Promoting and Incentivising Federated, Trusted, and Fair Sharing and Trading of Interoperable Data ASsets

D4.2
The PISTIS Product
Alpha version

Editor(s)	Marios Zacharias
Lead Beneficiary	SPH
Status	Final
Version	1.00
Due Date	30/09/2024
Delivery Date	17/10/2024
Dissemination Level	PU

Funded by the European Union under Grant Agreement 101093016. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

Project	PISTIS – 101093016
Work Package	WP4 - System Architecture, Continuous Integration, Testing and Technical Verification
Deliverable	D4.2 – The PISTIS Product – Alpha version
Contributor(s)	SPH, ASSENTIAN, ATHENA, ATOS, FHG, EUT, ICCS, Suite5, UBITECH
Reviewer(s)	Nikos Papagiannopoulos (AIA) Alexandrer Stocker (VIF)
Abstract	This deliverable accompanies the Alpha version of the PISTIS product, presenting the different components (by updating the D4.1 descriptions) that have been integrated into this first delivery of the platform.

Executive Summary

Deliverable D4.2 provides an overview of the 1st release of the platform, which is codenamed **Alpha version**. Due to the OTHER nature of the deliverable, the document at hand acts as an accompanying manuscript of the platform's code delivered in GitHub, aiming to describe at high level its synthesis.

The current release of the first prototype version of the Integrated PISTIS product is a low fidelity prototype, aimed to interconnect the different components that are being developed in WP2 and WP3, and serve the initial needs of the demonstrators for executing their scenarios.

Moreover, this deliverable provides an update on the architecture, as well as the changes of the different components of the platform, that have been initially described in D4.1 and evolved during the agile software development processes and the integration phases of the project. As in deliverable D4.1, these components are grouped into bundles and in this deliverable the main interactions between the different components are provided. For more information regarding each component, the reader is referred to WP2 and WP3 deliverables, D2.2 and D2.3 respectively.

Table of Contents

List of Figures

Terms and Abbreviations

ABAC	Attribute Based Access Control
ABE	Attribute Based Encryption
ADB	Asser Description Bundle
AI	Artificial Intelligence
API	Application Programming Interface
CF	Collaborative filtering
CRUD	Create, Read, Update, Delete
DCAT	Data Catalogue Vocabulary
DLT	Distributed Ledger Technology
DNN	Deep neural networks
DoA	Description of Action
DVDs	Data Value Dimensions
DVS	Data Valuation Service
eIDAS2	electronic IDentification, Authentication and trust Services 2
EU	European Union
FAIR	Findable, Accessible, Interoperable, Reusable
FTP	File Transfer Protocol
GDPR	General Data protection Regulation
GNN	Graph Neural Networks
HTTP	HyperText Transfer Protocol
ID	Identity
IDS	International Data Spaces
IOTA	Internet of Things Application
JSON	JavaScript Object Notation
JWT	JSON Web Token
kNN	k-Nearest Neighbour
LSH	Locality-Sensitive Hashing
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
MVP	Minimum Viable Product
OIDC	OpenID Connect
PROV	Provenance
RBAC	Role Based Access Control
RDF	Resource Description Framework
REST	Representational state transfer
SE	Searchable Encryption
SQL	Structured Query Language
SSI	Self-Sovereign Identity
ToC	Table of Contents
UUID	Universal Unique Identifier
WP	Work Package
XAI	eXplainable AI
YAML	Yet Another Modelling Language

1 INTRODUCTION

The current deliverable presents the initial prototype of the PISTIS platform called Alpha.

This prototype is the cumulative outcome of the WP4 activities of the project until M21, as specified in the project's DoA and will serve as the initial proof of concept of the platform, to test and achieve integration between components, as well as the first version of code to be deployed to the PISTIS demonstrators to execute their scenarios.

1.1 DOCUMENT STRUCTURE

D4.2 is structured as follows:

- Section 1 is the introduction and this document structure description.
- Section 2 provides an update on the PISTIS architecture, as initially presented in D4.1.
- Section 3 describes the Alpha version of the PISTIS platform, discussing also the main 3 flows of operation which are defined in the PISTIS concept.
- Section 4 provides the conclusions of this document.
- Sections 5-15 (as annexes) provide the updated descriptions of the different components, based on the original component descriptions that were part of D4.1.

