9 Data Retriever Module (RET)

9.1 Data Retriever Module (RET)

Owner(s):	IBM
DOA Task:	T4.1
Tier:	2
Nature:	Optional
Result:	Outcome

DS2 Data Retrieval

This task will create a federation mechanism to enable different data spaces to interoperate. This task will orchestrate the lifecycle of data from data collection to data exchange, to data disposal/deletion across a federation of distributed data stores. The data lifecycle will include the establishment of a data contract between the data sources, the establishment of trust between entities in the data flow, and the adherence to data sovereignty and security requirements in the resulting federated data set. Other challenges will be addressed, such as accountability for use of purpose, the propagation of new domain-specific data restrictions (such as policy changes) across the federation, and methods for non-repudiable lineage across the lifecycle. Additionally, topics to reduce the latency involved in the transfer of huge amounts of data, such as caching, and data relocation will be investigated. This task will examine current, emerging technologies in this field, such as work being led by European Data Spaces and from the GAIA-X project as a basis for extension

9.1.1 Introduction

Purpose: 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 parameters for the connector based on a textual request in natural language.

Description: Work in this task will use machine learning inference, together with the published OpenAPI for the data source to allow a user to request data through a natural language query. (e.g. "Create a call to give me all monitoring events for process X which occurred on Jan 10, in 15 second intervals, between 5:00am – 6:30am from data source Y"). A link to the associated OpenAPI description will be stored in the Catalogue Meta Data Broker alongside the description of the data source.

The Data Retriever Module will translate the user request into the required REST call, based on the information in the OpenAPI description. A feedback mechanism within the Data Retriever Module verifies the results produced against the OpenAPI spec and re-executes REST generation if the verification fails. This is an important step to prevent machine learning hallucinations in the results.

The generated REST call will be used by the data consumer's connector to request data from the data producer. This functionality will require an extension to the Consumer Connector to support obtaining and passing REST query parameters to the data plane, and an extension to the Provider Connector to obtain the authentication token require for backend data access. Additionally, other extensions to the Connector environment will most likely be necessary.

9.1.2 Where this component fits

9.1.2.1 Big Picture

In the IDSA model, a data consumer requests from a data provider a data set, and the data provider then creates a data package which can be pull by the consumer. This model is well adapted for simple scenarios where catalogued data sets exist and are the data consumers are expected to manage all required data extraction on their side. Potentially this may mean that huge amounts of extraneous data may need to be downloaded with an associated cost, when only a slice of the data set is really required. Additionally, this method neglects the native querying capabilities that the provider's backend data server may offer and requires the data consumer to write code to mimic the original querying capabilities.

Instead, the Data Retriever module provides a federation mechanism to allow data providers and consumers in different data spaces to interoperate in the exchange of data in a much efficient and accessible manner. The module accomplishes this by allowing a data consumer to request subsets of data by taking full advantage of the REST interface parameters exposed by the data provider, and thus reducing data transfer costs and transfer times. Consequently, this eliminates the need for the data consumer to write filtering or selection code for what can be handled by selection of the REST parameters passed to the data provider's data source backend.

The Data Retriever module takes as input the OpenAPI specification for the remote data source server at the Data Provider and a free text description of what data is being requested. Using a machine learning (ML) Large Language Model (LLM) with Retrieval Augmented Generation (RAG), the Data Retriever formulates a REST call which gets passed to the data consumer's Connector for forwarding to the data provider's Connector. The data provider's Connector will then proxy this request to the backend data server for data retrieval.

