definition of common terms related to DS2 as well as a list of abbreviations, is available at https://www.dataspace2.eu/results/glossary

1.3 External Annexes and Supporting Documents

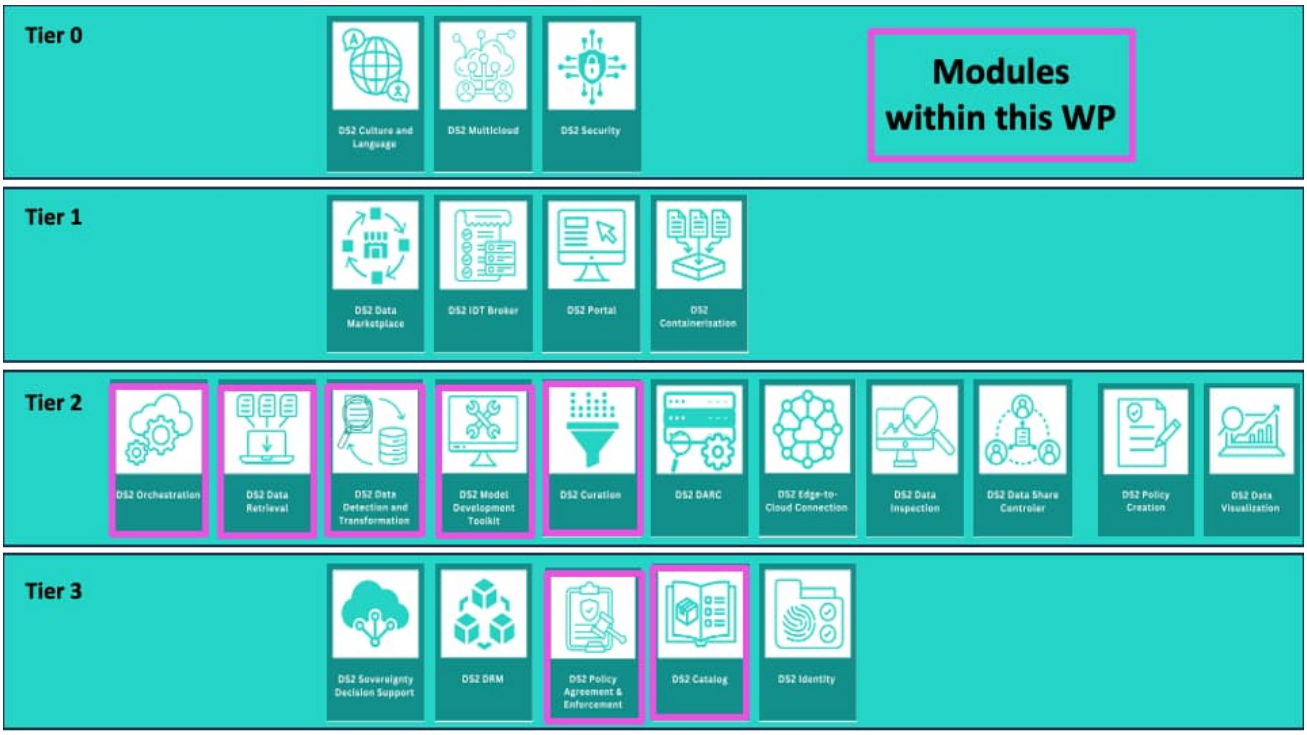
External Documents:

- DS2 D2.2 Requirements, baselines, KPIs, Architecture & Specifications
- DS2 Communication, Dissemination, and Exploitation report (M18)
- Risk identification spreadsheet

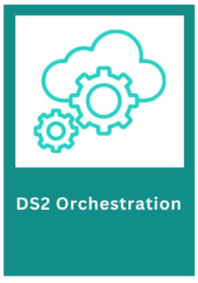
2 MODULE DESCRIPTION AND OVERALL STATUS

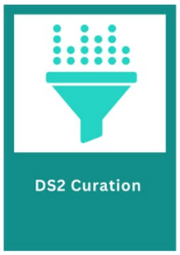
2.1 Modules

This WP has the following modules:



This WP has the following modules:

Module	Purpose
Tier 2	
ORC 	To design and then orchestrate at runtime In-Dataspace, Inter-Dataspace, internal, and third-party services which facilitate common data-orientated operations such as transformation of data, checks on data, data updates etc. The Orchestrator contains a flexible GUI to design workflows and decision points on these services and run time component to implement the workflow.
RET 	A DS2 data catalogue for a data space can potentially reference a large number of data sources accessed by different backend applications. All requests for data must go through a data space connector both at the consumer and producer side. For a connector to access stored data, the connector must generate the required API code to communicate with the storage backend. This can represent a major learning and programming effort even for an experience programmer and is certainly not a task which the average non-technical data space consumer can easily take on. It is the goal of the Data Retriever Module to vastly simplify the effort required to access a data source by automatically generating the required REST call for the connector through machine learning.
DDT	Dataspaces allow for data to be shared between data providers and data consumers. A lot of data comes from sensors and devices at a high rate. To allow for a well-defined data structure and

	quality during the data generation and exchange, the DDT is a module that can analyse data on-the-fly.
MDT 	The main purpose of the DS2 Model Development Toolkit Module (MDT) is to provide a set of tools to allow the users to develop algorithms based on the CRISP-DM standard to assist in the whole development cycle (training, test, etc.) and package the algorithms which can be deployed as an executable software component.
CUR 	Data obtained from disparate sources runs the risk of remaining siloed unless it is curated to match the format of similar data from other data sets. Manual curation of datasets, however, can be a labour-intensive task, and not suited to DS2's dynamic nature of federating dataspaces. The aim of this task is the automatic creation of pipelines to curate data through machine learning. The Data Curation Module is invoked on one or more data sets and uses machine learning to allow for natural language queries across the data sets – seamlessly joining data sets and performing conversions between column names and formats where required.
Tier 3	
PAE 	The primary function of the Policy Agreement and Enforcement Module (DS2 PAE) is to ensure compliance with the established policies and regulations governing data exchange among users in different data spaces. Henceforth, policies, regulations, and agreements are synonymous with the term "policy". The policies are evaluated as the control plane stage of data sharing in the Connector. The policies serve two main purposes: Access Control and for Usage Control. Access Control determines whether access to data is granted or denied. Usage Control dictates how the data can be used once access is granted.
CAT 	The Catalogue Module is a Module designed to support the exchange of data within and across different data spaces. It ensures robust data governance, secure data exchanges, and compliance with sovereignty requirements. The main goal of DS2 catalogue is to enhance the functionalities of catalogue systems within existing reference architectures, enabling them to support both intra-data space and inter-data space operations. This includes defining data models for data product offers, data product offer searches, and interactions with members of other data spaces, thereby fostering collaboration across different data spaces.
Tier 2	

The module fit is as follows:

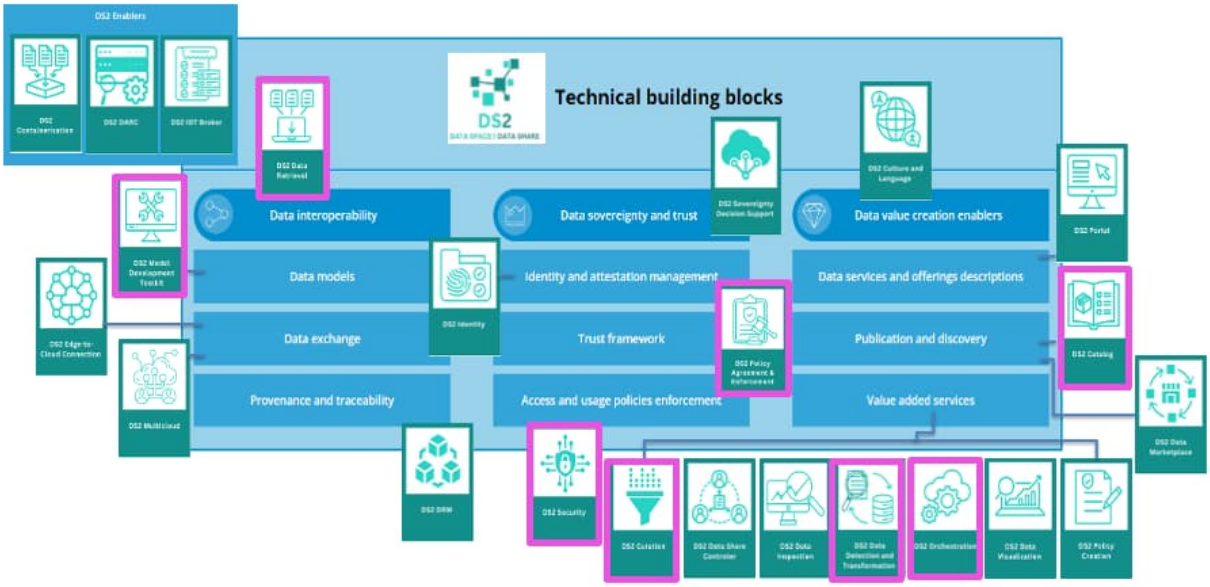


Figure 1. Module Fit

2.2 Status

Module	Task	Partners	License Type	Software Status @ M18	Estimated Completion Month	How to Install "vs 0.1" at M18	How to Configure "vs 0.1"	Marketing Video	PORTAL Helm Char	Marketplace Entry
OCR	T4.1	ICE	Open Source Apache 2.0	40%	M24: 70% - Sufficient for 1stValidation M30: 100%	On Github	By education session	M30	M24	M30
RET	T4.1	IBM	TBD	80%	Currently working. Functionality to be extended	On Github	On Github	M30	M24	M30
CUR	T4.3	IBM	TBD	70%	Currently working. Functionality to be extended	On Github	On Github	M30	M24	M30
PAE	T4.1	INDRA	Apache 2.0	40%	M30	On Github	By education session	M30	N/A (To be bundled with the connector)	
DDT	T4.2	SWAG	Apache 2.0	40%	M30	On Github	By education session	M30	M24	M3
MDT	T4.3	INDRA	Apache2.0	40%	M30	On Github	By education session	M30	M24	M30