2 UPDATED PISTIS HIGH LEVEL ARCHITECTURE

The overall PISTIS platform is envisioned as a federated environment of collaborating stakeholders, governed by a central cloud-based entity.

PISTIS constitutes a Data Space where stakeholders will be operating a "PISTIS Data Factory" deployment at their premises (locally or on own cloud resources) (coloured green in the following figure with blue colour indicating the data storage facilities of each stakeholder) that is used to manage their own data, all connected through the "PISTIS Cloud Platform" (coloured orange in the following figure), which is used to provide services that facilitate data monetisation and transaction execution. This macroscopic view of the architecture is shown in the next figure. Blockchain nodes are made available to facilitate transactions, however their presence in the network does not need to follow a 1:1 matching with the Data Factories.

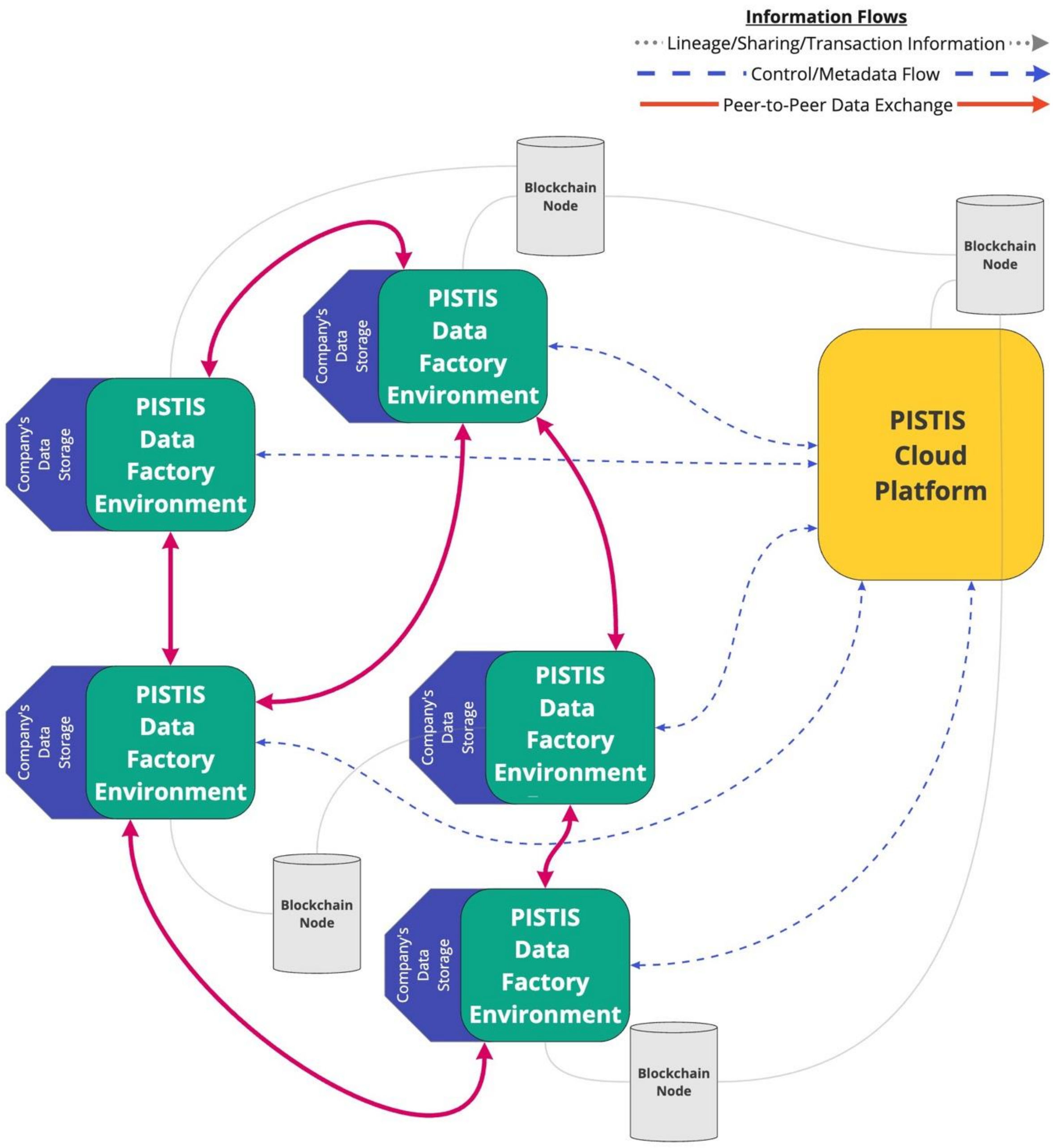


Figure 1: Macroscopic view of PISTIS Architecture

It is noted that the actual flow of data (e.g. of the “data assets”) is performed only via a peer-to-peer communication channel set-up between the different “PISTIS Data Factories” of the Data Provider and of the Data Consumer, increasing in this manner the overall security and trust of the PISTIS platform and catering for data sovereignty and privacy.

Figure 2 gives a closer look at the PISTIS architecture, as it evolved after the delivery of D4.1. It shows the various bundles and the components within them. Additionally, it illustrates the main data, metadata, and control flows across the entire platform.

The main changes from the architecture presented in D4.1 is the placement of some components, which are part of the Data Factory environment in the latest architecture (realised by the Alpha version of the platform). The reason behind moving these components from the central environment to the Data Factories had to do with the need to further strengthen data confidentiality and privacy, as in such a way no data (or metadata) would be leaving the Data Factory, unless it is necessary. For example, the preparatory stages of the publication flow (see section 3 below) happen entirely at Data Factory level, and the Cloud platform is engaged only at the end of the process where the user confirms to add a new listing to the PISTIS catalogue