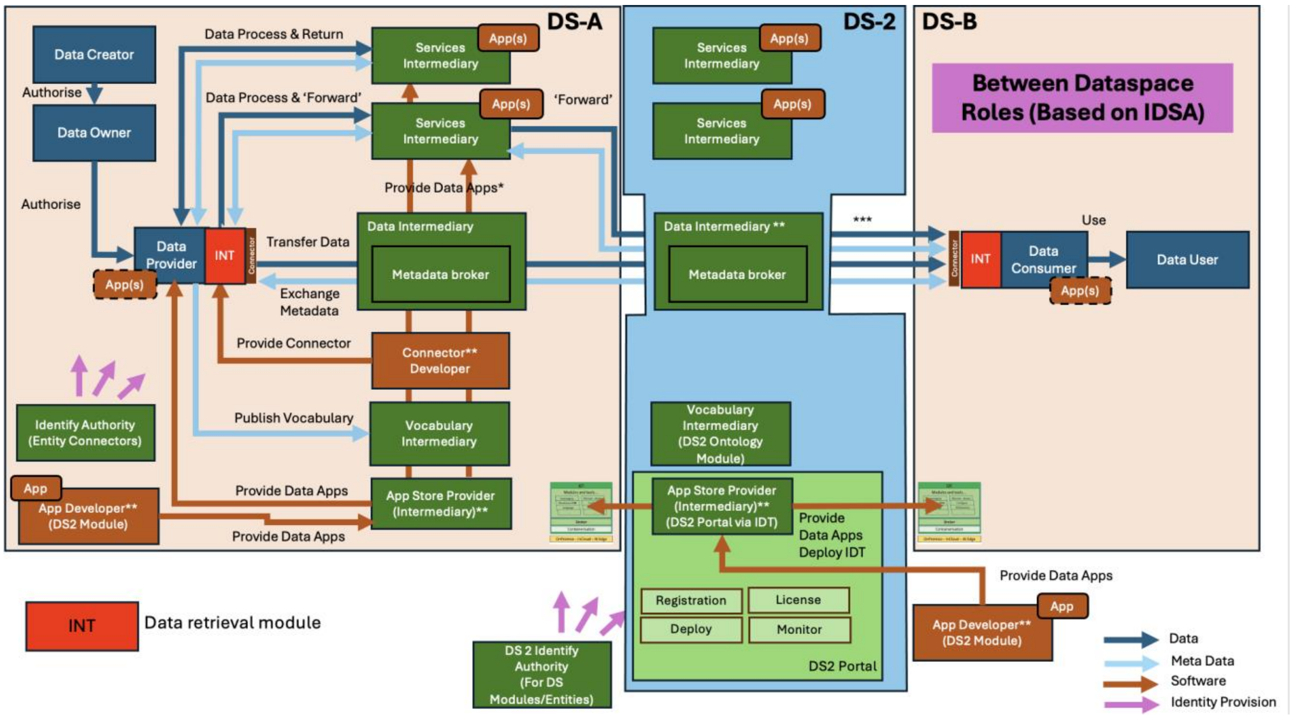
Additionally, the Data Retriever incorporates a feedback loop between result evaluation and LLM REST formulation to increase the accuracy of the generated result. It does this by refining the original query based on the evaluation and reevaluating against the LLM.

In a "standard" data space data request, the provider prepares in advance a data offering, which is static data set. To allow REST parameters to dynamically influence the data selection, extensions to the data consumer's and provider's connector will be provided to allow the connector to work in Data Plane Transfer mode.

There are a large number of technical challenges involved in this work such as:

- The composition of the correct REST call given a free text user request for data and a formal OpenAPI specification. This involves development of the Retrieval Augmented Generation (RAG) portion to improve results, the development of the verification of results through machine learning, the proper creation of the vector store, etc.

- Data consumers may be non-technical participants. Therefore, the Data Retriever will need to not only be resilient to poorly formulated data requests but will need to identify the source of incorrect results as stemming from incomplete or incorrect queries and guide the consumer in correcting them.
- There are a large number of technical issues involved in integrating with the Connector, since both the EDC and IDSA reference implementations only supply minimal implementations and very little documentation.



Where	Status
Within a single Dataspace for use between participants in that Dataspace only	Yes: Within a dataspace there are many data producers. The Data Retriever module will be able to create the REST command required by the data consumer's connector to access the data from the different providers.
Deployed and used by a single participant to enable the participant in either an In-Data space or Inter- Data space scenario	Yes: A single participant will use this module to generate the REST call required to specify and access a data package.
Across Dataspaces without Service Intermediary	Yes: This is essentially the same case generating the required REST call within a dataspace
Across Dataspace with Intermediary	No -

Other Comments	Current plans are only to support REST calls.