CAT	T4.3	VTT	TBD	Consortium confidential on Github (not including DKAN core).	Currently working. Functionality to be extended	On Github	On Github	M30	M24 (excluding DKAN core)	M30

2.3 Major Changes/Deviations impacting this WP

- Partner change - SWAG. As per the periodic report, SWAG will leave the project and its activity will be taken over by Blue Bridge (Pending Amendment and EU approval). Since the same personnel will be involved, and the IPR is open-source, no issues are foreseen.
- Work on "transferring huge amounts of data" from T4.1 overlapped with T6.1. Since the same partner (DIGI) is involved in both efforts, this work was transferred to T6.1.
- Partner i4RI took over the ORC activity from ICE, and with ICE spending those resource in WP6

3 SOFTWARE PROGRESS

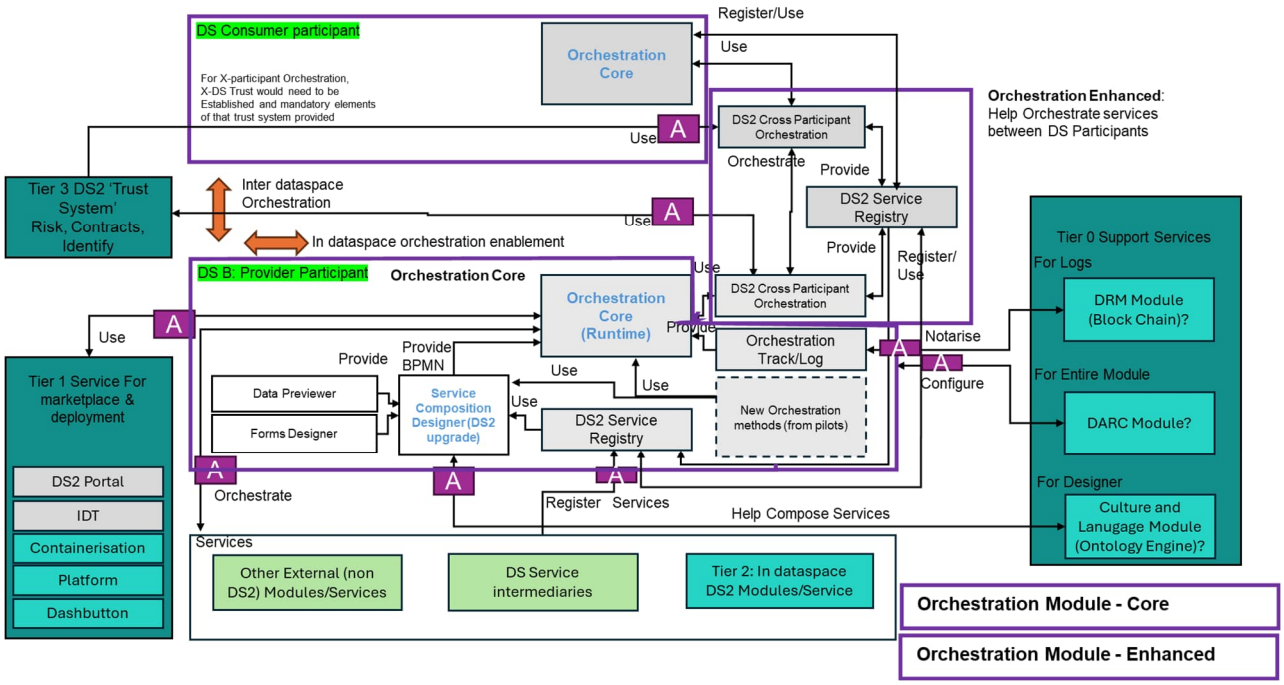
The software, documentation, and progress for the modules developed is located in the DS2 GitHub repository accessible by the links below. These links give the current module documentation and at the top further links to the software and module progress. By plan, at this interim state of the project, the modules are still under development and documentation is limited to essential information. The framework for documentation and progress monitor is identified in Annex A of the WP6 equivalent deliverable "DS2 D6.1 - FEDERATED IDT PLATFORM - PHASE I". Progress on module development is monitored through two-weekly sprints with detailed highlights.

Module documentation will evolve as the project progresses.

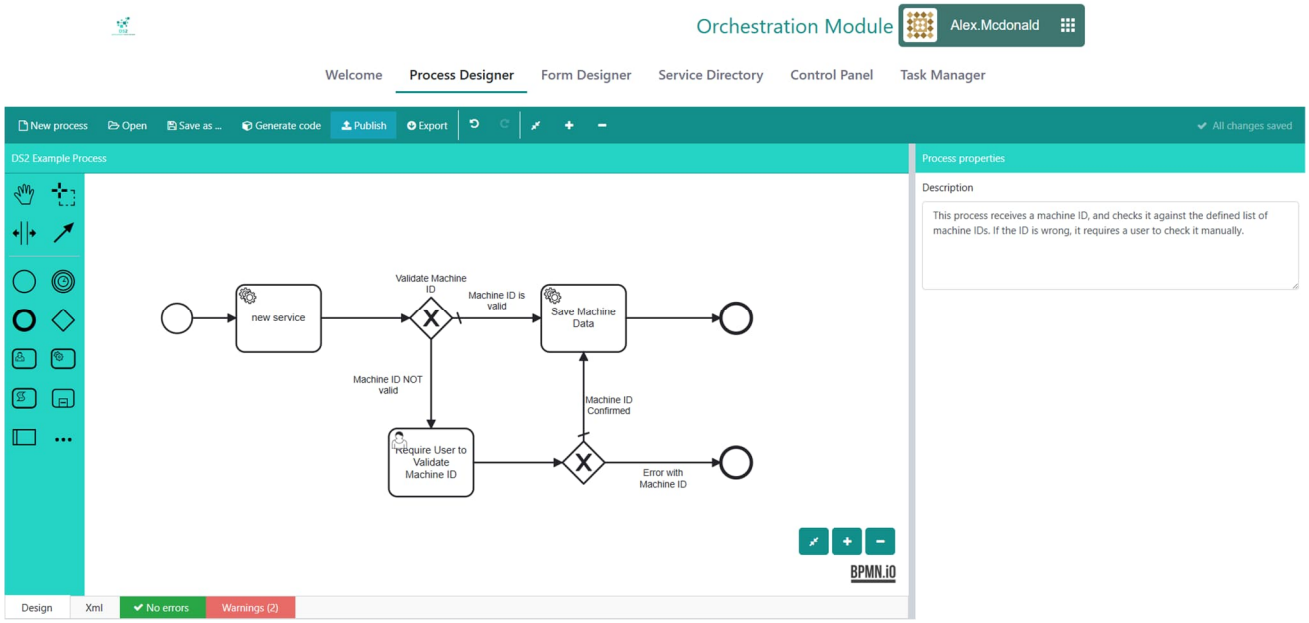
Module	Link
ORC	https://ds2-eu.github.io/documentation/modules/ORC/
RET	https://ds2-eu.github.io/documentation/modules/RET/
CUR	https://ds2-eu.github.io/documentation/modules/CUR/
PAE	https://ds2-eu.github.io/documentation/modules/PAE/
DDT	https://ds2-eu.github.io/documentation/modules/DDT/
MDT	https://ds2-eu.github.io/documentation/modules/MDT/
CAT	https://ds2-eu.github.io/documentation/modules/CAT/

3.1 Module: ORC - Orchestration

3.1.1 Architecture Diagram



3.1.2 Sample Interface



3.1.3 Primary Feature Progress

Functionality	0	1	2	3	4	5	6	7	8	9	--	M18	M24	M30	M36
Orchestration Core (Runtime)	25	25	25	25	25	25	25	50	75	100	--	100	100	100	100
DS2 Service Registry	50	50	55	55	60	60	60	80	100	100	--	100	100	100	100
Service Composition Designer (DS2 Upgrade)	25	25	55	55	55	55	55	75	100	100	--	100	100	100	100
Forms designer upgrade	0	0	5	5	5	5	10	15	15	15	--	0	100	100	100
Data Previewer	0	0	0	0	0	0	0	0	0	0	--	0	100	100	100
New Orchestration Methods orientated to data flow (from pilots)	0	0	5	5	5	5	5	5	5	5	--	0	0	100	100
Orchestration track/log	0	0	0	0	0	0	0	0	0	0	--	0	100	100	100
Blockchain and Blockchain API	0	0	0	0	0	0	0	0	0	0	--	0	0	100	100
DARC and DARC API	0	0	0	0	0	0	0	0	0	0	--	0	0	100	100
Ontology Engine and Ontology API	0	0	0	0	0	0	0	0	0	0	--	0	0	100	100
Dash Button	50	80	85	85	85	85	85	90	100	100	--	100	100	100	100
Tier 3 Trust Stack & Tier 3 API	0	0	0	0	0	0	0	0	0	0	--	10	100	100	100
DS2 Registry Service	0	0	5	5	5	5	5	5	5	10	--	10	100	100	100
DS2 Cross DS Orchestration	0	0	5	5	5	5	5	5	5	10	--	10	40	100	100

3.1.4 Activity

In the first few months of development, the team focused on establishing the foundational documentation for the software, with particular emphasis on the architecture and requirements documents (Deliverable 2.2). This initial phase helped set the stage for development work, which began with adapting existing components to the DS2 framework. Alongside this, research into data-related services was started, particularly focusing on service orchestration for data flows and the creation of custom forms for handling data interactions. Another key task during this period was adapting the Orchestration UI to fit the DS2 design style.