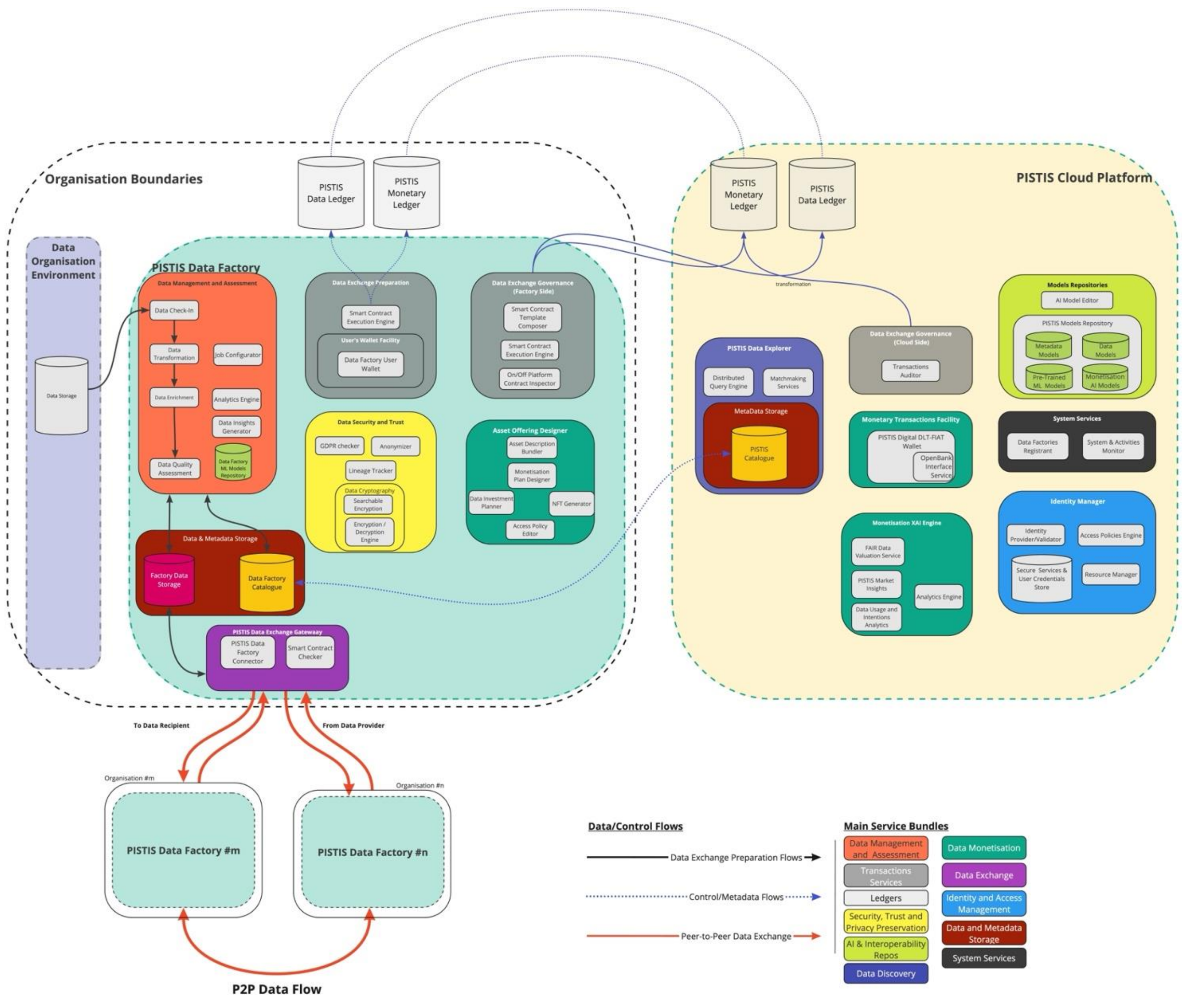


Figure 2: PISTIS Architecture

2.1 Bundles Composing PISTIS

As shown in Figure 2, the PISTIS platform consists of the following bundles:

- **Data Management and Assessment bundle**, that is responsible for the collection of data from existing repositories available to an organisation, the refinement, transformation and improvement of data, judging also its quality and providing services to improve it and make it interoperable.
- **Data & Metadata Storage bundle**, that is delivering a catalogue for the data available for each organisation and those that are made available as "published" data over the whole ecosystem, alongside with the appropriate data storage facilities to hold the data.
- **Data Discovery bundle**, that provides services for searching and discovering the available data assets that might be of interested to a Data Consumer
- **Data Exchange bundle**, that facilitates the peer-to-peer exchange of the data assets between a Data provider and a Data Consumer, adhering to the terms of the contract that has been signed to govern the overall transaction.
- **Data Monetisation bundle**, that provides different services for the Data provider to understand the value of its data within the market environment of the platform and to place the data on the PISTIS Data Catalogue using different methods to monetise by engaging in transactions performed with the aid of blockchain technology.
- **Security, Trust & Privacy Preservation bundle**, that is offering services for strengthening data security and privacy.
- **Transaction Services bundle**, that provides all the necessary functionalities to author, execute and validate the different transactions with the use of smart contracts.
- **Ledgers bundle**, that provide the necessary functionality for storing the data and the monetary transactions.
- **AI & Interoperability Repos bundle**, that provides the different repositories for storing and propagating different models (data models, AI models and metadata models) that need to be consumed by the various components.
- **Identity & Access Management bundle**, that governs identity provisioning and validation and access management both at data and component level.
- **System Services bundle**, that is used by system administrators to configure and monitor the overall PISTIS environment.

The different components of these bundles that were initially described in D4.1 and have been updated since, are presented in the annexes, alongside with the description of their API endpoints/functionalities, described using the Swagger toolchain, and adhering to the OpenAPI Specification 3.0.

It should be stressed out that due to PISTIS following the agile software development method of work, the current architecture, and the current description of the different components, are subject to changes, depending on the course of the development activities and technical decisions that will be made during the project to reach the most viable and valuable product that can satisfy the needs of the users.