9.1.2.2 Within a single Dataspace (where applicable)

The data retriever component prepares a request for data based on the OpenAPI description of the requested data and a natural language description of what is requested and generates the required REST call parameters to extract the data. The REST call is subsequently executed by the Consumer's connector. The destination target of the REST call may be either within the same dataspace or to a different data space. The REST creation process is the same.

9.1.2.3 Deployed and used by a single participant (where applicable)

The generated REST call, as explained above, may include a target address which is within the participant's data space or to a data provider in a different data space.

9.1.2.4 Across Dataspaces without a Service Intermediary (where applicable)

This will be the same process as within a single dataspace.

9.1.2.5 Across Dataspaces with a Service Intermediary (where applicable)

N/A

9.1.3 Component Definition

The figure below represents the actors, internal structure, primary sub-components, primary DS2 module interfaces, and primary other interfaces of the module.

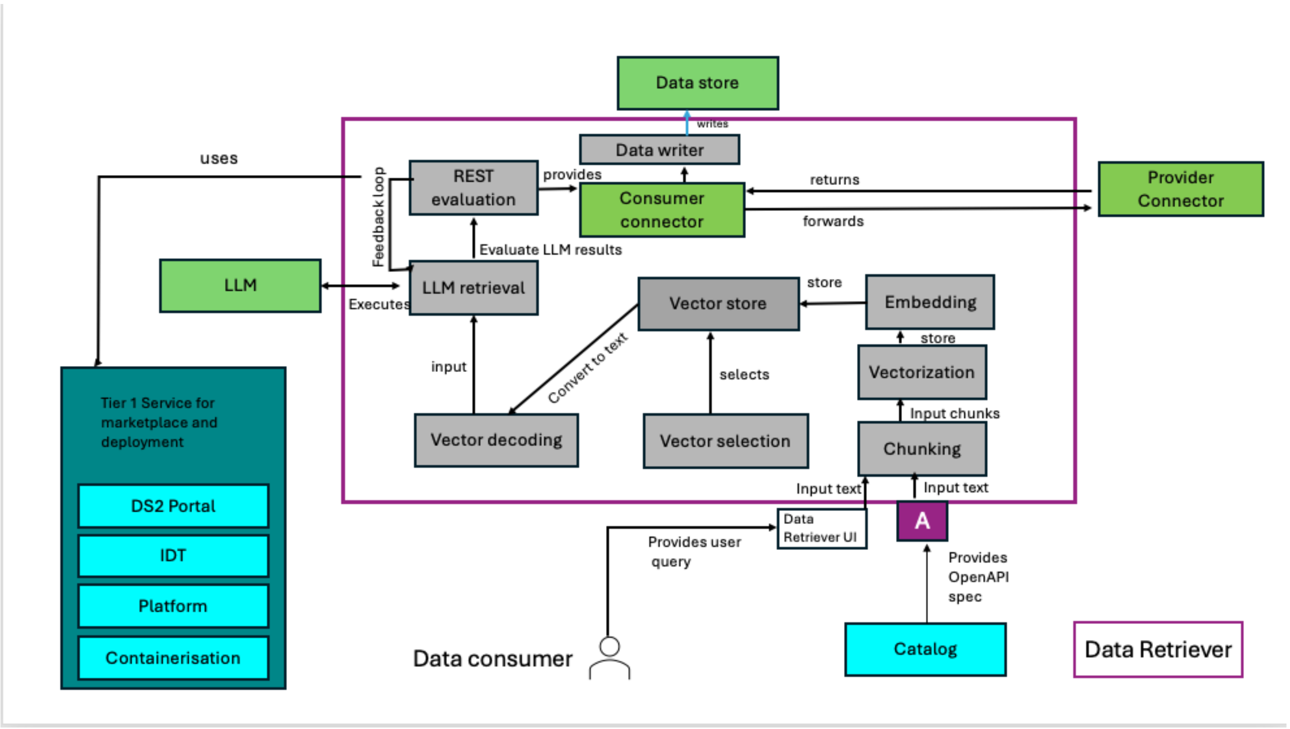


Figure 1: Schema for the Module

This module has the following subcomponent and other functions:

- **Vector store:** To implement RAG (Retrieval Augmented Generation) for the Machine Learning retrieval from the LLM, a vector store needs to be created for the OpenAPI specification. This requires careful selection from the specification and needs to be tuned around the way that specification is structured. It will receive input from the embedding component
- **Chunking:** Extracting relevant entries from the OpenAPI spec and arranging them in chunks.
- **Vectorization**: Encodes chunks for storage in the vector store
- **Embedding**: Enters the generated vector(s) into the vector store.
- **Vectorization Selection:** Determines the best match for the vector search based on tuned parameters.
- **Vector decoding:** Decodes a stored vector into text for input to the LLM
- **LLM retrieval:** Applies the user query to the LLM, supplemented with OpenAPI input
- **REST evaluation:** Compares the created REST query returned from the LLM to the original OpenAPI spec to validate results. There is a feedback loop that can reevaluate the query if the evaluation step fails which is also based on machine learning. If the evaluation determines that the query input by the user is either poorly formulated or missing parameters, feedback will be provided to the user.
- **Data Writer:** Extends the consumer connector to catch the returned reply from the producer connector and write the results to the local file system.

Interacting, external modules to the Data Retrieval Module include:

- **Connector:** Both Provider and Consumer connectors which will need to be extended to support the dynamic passing of REST query parameters, and the ability for the Provider Connector to access a backend data store with credentials.
- **Retrieval LLM:** An LLM (expected to be IBM Granite) will be required. The base knowledge in the trained model will be supplemented with the OpenAPI descriptions as part of the Retrieval Augmented Generation process.
- Catalog Module Metadata Broker subcomponent will supply OpenAPI descriptions for all data access queries.
- **Tier 1 Service Stack for Marketplace and deployment and API**: The full stack will be implemented as generically described elsewhere in this document.

9.1.4 Technical Foundations and Background

The Data Retriever component will use the Open-Source IBM Granite LLM to formulate the REST query required to access data sources, both within and external to data consumer's dataspace. As such, this component will play an essential role in federating data and dataspaces.

Additionally, the Connector will be extended to support Data Plan Transport will is required for dynamic compilation of data packages on the producer side.

Subcomponent/Component	Owner	License
IBM Granite LLM foundation model	IBM	Apache 2.0

9.1.5 Interaction of the Component

The following table specifies the primary input/output controls/data to blocks which are not part of the module

With Module/Feature	Receives From/Gives To	What
Catalog Metadata Broker Subcomponent	Receives From	Link to the OpenAPI description of the requested dataset
DS2 Connector	Receives From	Receives the formulated REST command from the Data Retriever and forwards to the producer connector.

9.1.6 Technical Risks

Risk	Description	Contingency Plan
Data not accessible through REST calls	The Data Retriever will only support the generation of REST calls to access remote data sources.	In the event that data is required from a source which does not expose a REST endpoint, a REST server will need to be installed by and at the data provider site which will include a method for retrieving the stored data. An OpenAPI description for this interface will be created.

9.1.7 Security

Security Issue	Description	Need
Secure communication for metadata	The Data Retriever will both receive and return metadata associated with what type of data is requested.	The request for a dataset may include metadata which itself is sensitive. Therefore, communication between data consumer and the Data Retriever and the Data Retriever and the DS2 Connector needs to be protected.

9.1.8 Data Governance

Data Governance Issue	Description	Need
Metadata Catalogue	This module will access metadata associated with a requested dataset from the Metadata Catalogue	Published Metadata Catalogue information will be freely available to all entities that are allowed to access the Metadata Catalogue.