As the project progressed, the Dash Button integration was finalized, and the DS2 theme was applied comprehensively across the system. One major achievement was the complete replacement of default Bootstrap elements with custom components in the Service Registry, aligning the interface with DS2 standards.

Furthermore, the team began exploring how the Orchestration Module could be utilized across multiple data spaces, enabling the potential for cross-dataspace and cross-participant service interactions. This involved investigating how data and services could seamlessly be shared and accessed across different platforms and participants. A significant technical issue between the Dash Button and Keycloak was identified and resolved, further stabilizing the system.

In addition to ongoing development, the team dedicated efforts to maintenance tasks and began evaluating the necessary UI changes required for future platform expansions. Initial research into using the Orchestration Module across different data spaces was also carried out, exploring how it could function in a multi-dataspace environment. This exploration aimed to ensure the system could meet the needs of a broader range of data interactions as the project developed.

During the preparations for the Darmstadt plenary demonstrations and the weeks following, improvements were made to the Service Registry front end, ensuring a more user-friendly experience.

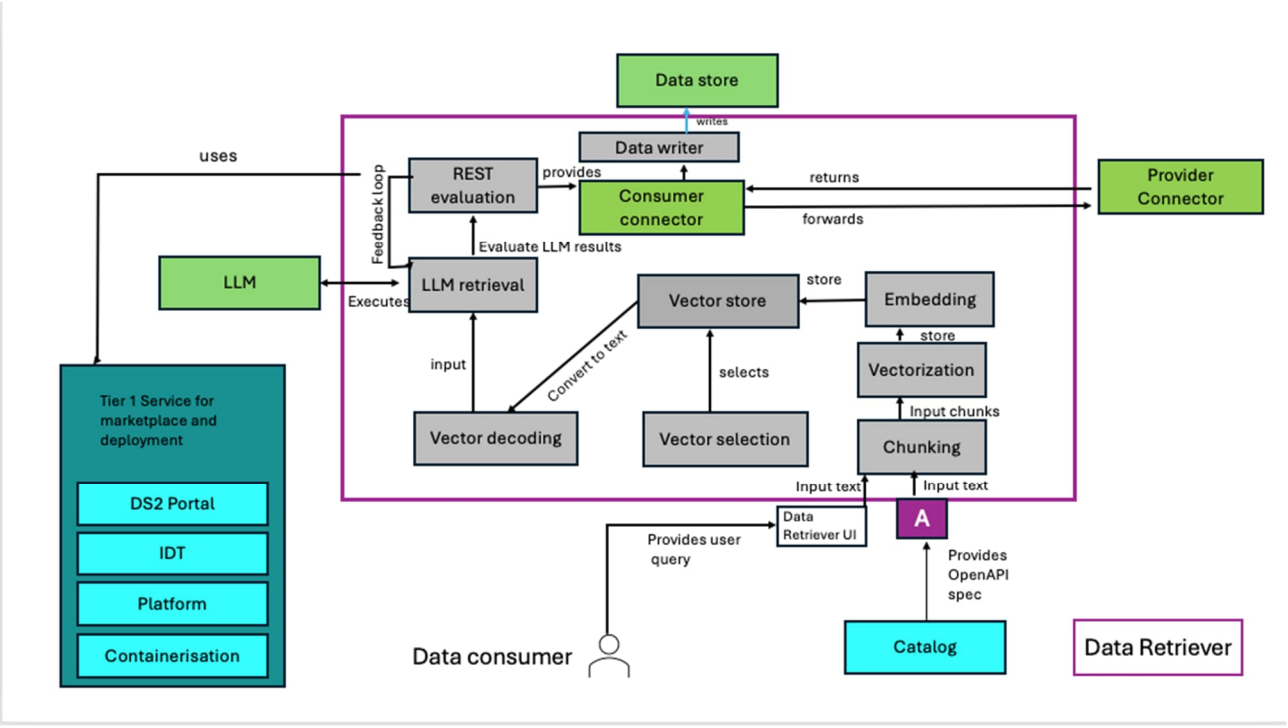
The Welcome page of the ORC module was also updated, serving as both an introduction to new users and a user manual. It now offers guidance on how to navigate and use the module, making its features and functionality more accessible to a wider audience.

3.1.5 Use Case Validation Plan

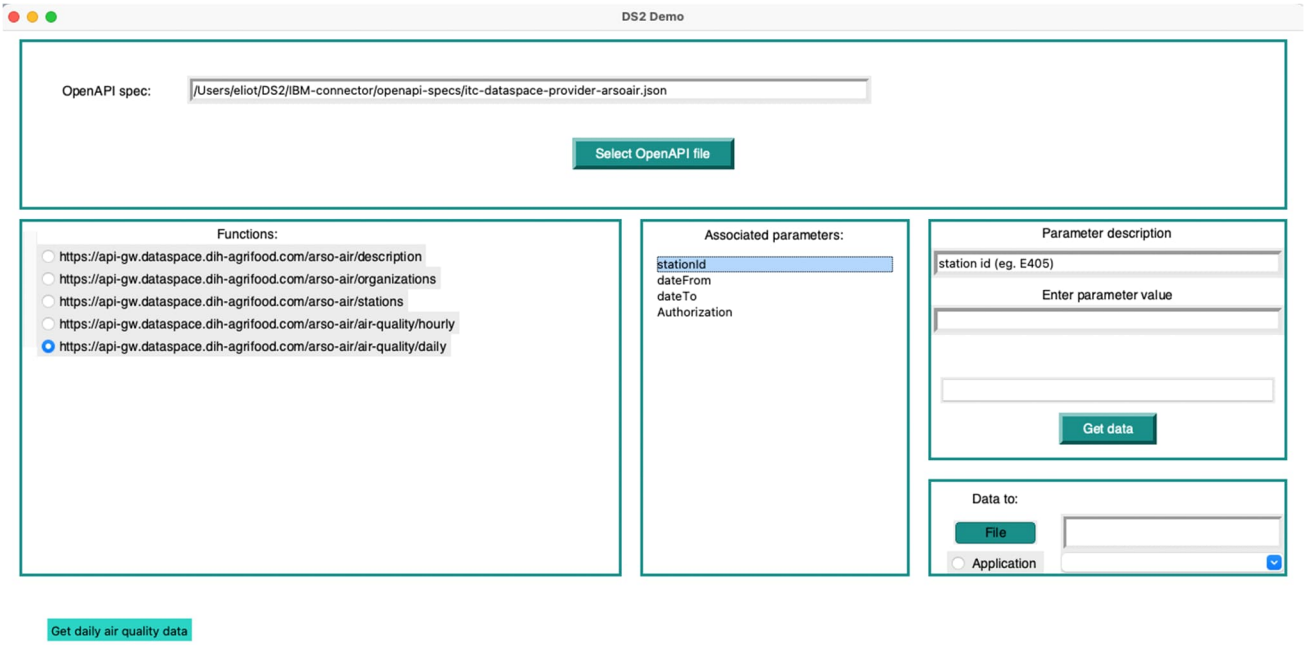
	KEY:	Default inst7all, UC = Validation	Specific To Case UC = Validation
Module ID	**Precision Agriculture**	**Green Deal**	**City Scape**
ORC	UC3.1, UC3.2, UC3.3		UC1.1, UC1.2, UC1.3

3.2 Module: RET – Data Retrieval

3.2.1 Architecture Diagram



3.2.2 Sample Interface



3.2.3 Primary Feature Progress

Functionality	0	1	2	3	4	5	6	7	8	9	-	M18	M24	M30	M36
LLM-aided function calling	100	100	100	100	100	100	100	100	100	100	-	100	100	100	100
Integration testing	90	90	90	90	90	90	90	100	100	100	-	100	100	100	100
Other use cases	75	75	75	75	75	75	75	80	90	90	-	90	100	100	100
Revisions	0	0	0	0	0	0	30	90	95	100	-	100	100	100	100
DS2 final integration	0	0	0	0	0	0	0	0	0	0	-	0	0	100	100
Transform redesign	0	0	0	0	0	20	25	40	50	50	-	50	80	100	100

(Note that work on this module began before the official start of Sprint 0)

3.2.4 Activity

Data offered by a data provider may be either a fix set of data, or more typically for large sets of data, a slice extracted from a larger set of data, selectable through parameters such as a time window or the ID of a specific entity. While in the former case the creation of the REST call to obtain the data is straightforward, the latter case can be challenging, especially if special formatting of the request parameters – such as creating timestamp in a given RFC format – is needed. Additionally, the data in the data offering may not conform to the requirements of the data requester (format, data field names, units etc.). Both of these cases can benefit from an innovative solution to allow a data consumer to obtain and use data, which otherwise in the worst case, will prevent non-technical users from taking advantage of the data space concept.

The Data Retrieval module (RET) addresses both of these issues through the use of a machine learning Large Language Model (LLM).

Functionality in RET was developed which allows the user to obtain a data offering based on its OpenAPI specification, guiding the user through the defining the required input parameters, and then using LLM function-calling technology to correctly compose the required REST call. This module then added additional functionality to execute the generated REST call.