2.2 ARCHITECTURE'S ON-LINE RESOURCES

The current state of the architecture, as well as the documentation of all APIs of the different components is also provided as an online resource in the project's GitHub repository.

https://github.com/orgs/PISTIS-Platform/

This is a public repository, which will be used as an online resource not only for the project's technical teams, but also for all interested stakeholder that would like to learn more about PISTIS and fork the code that will be developed during the project. The Alpha version of PISTIS is managed in the private repositories, access to which can be provided on demand.

3 The PISTIS Platform – Alpha Version

3.1 PISTIS Platform Overview

The PISTIS platform is the overall technological output of the project that aims to provide a platform to stakeholders to share and monetise their data, using novel monetisation methods that are built around trustworthy technologies.

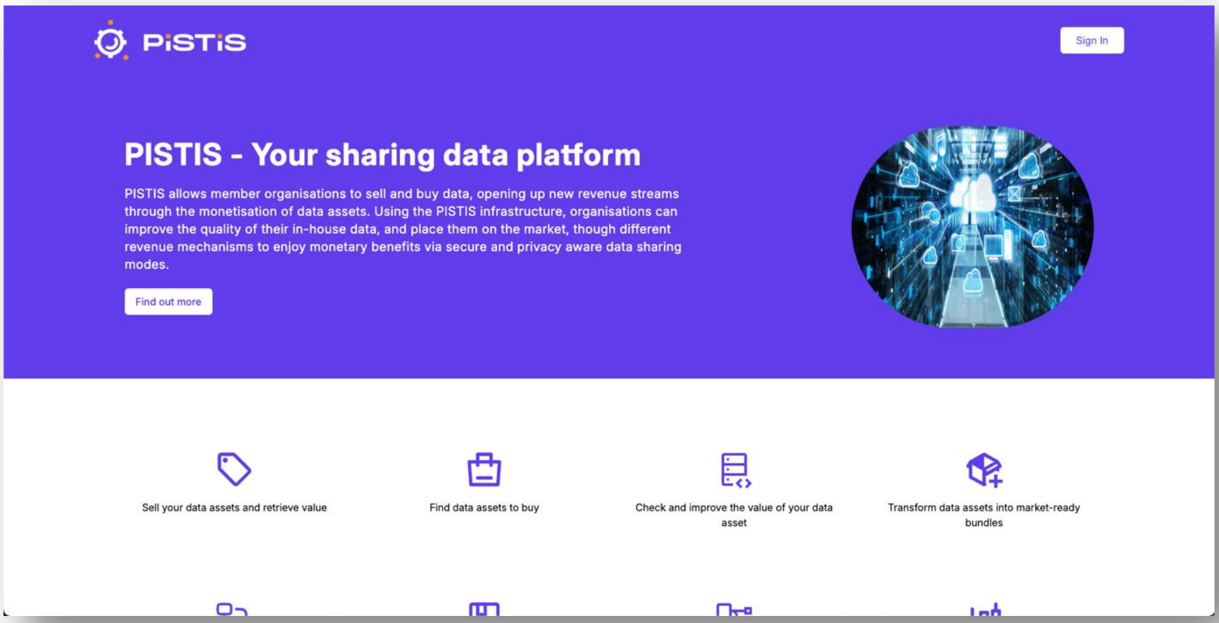
As described in the architecture, PISTIS works on the principle of federation, maximising data privacy and sovereignty, and this is expressed by the notion of various “Data Factories” that exchange data in a peer-to-peer manner (e.g. with no intermediary actor involved), while all adhering to a predefined set of rules and guidelines that is governed by a central entity “the Cloud platform” which is the key link between all involved organisations in the PISTIS ecosystem.