9.1.9 Requirements and Functionality

This module will be used in the following use cases:

City Scape	✓
Green Deal	✓
Agriculture	✓
Inter-Sector	TBD

Their requirements and functions/extensions to achieve them relative to this module, specifically extracted from the use case are as per the table below noting that in many cases further discussion might need to take placed between pilot partner and module partner to determine if a fit or the scope of the precise fit.: :

WHERE	WHAT	WHY	Run/Design Time	Priority
	Use Case 1: City Scape			
Section 2.2 UC1.1	"Data exchange"	Provide the REST command to connect to data sources	R & D	M
Section 2.2 UC1.1	"Data exchange"	Provide the REST command to connect to data sources	R & D	M
Section 2.2 UC1.1	"Data exchange"	Provide the REST command to connect to data sources	R & D	M
Section 2.2 UC1.1	"Sharing and gathering data from multiple sources and sectors"	Provide the REST command to connect to data sources	R & D	M
	Use Case 2: Green Deal			
Section 2.2 UC2.1	"Relevant data sources to be obtained from both data spaces within the use case. "	Provide the REST command to connect to data sources	R & D	M
Section 2.2 UC2.2	"Relevant data sources to be obtained from both data	Provide the REST command to connect to data sources	R & D	M

	spaces within the use case. “			
Section 2.2 UC2.3	Dataset shared through the MOMS DATA SPACE	Provide the REST command to connect to data sources	R & D	M
Section 2.2 UC2.4	Dataset shared through the MOMS DATA SPACE	Provide the REST command to connect to data sources	R & D	M
Section 2.2 UC2.5	Dataset shared through the MOMS DATA SPACE	Provide the REST command to connect to data sources	R & D	M
	Use Case 3: Agriculture			
Section 1.1.4	“Data Integration and Accessibility”	Provide the REST command to connect to data sources	R & D	M
Section 2.2 UC3.1	“Retrieve data from DigiAgro DS where sensor data from crop owners is collected and stored. “	Provide the REST command to connect to data sources	R & D	M
Section 2.2 UC3.2	“Integrate environmental data and weather forecast data seamlessly between the two dataspaces”	Provide the REST command to connect to data sources	R & D	M
Section 2.2 UC3.3	“Integration of data from DigiAgro DS with the models in	Provide the REST command to connect to data sources	R & D	M

	AgroScience DS.”			

9.1.10 Workflows

The following sub-sections describe the sequence diagrams of the Module

9.1.10.1 Read Data Sources

This feature provides the capability create the REST API call required to access a data set. The sequence diagram is shown in Figure 2: Data request

The main steps/functionalities are as follows:

- User selects data offering for importing from the metadata catalog. The selection process retrieves the metadata description for the data source, which includes the OpenAPI spec.
- User creates a natural language query to specify what data is required from the data offering and the metadata for the data offering, together with the query is sent to the Data Retriever component
- The Data Retriever formulates the required REST call based on the OpenAPI specification and machine learning analysis of the natural language query.
- The Data Retriever performs a validation of the created query, and if necessary, returns to the caller hints to improve the query.
- Requesting REST call sent to consumer’s DS2 Connector which passes it to the Producer Connector.

9.2 The REST call is forwarded to the data producer’s DS2 Connector

- Connector extracts data set from the data server
- Data returned to data consumer