RET was tested against data endpoints in both the Green Deal and City Scape user cases. As part of the testing, it was discovered that due to a server implementation error in one of the Green Deal end points, what technically was a correctly composed query was failing. RET was then expanded to handle the regeneration and re-execution of failed REST calls which proved to be successful.

An application-based GUI was developed for RET. As part of the GUI, the user can select a destination for the downloaded data which is currently a local file, although the code can be easily modified to include cloud stores.

The RET module was then further expanded to enable on-the-fly transformations of the download data, mapping between the data source schema to a desired local schema, based on a set of predefined library transformations.

The technology to have the LLM automatically select the required transformation routine required for the schema translation was investigated and prototyped. To test this, we developed a simple application which required transformation of some of the air pollution data in the Green Deal use case in order to graph it.

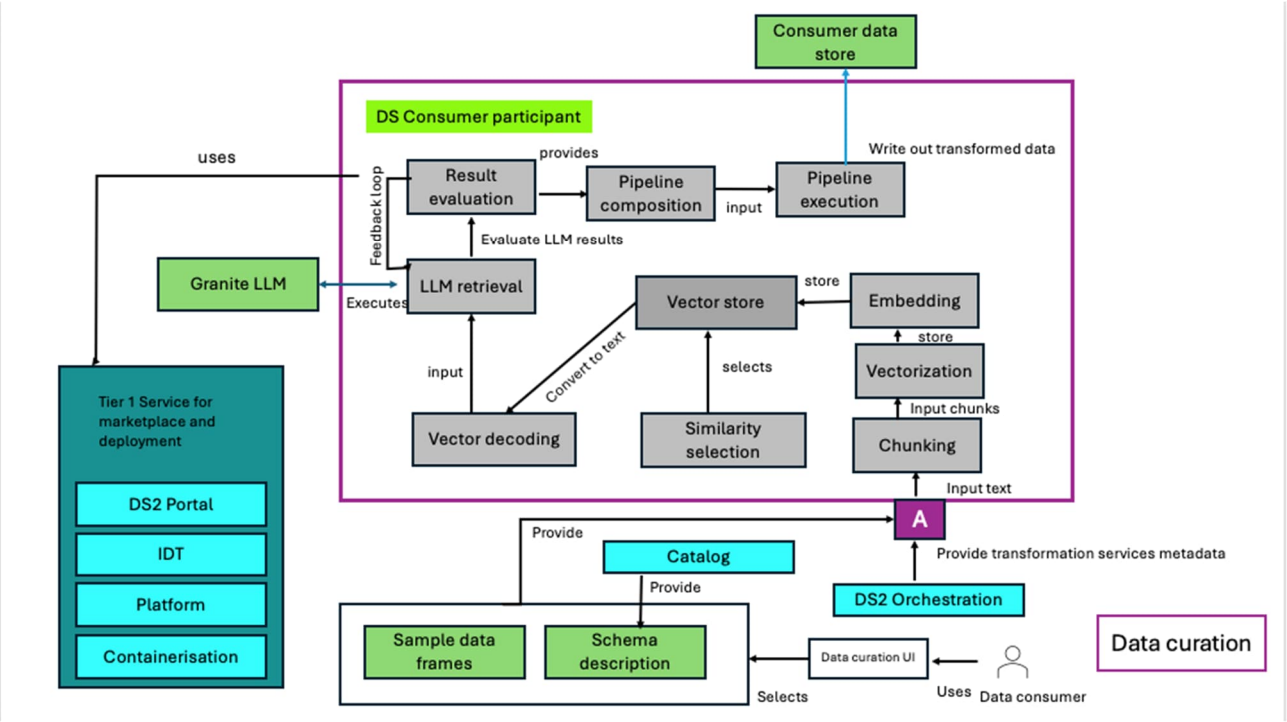
In the end the LLM-based solution was not included in the current code base as it was overkill for the current use case needs. Instead, a configuration file based solution was subsequently developed and tested.

3.2.5 Use Case Validation Plan

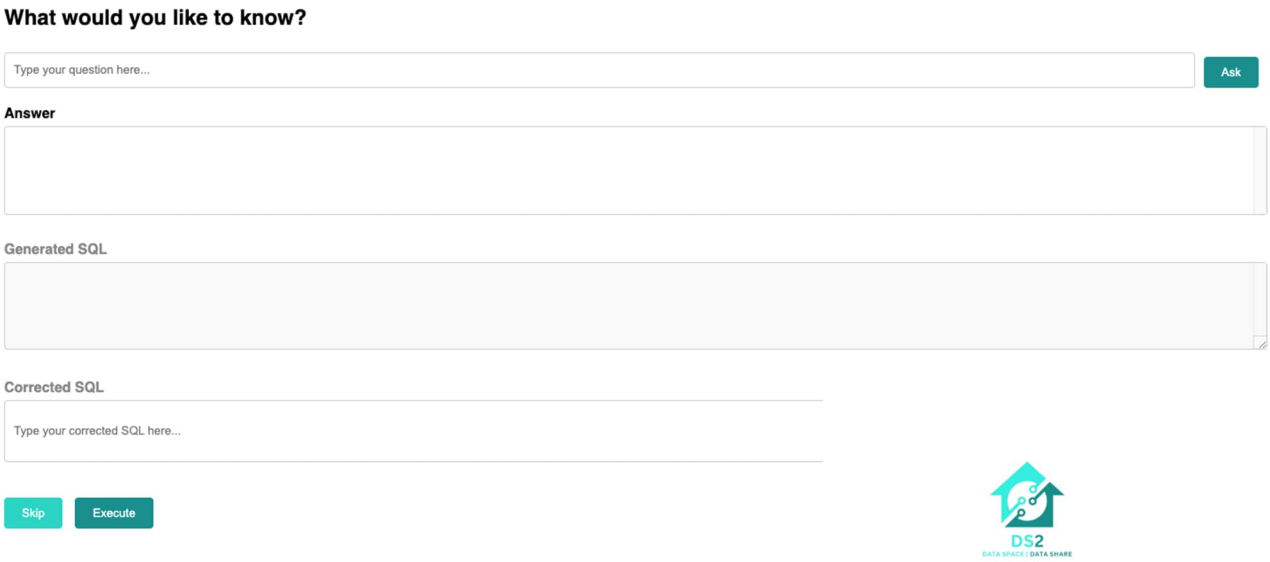
	KEY:	**Default install, UC = Validation**	**Specific To Case UC = Validation**
Module ID	**Precision Agriculture**	**Green Deal**	**City Scape**
RET	UC3.1	UC2.2	UC1.1, UC1.3, UC1.4

3.3 Module: CUR – Data Curation

3.3.1 Architecture Drawing



3.3.2 Sample Interface



3.3.3 Primary Feature Progress

Functionality	0	1	2	3	4	5	6	7	8	9	-	M18	M24	M30	M36
Data Mapping	0	90	90	100	100	100	100	100	100	100	-	100	100	100	100
Federation of data sources	50	60	70	90	90	95	100	100	100	100	-	100	100	100	100
On-the-fly creation of a database	90	90	95	100	100	100	100	100	100	100	-	100	100	100	100
NL2SQL	80	90	90	90	95	95	95	100	100	100	-	100	100	100	100
NL2SQL using Slovenian data	80	85	90	90	95	95	95	95	95	95	-	95	100	100	100
NL2SQL using Romanian data	80	85	90	90	95	95	95	95	0	0	-	100	100	100	100
NL2SQL using Greek data	0	0	0	0	10	0	0	0	0	0	-	100	100	100	100

(Note that work on this module began before the official start of Sprint 0)

3.3.4 Activity

While the goal of the Data Retrieval module is to aid in obtaining data offerings, the data, once obtained, essentially exists in what can be considered as local silos. The goal of the Data Curation module (CUR) is to make this data appear as federated to a user, providing behind the scenes the data curation required to enable this.

As part of this solution, a solution for transforming structured data into SQL tables was developed. An agentic workflow was developed which, at a high level, takes natural language queries, transforms them into SQL queries, executes the query, and then reformats the answer to a user-friendly form. As part of this flow, a stage was developed to extract schema information which is fed into the LLM to direct the composition of the query. Additionally, this stage also developed the data curation rules for the LLM, which include data casting rules, timestamp formatting, the boundaries required for geo-spatial windows when encountering longitude and latitude values, and other directions to enable successfully JOINs on disparate SQL tables.

The CUR module was tested against actual use case data for both the Green Deal and City Scape use cases, evaluating different machine learning models. Experimentation on this data showed that while the SQL generation was typically very good, there were still "hard" queries that could not be resolved by AI, and so a stage for human intervention was introduced to the agentic flow. The development of a vector store (https://milvus.io/) based RAG solution was then created in order to allow the LLM to learn from the human intervention in future NL2SQL calls, and the agentic workflow was further expanded to accommodate this.

A set of benchmarking queries for both the Green Deal and City Scape use cases were developed and evaluated against the CUR solution. Additionally, a version of CUR was created which works against the well-known BIRD-SQL benchmark (https://bird-bench.github.io/) and extensive testing of CUR against the benchmark was carried out, once again, evaluating the effectiveness of different underlaying LLMs.

One result of the benchmarking was the realization that vector store retrieval based on standard semantic similarity to previously asked questions was not always effective. Therefore, a hybrid retrieval strategy was developed, where both semantic and keyword search of the vector store were used. The agentic workflow was expanded to include a stage for re-ranking of the combined retrieval results, which was then used to provide multi-shot examples to the LLM for NL2SQL.