As such, the alpha version of the PISTIS platform consists of:

- The PISTIS Cloud Platform which is a centrally hosted environment that is used for the management of the ecosystem, control of transactions and the exposure of catalogue data.
- The PISTIS Data Factory which is an infrastructure that is hosted at the premises of each stakeholder, and is maintained and managed by the same actor, and which connects to the Cloud Platform for it to become a member platform of the PISTIS ecosystem

The URL of the PISTIS Cloud platform is: http://www.pistis-market.eu

At this point, access to the platform is only available for project’s partners

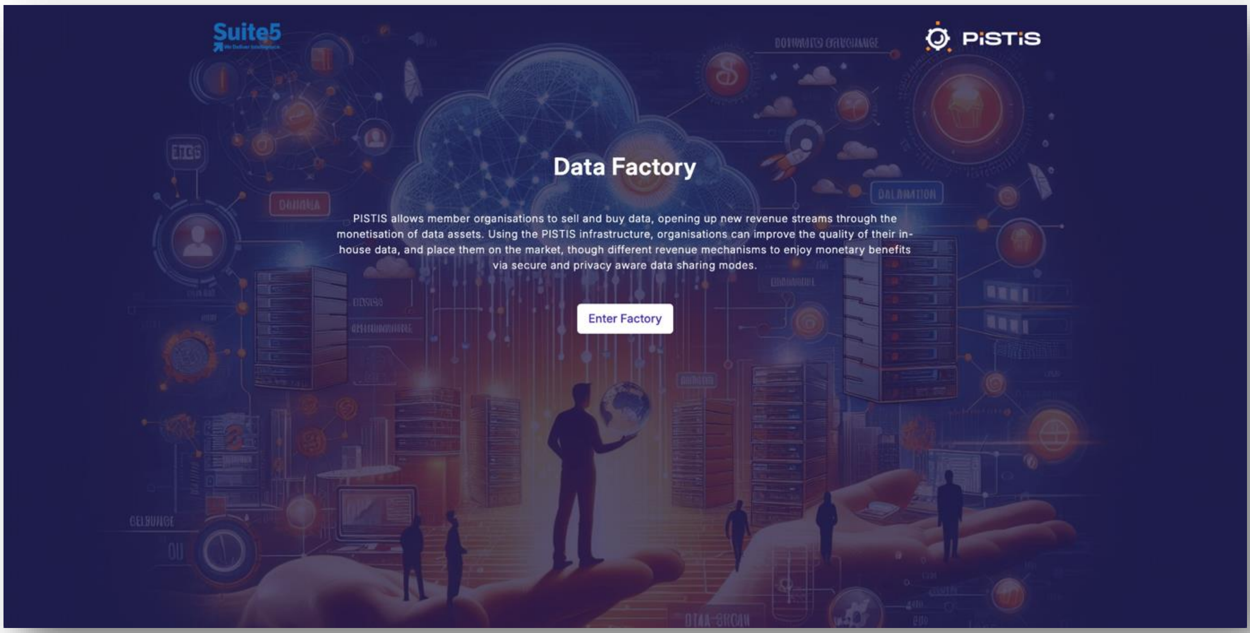

PiSTiS
Sign In
Find out more
Sell your data assets and retrieve value
Find data assets to buy
Check and improve the value of your data asset
Transform data assets into market-ready bundles

PISTIS - Your sharing data platform

PISTIS allows member organisations to sell and buy data, opening up new revenue streams through the monetisation of data assets. Using the PISTIS infrastructure, organisations can improve the quality of their in-house data, and place them on the market, though different revenue mechanisms to enjoy monetary benefits via secure and privacy aware data sharing modes.

Figure 3: PISTIS Cloud platform landing page

The PISTIS Data Factory, as identified in the architecture, is the environment that is hosted in the premises of each organisation (locally, or on any other cloud provider). This infrastructure incorporates all the components that have to do with data registration and onboarding, as well as part of the components that have to do with placing the data on the market, and of the ones that are required to execute the transfer of the data from one Data Factory environment to another (e.g. in the case of a seller and a buyer). The following figure shows tea landing page of a Data Factory.



Data Factory

PISTIS allows member organisations to sell and buy data, opening up new revenue streams through the monetisation of data assets. Using the PISTIS infrastructure, organisations can improve the quality of their in-house data, and place them on the market, though different revenue mechanisms to enjoy monetary benefits via secure and privacy aware data sharing modes.