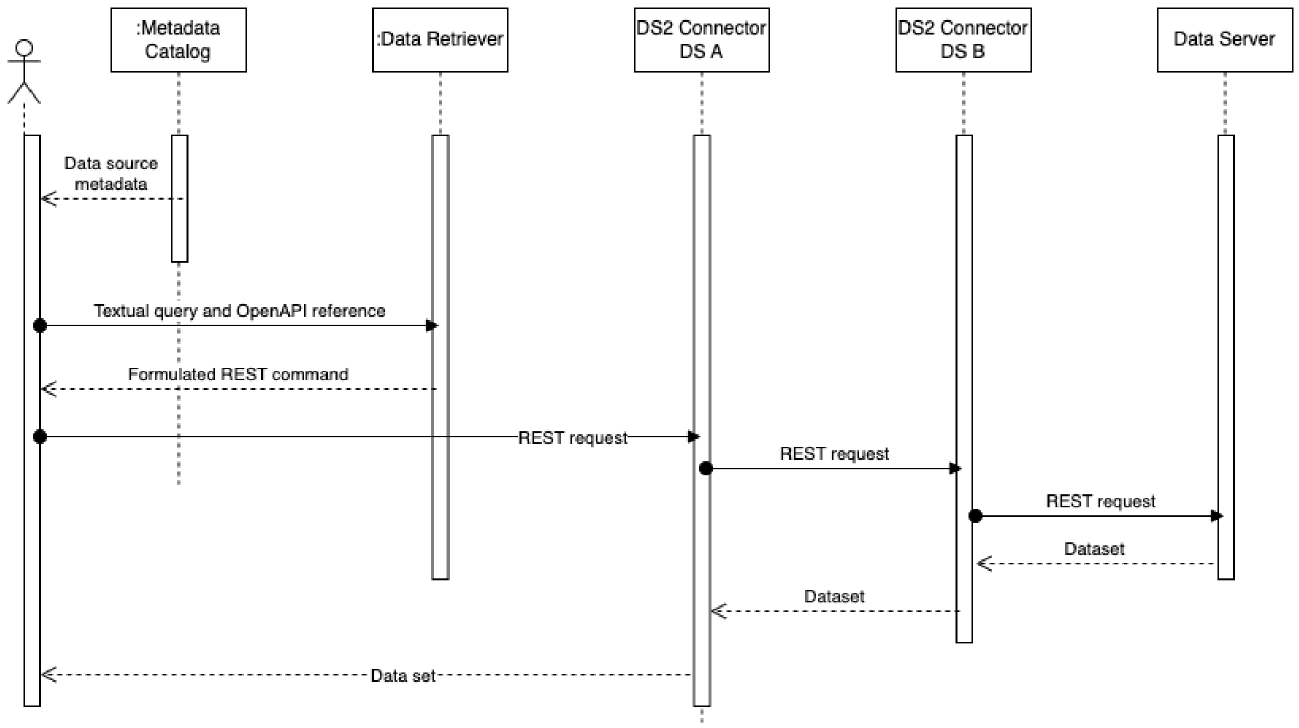


Figure 2: Data request

9.2.1 Role, Resourcing, and Milestones

Sub-component	Main Activity	M18	M24	M30	M36
Connectors (EDC, IDSA)	Analysis of the capabilities of existing Connector reference implementations in regard to data retrieval with dynamic REST parameters.	🟦			
Connectors	Implementation of Connector extensions required to support dynamic REST parameters		🟦		
Provider Connector	Support for data extraction requiring authentication keys		🟦		
Rest Evaluation/LLM retrieval	Better handling of bad-path results (e.g. due to incorrect user-supplied text)		🟦		
Integration testing	Integration testing for Green Deal use case		🟦		
Data Retriever	Expand to other use cases		🟦		
Data Retriever	Revisions to core functionality based on feedback from integration testing and use case performance		🟦		
DS2 componentization, final integration	DS2 componentization, final integration			🟦	
Table Total/DOA Task Total/Resilience	**Comments:**				

9.2.2 Open Issues

The following table summarise open issues/uncertainties that need to be resolved during the next stages or implementation.

Issue	Description	Next Steps	Lead or Related Component
Data sources from the use cases	Links to the datasets are required	Wait for use cases to publish the links	All use cases except Green Deal.
Connector	Extensions to the EDC connector are require to support the use cases. Minimally HTTP and potentially AWS needs to be supported	Currently checking within the partner organizations for the required expertise.	None
Catalog Metadata Data Subcomponent	OpenAPI specification is required when selecting the data source	Discussions with the VTT Catalog team indicate that the catalog metadata should be extendable.	Catalog (VTT)
Data Source Retrieval	In the event that data is required from a source which does not expose a REST endpoint, a REST server will need to be installed by and at the data provider site which will include a method for retrieving the stored data. An OpenAPI description for this interface will be created. Note that based on available Connector support, it might be decided to natively support AWS S3 format.	Will depend on use case data sources and the technical feasibility of supporting AWS	IBM/use case partner