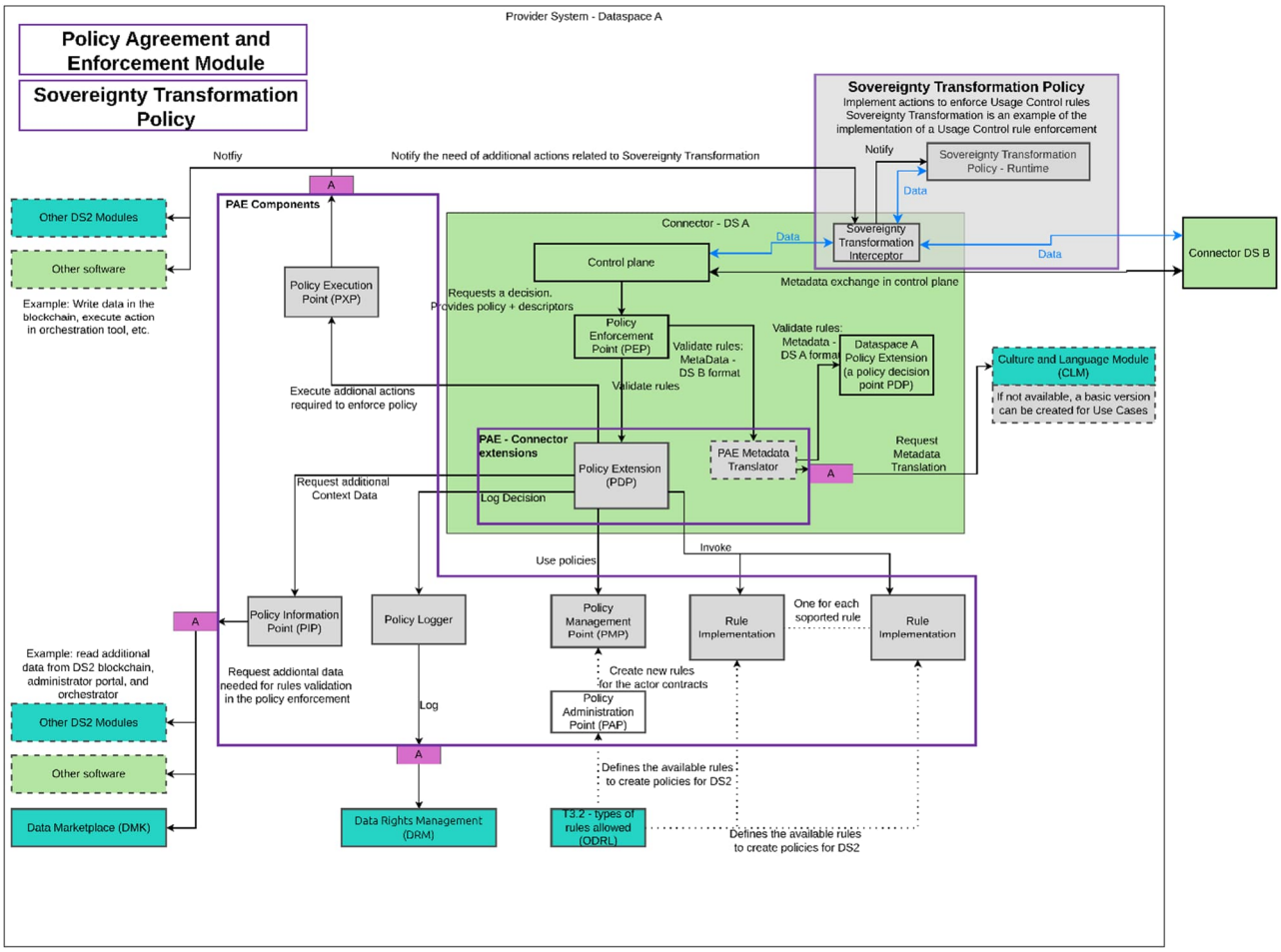
Finally, a web-based GUI was developed for CUR which allows for users to ask questions in a natural language, and to intervene with manual SQL creation when the agentic workflow flags this as necessary.

3.3.5 Use Case Validation Plan

	KEY:	Default install, UC = Validation	Specific To Case UC = Validation
Module ID	**Precision Agriculture**	**Green Deal**	**City Scape**
CUR	UC3.1	UC2.1	

3.4 Module: PAE – Policy Agreement and Enforcement

3.4.1 Architecture Diagram



3.4.2 Sample Interface

This is a backend module deployed as EDC Connector Extensions. It does not have a UI.

3.4.3 Primary Feature Progress

Functionality	0	1	2	3	4	5	6	7	8	9	- -	M18	M24	M30	M36
Policy Extension (PDP)	50	55	65	65	65	75	85	95	95	95	- -	95	100	100	100
Metadata Translator	0	0	0	0	0	0	0	0	0	0	- -	0	0	100	100
Sovereignty Transformation Interceptor	0	0	0	0	0	15	30	40	40	50	- -	50	100	100	100
Policy Management Point (PMP)	10	10	10	10	10	10	10	10	10	10	- -	10	10	100	100
Policy Administration Point (PAP)	10	10	10	10	10	10	10	10	10	10	- -	10	10	100	100
Policy Logger	0	0	0	0	0	0	0	50	75	75	- -	75	100	100	100
Policy Information Point (PIP)	0	0	0	0	0	10	20	70	70	100	- -	100	100	100	100
Policy Execution Point (PXP)	0	0	0	0	0	20	40	50	50	50	- -	50	100	100	100
Rule Implementation	20	30	40	40	40	40	40	50	50	50	- -	50	75	100	100
Sovereignty Transformation IDE UI	0	0	0	0	0	0	0	0	0	0	- -	0	100	100	100
Sovereignty Transformation IDE Controller	0	0	0	0	0	0	0	0	0	0	- -	0	100	100	100
Testing Framework	0	0	0	0	0	0	0	0	0	0	- -	0	100	100	100
Job Definition Storage	0	0	0	0	0	0	0	0	0	0	- -	0	100	100	100
Runtime Controller	0	0	0	0	0	0	0	0	0	0	- -	0	100	100	100
Sovereignty Transformation Job	0	0	0	0	0	0	0	0	0	0	- -	0	100	100	100

Job Logger	0	0	0	0	0	0	0	0	0	0	- -	0	0	100	100
Job State Storage	0	0	0	0	0	0	0	0	0	0	- -	0	0	100	100
Data Right Management (DRM)	0	0	0	0	0	0	0	0	0	0	- -	0	0	100	100
Data Marketplace	0	0	0	0	0	0	0	0	0	0	- -	0	0	100	100
Culture and Language Module	0	0	0	0	0	0	0	0	0	0	- -	0	0	100	100

3.4.4 Activity

The development of the Policy Enforcement (PAE) module has progressed as expected. The objective of this module is to provide to the DS2 Connector with the capability to enforce the necessary policies governing DS2 data transactions. Since the DS2 Connector is based on the EDC Connector, the PAE module is being implemented as a set of EDC Connector extensions.

Throughout the course of the project, the EDC Connector's support for policy enforcement has evolved. These changes introduced some challenges during development, but on the other hand, also brought improvements to the connector's native policy-handling capabilities.

Regarding the types of policies that can be enforced, the PAE module supports policies that rely on attributes defined within the DS2 namespace. In this context, attributes are considered metadata related to users or datasets, which are used in the evaluation logic of the policies. For example, a rule might allow access to a dataset only for users located in a specific geographic region.

To enable advanced policy enforcement, several key architectural components are being developed: the Policy Information Point (PIP), the Policy Execution Point (PXP), and the Sovereignty Transformation Interceptor. The PIP is designed to retrieve external attributes and contextual data needed for policy evaluation, for instance, metadata about participants obtained from the DS2 Portal. Whilst the EDC Connector already provides context information from its identity manager and internal configuration, the PIP serves as an additional source to enrich that context. The PXP and the Sovereignty Transformation Interceptor enable policy-driven data transformations, especially in scenarios involving data sovereignty, where actions, such as anonymization, must be applied based on policy rules.