Figure 4: PISTIS Data Factory landing page

3.2 Core Workflows over the PISTIS Platform

PISTIS is a holistic platform that can be used to monetise on data that one stakeholder (seller or data owner) can provide to another interested party (buyer or data consumer), over a marketplace, where monetisation rules are clear, and transactions can be executed automatically, without the need to negotiate over prices and offers.

As such, PISTIS can be considered as a secure “App Store for Data”, where owners can create listing by exposing information about their assets (but only on metadata level), and consumers can select to buy these, paying directly the asking price and then fetching the data directly from the Data Factory of the owner.

This overall concept can be distinguished in 3 major flows that handle the Data Registration and Processing to the data factory, the (meta)Data Publication to the Cloud platform, and the eventual Data Discovery & Acquisition which are described below and include the major components of the platform that are required to set up and finalise a data transaction. Other components which are part of the architecture might not be part of these flows (as described below), as they are utilised either for management purposes, or are providing extra insights to data owners or consumers but are not considered a requirement for the execution of a transaction.

3.2.1 Data Registration Flow

This flow includes all the prerequisite activities that need to happen before a data owner chooses to create a listing and are revolving around uploading the data to the Data Factory of the data owner, and the treatment of those data to transform them into quality assets that hold more possibilities to be sold over the platform.

This flow starts with the onboarding (via upload) of data into the platform, and the next steps have to do with transforming the data, identifying insights within the data, semantically enriching them and mapping them to a data model, and checking their quality. During those steps, data are stored in the data storage facility of the Data Factory and lineage is tracked, and the final output becomes part of the Factory Data Catalogue (at metadata description level) and they data itself is stored in the Data Factory Storage. The above-mentioned components are orchestrated by the Job Configurator Component. Other optional steps in this flow include the GDPR checking of the data, as well as the Anonymisation of the data, while encryption methods might also be applied, to increase the security of the data, and make them also compliant with possible regulations that might exist regarding the retention and storing of data.

In principle, during this step, the following PISTIS components are involved:

- Data Check-In
- Data Transformation
- Job configurator
- Analytics Engine
- Data Enrichment
- Data Quality Assessment
- Data Insights Generator
- Data Factory Catalogue
- Factory Data Storage
- Anonymizer
- Lineage Tracker
- GDPR checker
- Searchable Encryption
- Encryption/Decryption Engine
- On/Off Platform Contract Inspector
- Smart Contract Execution Engine

The overall flow is described in the next Figure.