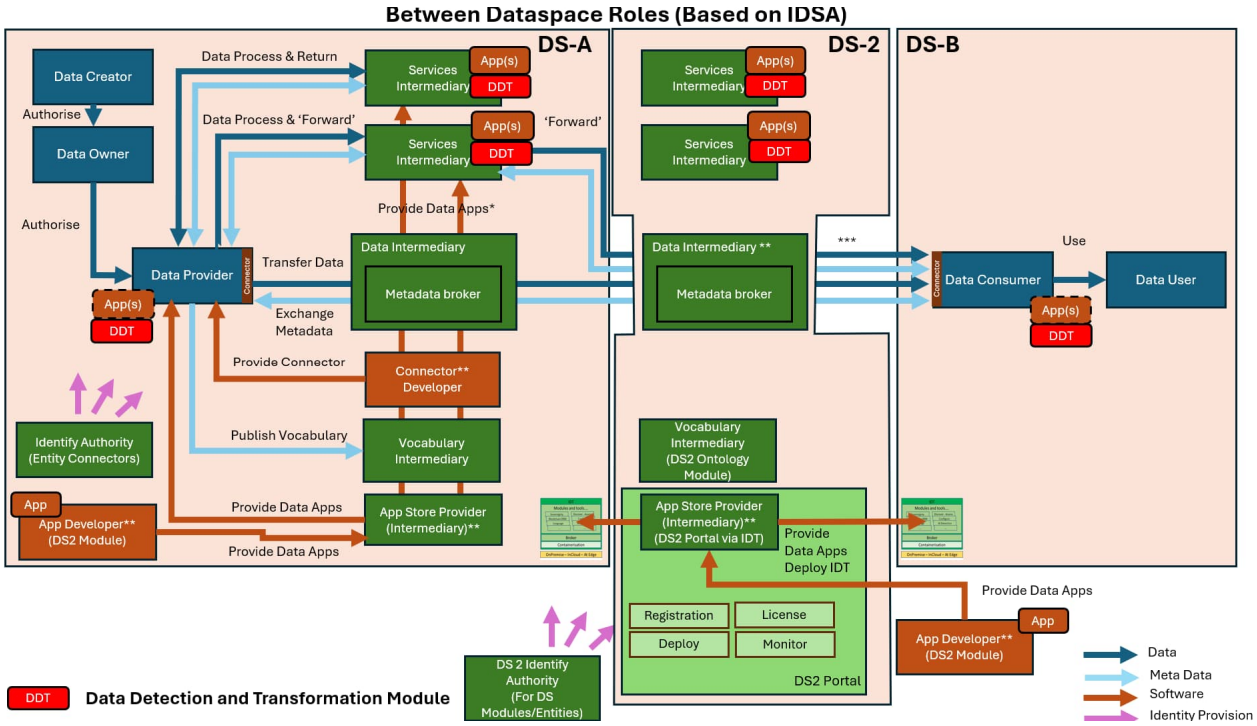
The Sovereignty Transformation Interceptor is based on the integration between the PAE and DINS modules. That integration is been done by creating EDC Connection extensions that allows to DINS be executed in the data transaction within the Data Plane of the EDC Connector. This connection enables dynamic, rule-based transformations and fosters real-time interaction between enforcement and transformation components.

3.4.5 Use Case Validation Plan

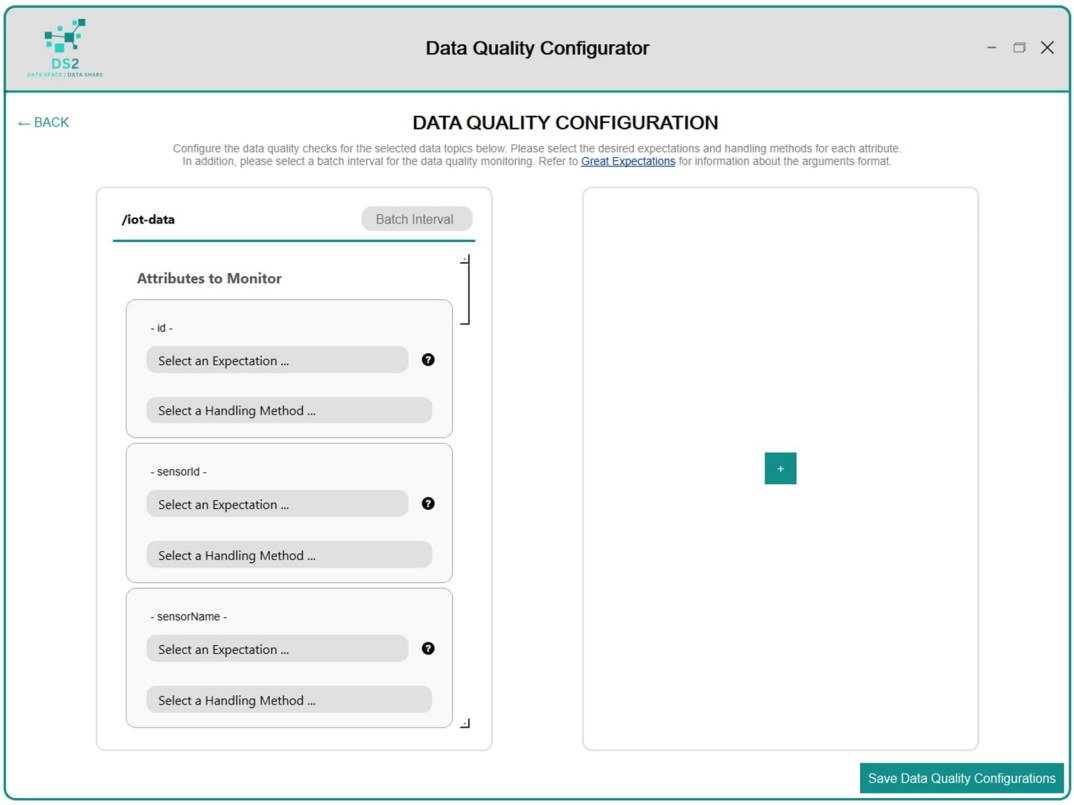
	KEY:	**Default install, UC = Validation**	**Specific To Case UC = Validation**
Module ID	**Precision Agriculture**	**Green Deal**	**City Scape**
PAE	UC3.1, UC3.2	UC2.1	UC1.1, UC1.4

3.5 Module: DDT – Data Detection and Transformation

3.5.1 Architecture Drawing



3.5.2 Sample Interface



3.5.3 Primary Feature Progress

Functionality	0	1	2	3	4	5	6	7	8	9	-	M18	M24	M30	M36
Batch Processing	100	100	100	100	100	100	100					100	100	100	100
Data Validation w. great-expectations	60	100	100	100	100	100	100					100	100	100	100
Config Interpreter	70	75	75	75	75	75	75					75	75	100	100
Containerization	30	30	40	40	40	60	60					60	90	100	100
Data Exception Handler	40	40	40	40	40	50	60					60	80	100	100
Configuration UI	40	50	60	60	60	70	70					70	90	100	100
Expectation rule customizer	20	20	20	20	20	20	20					20	50	100	100
Data Syntax Check	-	-	-	-	20	20	20					20	75	100	100

3.5.4 Activity

DDT integrates into the data pipeline where data is collected from edge devices and forwarded to cloud platforms. To ensure that this data is trustworthy and shareable across different dataspaces, it is critical to verify its quality. The module subscribes to incoming MQTT topics and evaluates each message against predefined data quality rules.

At the beginning of development, key functionalities were defined based on use case requirements and the general DDT concept. Batch processing as a core functionality was addressed and implemented first. Here, data within messages are collected and buffered until a configured batch size is reached. Once the required batch is complete, it is passed on for validation.

In the next step, functionalities for carrying out data validation were designed and implemented. Here, the team explored options for performing rule-based validation, and as a result, the Great Expectations library - a leading open-source tool for automated data validation, documentation, and monitoring - was selected and integrated in DDT. In addition to predefined quality rules, the team started working on an automated mechanism to infer attribute types - distinguishing identifiers, measurements, and context data. This allows for a more context-aware syntax validation compared to the functionalities provided by the Great Expectations library.

In the case of exceptions in the processing and validation of data, support for data exception handling was implemented. The module can be configured to:

- Apply a correction strategy, such as inputting missing values or smoothing outliers.
- Raise an alarm, which is then forwarded to the IoT platform.
- Each expectation rule can be individually configured to select the appropriate handler.

All DDT mechanisms are configured via a user interface. It was built according to the DS2 style guide provided to the consortium. It allows a flexible definition of operations on data, and to better support user of E2C, an E2C config interpreter was built by the team. It allows pre-configuring of the DDT by suggesting E2C endpoints and hence allows for a seamless integration between the configuration UI, DDT backend and E2C streaming services.

Concerning the intended deployment of the final module, containerization aspects have been considered from the very beginning. Currently, the DDT backend can already be deployed as a Docker container, however the UI component is still pending containerization.

Recently, the module was extended to send both raw and corrected data to an open-source IoT cloud platform (ThingsBoard) to visualize the results of data quality checks and corrections.

The main activities carried out by DIGI focused on researching and evaluating advanced data quality and outlier detection tools to enhance the DDT module's capabilities. Initial efforts involved exploring Great Expectations, an open-source data quality platform widely used by data engineers for validating data correctness, preventing low-quality data from propagating downstream, and enabling collaboration with non-technical stakeholders. Integration aspects were considered, highlighting Great Expectations' compatibility with various databases, data storage systems, and processing frameworks such as Apache Spark, Apache Airflow. Parallel to this, exploratory work was conducted on automated exploratory data analysis tools like AutoViz, which facilitates rapid visualization and understanding of datasets with minimal code. This supports early-stage data profiling and quality assessment within the DDT module. A significant focus was placed on outlier detection workflows, which are critical for identifying anomalous data points that could affect system reliability and security. The activities included defining the types of outliers to detect, selecting appropriate models, data collection and cleaning, feature engineering, model fitting, evaluation, and deployment for ongoing monitoring. Both univariate and multivariate outlier detection methods were studied, including classical algorithms such as k-Nearest Neighbors, Local Outlier Factor, DBSCAN clustering, and more advanced frameworks like PyOD and TODS for time-series anomaly detection. DIGI activities also covered graph-based outlier detection using

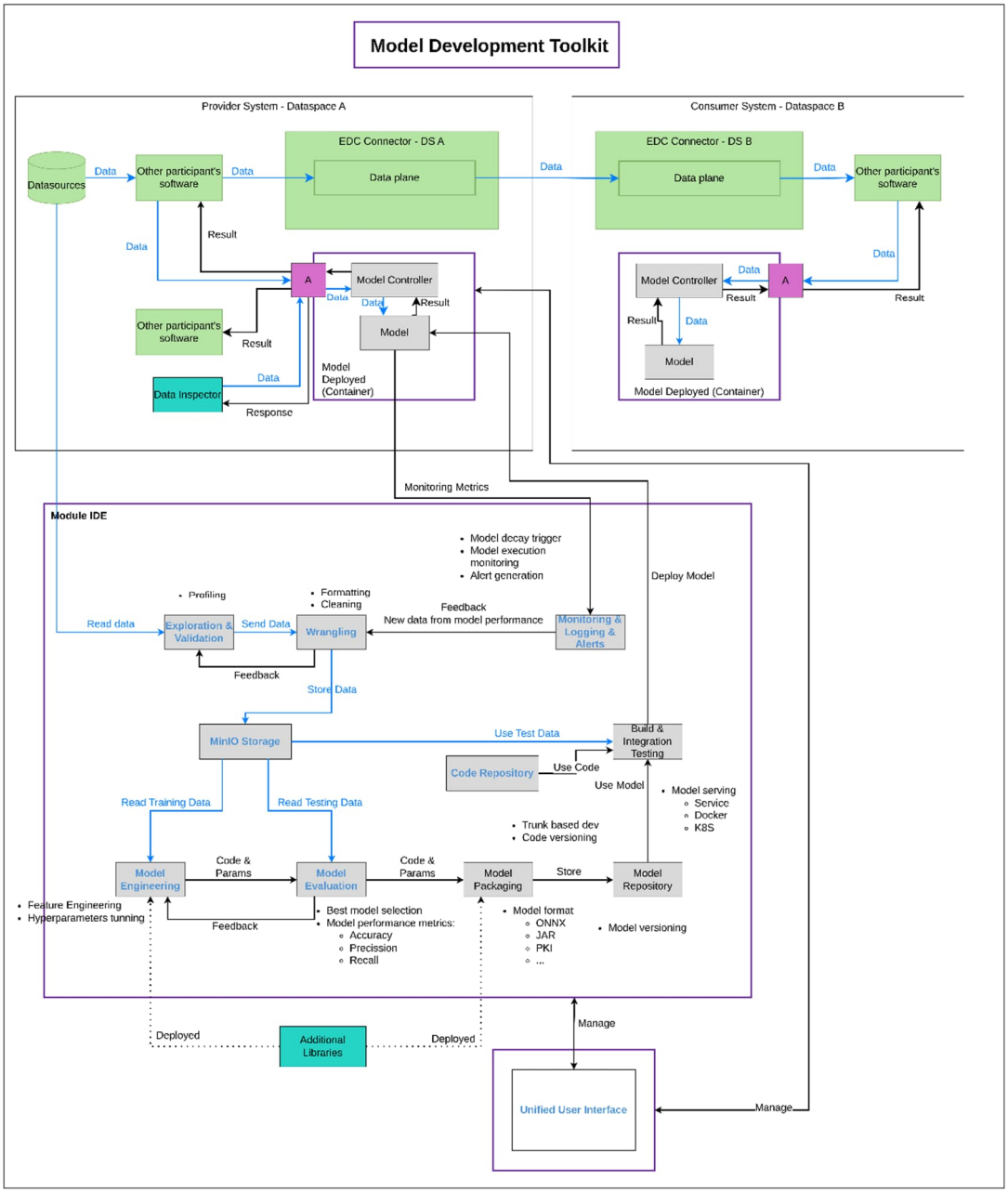
PyGOD, which leverages graph neural networks to identify anomalies in relational data structures, expanding the scope of anomaly detection beyond tabular data. Future development will focus on implementing these tools and algorithms to ensure robust data validation, continuous monitoring, and timely detection of security-relevant anomalies across the platform.

3.5.5 Use Case Validation Plan

	KEY:	**Default install, UC = Validation**	**Specific To Case UC = Validation**
Module ID	**Precision Agriculture**	**Green Deal**	**City Scape**
DDT	UC3.1, UC3.2	UC2.1	

3.6 Module: MDT- Model Development Toolkit

3.6.1 Architecture Diagram



3.6.2 Sample Interface

3.6.3 Primary Feature Progress

Component	0	1	2	3	4	5	6	7	8	9	- -	M18	M24	M30	M36
Exploration & Validation	10	10	10	10	10	10	10	100	100	100	- -	100	100	100	100
Wrangling	10	10	10	10	10	10	30	100	100	100	- -	100	100	100	100
MinIO	10	10	40	40	80	100	100	100	100	100	- -	100	100	100	100
Model Engineering	10	10	10	10	10	10	10	50	80	100	- -	100	100	100	100
Model Evaluation	10	10	10	20	30	30	30	70	90	100	- -	100	100	100	100
Model Packaging	0	0	15	15	15	15	15	30	20	30	- -	30	100	100	100
Model Repository	0	0	0	0	0	0	0	30	30	30	- -	30	100	100	100
Code Repository	0	0	25	25	25	25	25	70	100	100	- -	100	100	100	100
Build & Integration Testing	0	0	15	15	15	15	15	15	15	15	- -	15	100	100	100
Monitoring, Logging, & Alerts	0	0	0	0	0	0	0	0	0	0	- -	0	0	100	100
Model Controller	0	0	25	25	25	25	25	25	25	25	- -	25	100	100	100
Unified User Interface	0	0	0	0	0	0	0	0	0	25	- -	25	100	100	100
Data Inspector Integration	0	0	0	0	0	0	0	0	0	0	- -	0	0	100	100
Additional Libraries	0	0	0	0	0	0	0	0	0	0	- -	0	0	100	100
Bug Fixing and Maintenance	0	0	0	0	0	0	0	0	0	0	- -	0	0	0	100

3.6.4 Activity

The goal of the Model Development Toolkit (MDT) is to serve as a set of tools that facilitate the development and deployment of machine learning models within the DS2 ecosystem, for example, to enable analysis of data during data transactions.

To initiate the module's development, an internal example using synthetic data was created as a test base. This example supported the evaluation of changes across all components of the module and served as a demonstrator to showcase its capabilities to the owners of the use cases.

Guided by this example, the initial objective was to establish a minimum implementation for each core component. As outlined in the architecture document, several of these components are based on previous developments. Therefore, development efforts were concentrated on those components that lacked support or had only partial support in the existing background.

Upgrades were implemented in the MinIO object storage interface, significantly improving the user experience through a more intuitive UI that mirrors traditional file system interactions. Each user can now manage their own data within MinIO independently, enhancing usability and strengthening data governance.

Capabilities for packaging and deploying machine learning models were developed. A dedicated microservice template was created to support the deployment of trained models. This template is now available in the project's Git repository. The microservice is built using Jenkins pipelines and exposes a REST API described with an OpenAPI v3 specification that facilitate the integration of model artifacts into data transaction workflows.

Another important achievement was the integration and upgrade of MLflow, the open-source platform for managing the ML lifecycle. A newer version of MLflow was incorporated which required corresponding adaptations in the user interface. The integration is still ongoing and will support several components of MDT, particularly those related to Model Evaluation, Model Packaging, and the Model Repository.

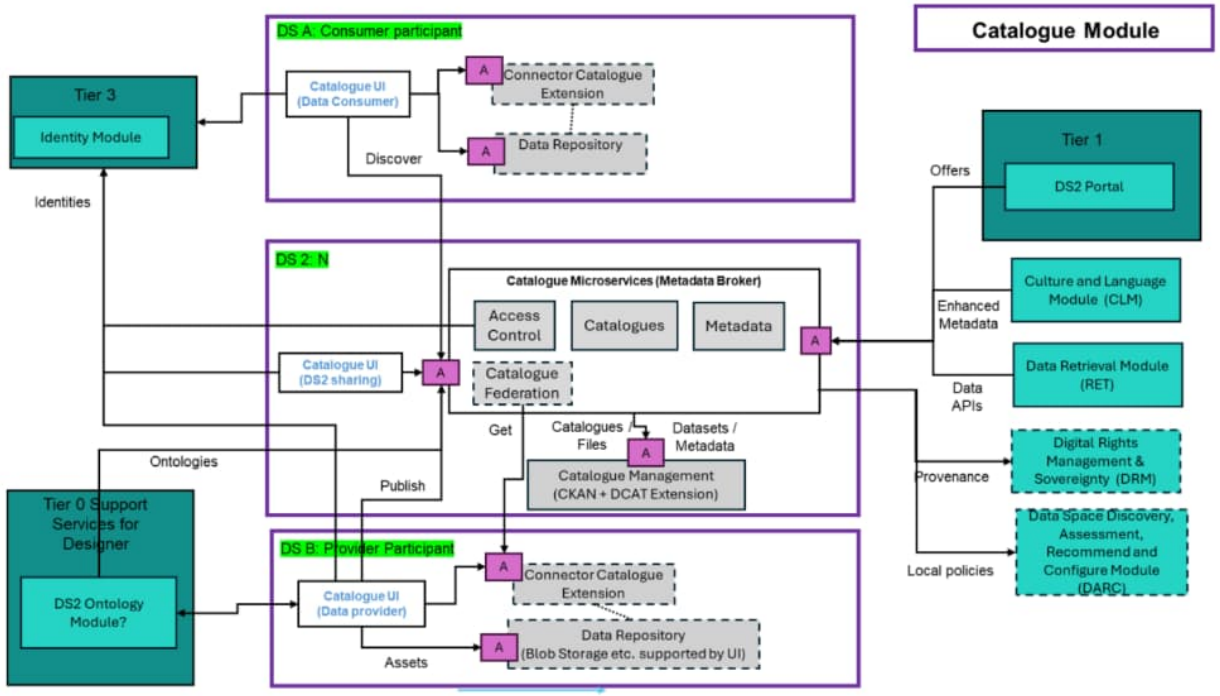
After the previous development, the effort focused on updating the Wrangling and Exploration and Validation components. In these cases, only version upgrades were necessary, as the components leverage previous background implementations.

3.6.5 Use Case Validation Plan

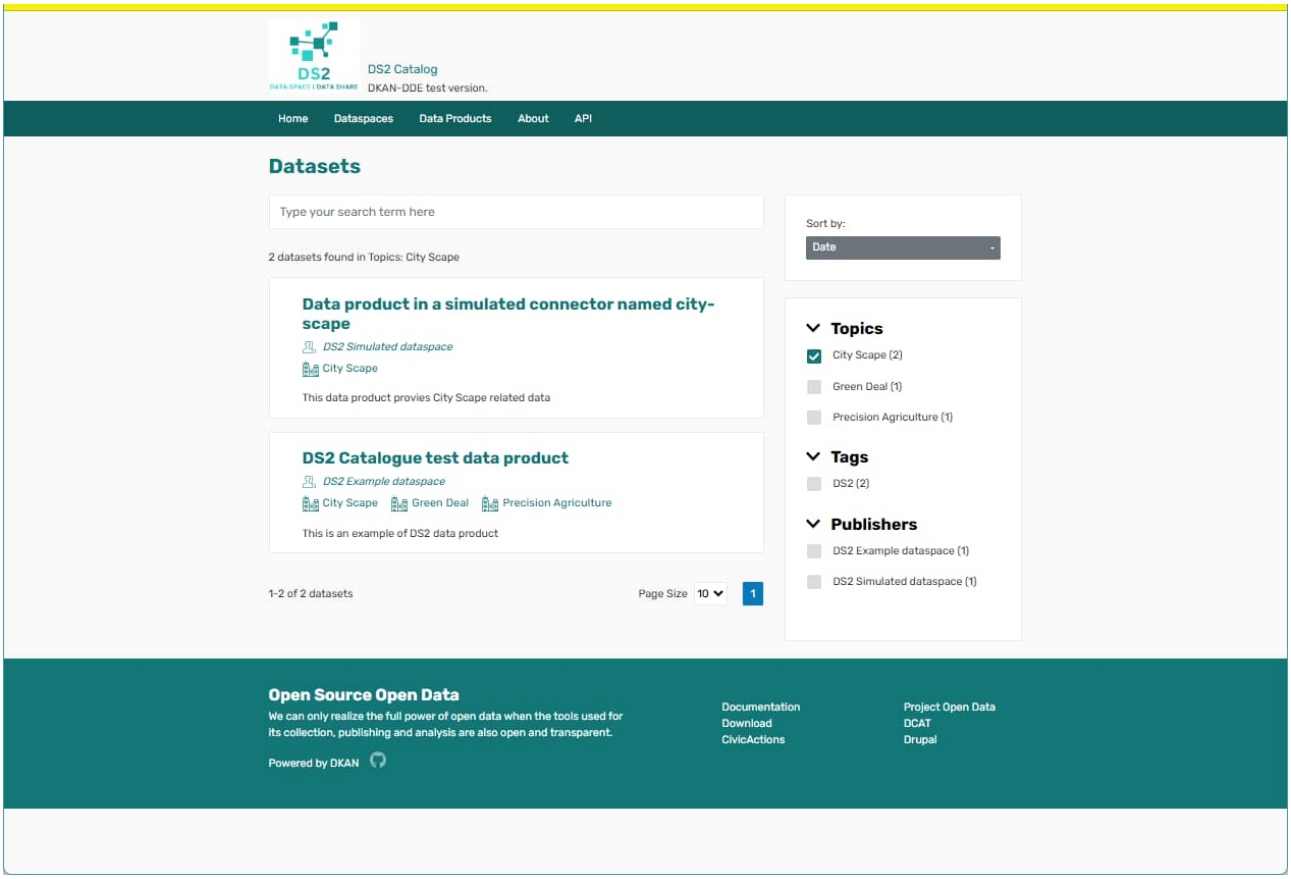
	KEY:	**Default install, UC = Validation**	**Specific To Case UC = Validation**
Module ID	**Precision Agriculture**	**Green Deal**	**City Scape**
MDT	UC3.1, UC3.2		

3.7 Module: CAT - Catalogue

3.7.1 Architecture Diagram



3.7.2 Sample Interface



3.7.3 Primary Feature Progress

Functionality	0	1	2	3	4	5	6	7	8	9	--	M18	M24	M30	M36
Catalogue Server	50	60	60	65	65	65	65	65	65	65	--	65	90	100	100
Catalogue server extensions										10	--	0	50	100	100
Microservices framework	15	40	60	70	75	75	75	75	75	75	--	75	100	100	100
Catalogue Protocol API			15	25	30	30	30	30	30	35	--	30	100	100	100
Extended Metadata API	15	30	30	30	30	30	30	30	35	40	--	30	80	100	100
DS2 Identity			5	5	5	5	5	10	15	20	--	5	50	100	100
Catalogue federation	20	20	20	25	30	35	35	35	35	35	--	35	80	100	100
Connector and extensions	50	50	50	50	55	60	60	60	60	60	--	60	90	100	100
UI (Sharing)	10	20	20	20	20	20	20	25	30	30	--	20	80	100	100
UI (Provider)	15	25	25	25	25	20	25	30	30	30	--	25	80	100	100
UI (Consumer)	10	20	20	20	20	20	20	20	20	20	--	20	50	100	100
Simulation						10	15	20	25	30	--	15	80	100	100

3.7.4 Activity

The goal of the CAT module is to support creating data assets (Data Products) and to provide publication and search interfaces with associated metadata, and data model schemas to support validation. The initial plan for implementation was to use the open-source catalogue platform, CKAN, to provide a UI with a technical API to integrate with rest of DS2 modules. During implementation activity it became evident that support for catalogues based on DCAT vocabulary was better supported in the DKAN platform, so this was adopted instead. The DKAN platform supports the CKAN technical API and also provides a better means to tailor the UI for DS2 project requirements. The UI and DCAT schema of DKAN were modified for better compatibility with catalogues provided by EDC connectors.

The API of CAT module will be implemented on top of the functionalities provided by DKAN platform. Initial selection of Fastapi.js and Moleculer.js libraries for Javascript based microservices framework and initial

versions of identity and catalogue access API functionality were implemented. The DKAN Dataset.json schema was modified to support DCAT version used by EDC connector and IDSA Catalogue protocol. Support for tailoring DKAN for project needs was found to be good using its external UI module. Installation is not straightforward as DKAN is built on top of DRUPAL platform. DS2 requires the CAT API to provide extended metadata to describe the data products. As the level of detail and implementation of real-world catalogues provided by connectors vary, the API must check all the data included into DS2 catalogue.

To test the API, a dataspace simulation was provided. This allows to run several EDC connector instances and define one or more product offerings they provide. The CAT module then reads the catalogue of connectors, checks the data and writes it to the DKAN platform so the products can be accessed in its UI. The handling of several dataspaces has also been implemented in the UI.

The work continues improving the API functionality for adding and querying extended metadata and integrating with DS2 identity module. During project, it has been evident that support for associated metadata in existing user interfaces of connector interfaces is lacking in user friendliness so the requirement of the CAT module to provide better UI support, especially for the data product provider, is considered important.

3.7.5 Use Case Validation Plan

	KEY:	**Default install, UC = Validation**	**Specific To Case UC = Validation**
Module ID	**Precision Agriculture**	**Green Deal**	**City Scape**
CAT	There will be one central server for all use cases		

4 ADDITIONAL ACTIVITIES

None

5 KPI, RISKS, AND PRIMARY ISSUES

KPI Status:

KPI ID	Description	Status
KPI 4.1	Demonstration in the use cases that vast volumes of data can be optimally transferred sourced cached from one or more different data stores/sectors and seamlessly incorporated into a workflow along with locally stored data.	This effort has been transferred to WP6.

KPI 4.2	Benchmarking of amount of transferred data from the onset of the anomaly and overall time of detection as compared to the existing technologies in place with the use case owners will lead to at least a 50% reduction in detection time.	Scheduled for the second half of the project.
KPI 4.3	Benchmarking of the time required to make data available using DS2 technologies vs. the time using existing technologies in place with the use case owners will lead to at least a 50% improvement in productivity.	The tools to enable this technology have been developed in T4.1 (RET) and T4.3 (CUR) and are awaiting trails with the use case partners.

Primary Risks:

- Maturity of EDC Connector software for policy enforcement
- Data type and quality inappropriate for the Data Detection and Transformation module.

Primary Issues:

- None

6 CONCLUSION

This deliverable presents an overview of the work performed regarding the modules in WP4 over the first 18 months of the project. The main source of information for further information on the modules can be found in the DS2 Github site, as described in Section 3.

In addition, a summary of the WP KPI's status and a table of Risks detected during the first phase of development have been reported.

Work on all modules is on track and in a number of cases, extra functionality is being added as opportunities for increased value are identified. No serious risks have been identified, and most modules already have working, baseline versions running which further servers to mitigate risk.

The Agile process has been adopted an in the software phase a two weekly sprint paradigm has been adopted with corresponding formal highlight updates and meetings to ensure progress.

The next steps will be to continue improving the modules, including integration with other modules, where required, as well as with the DS2 Connector. Testing of the modules with the use case partners is required for a verification of the technological results.

Software and documentation in the DS2 Github repository will continue to evolve.

In summary, WP4 is on track and expects to exceed its objectives.