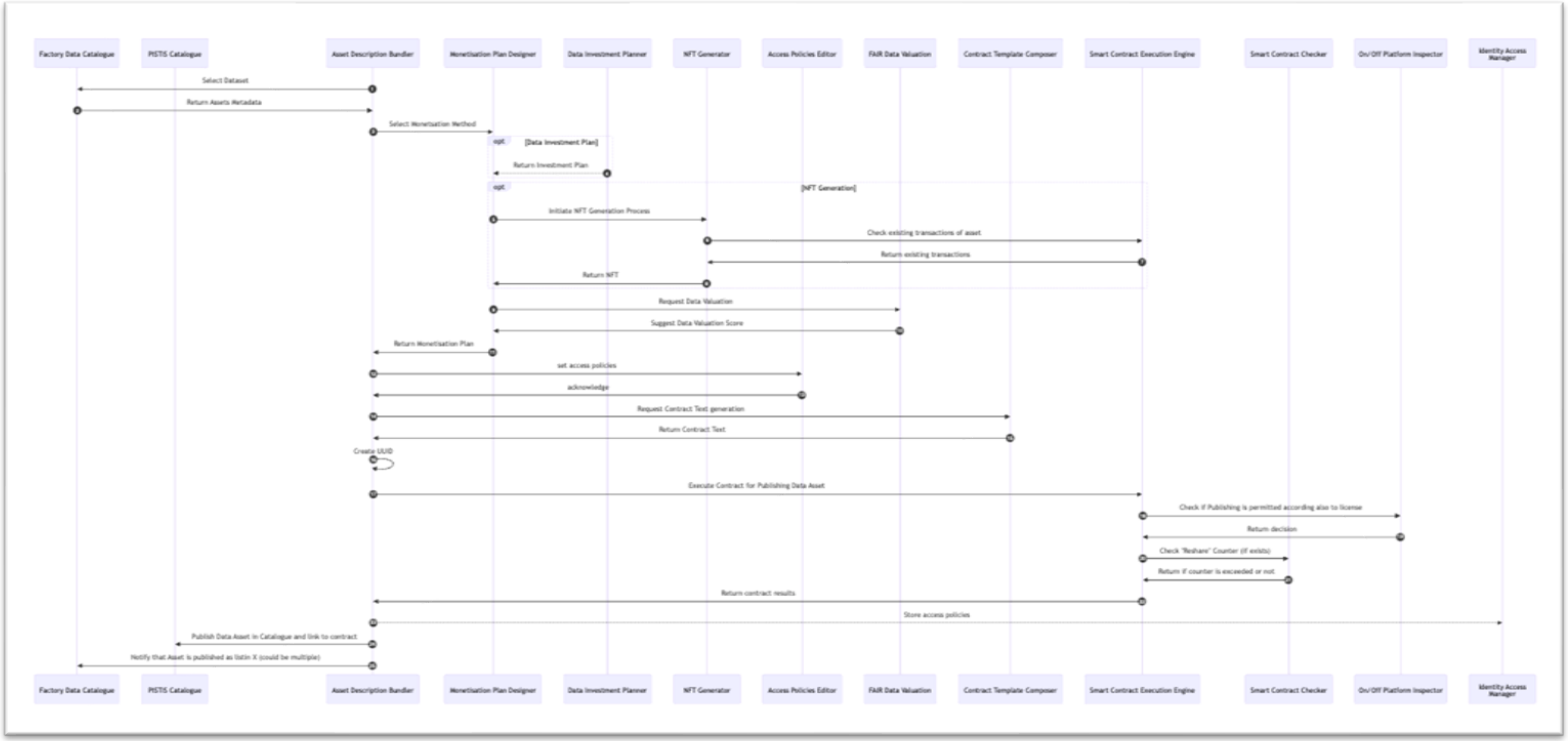

Factory Data Catalogue
PISTIS Catalogue
Asset Description Bundler
Monetisation Plan Designer
Data Investment Planner
NFT Generator
Access Policies Editor
FAIR Data Valuation
Contract Template Composer
Smart Contract Execution Engine
Smart Contract Checker
On/Off Platform Inspector
Identity Access Manager
Select Dataset
Return Assets Metadata
Select Monetisation Method
opt
[Data Investment Plan]
Return Investment Plan
opt
[NFT Generation]
Initiate NFT Generation Process
Check existing transactions of asset
Return existing transactions
Return NFT
Request Data Valuation
Suggest Data Valuation Score
Return Monetisation Plan
set access policies
acknowledge
Request Contract Text generation
Return Contract Text
Create UUID
Execute Contract for Publishing Data Asset
Check if Publishing is permitted according also to license
Return decision
Check 'Reshare' Counter (if exists)
Return if counter is exceeded or not
Return contract results
Store access policies
Publish Data Asset in Catalogue and link to contract
Notify that Asset is published as listin X (could be multiple)
Factory Data Catalogue
PISTIS Catalogue
Asset Description Bundler
Monetisation Plan Designer
Data Investment Planner
NFT Generator
Access Policies Editor
FAIR Data Valuation
Contract Template Composer
Smart Contract Execution Engine
Smart Contract Checker
On/Off Platform Inspector
Identity Access Manager

Figure 5: Data Registration Flow Diagram

3.2.2 Data Publication Flow

The Data Publication flow can be triggered once data is available in the Data Factory of a user, and he would like to make a listing. The user selects the data out of their own Data Catalogue and triggers the components to set the access policies for this dataset when it becomes available on the PISTIS Catalogue, defines the monetisation methods, checks the data valuation suggestions for the specific dataset and finally creates a bundle of the data asset and sends it to the PISTIS Data Catalogue so that it becomes available for acquisition by the, in the contract settings, permitted users, while at the same time this listing alongside with the template of the acquisition contract terms are also stored in the blockchain.

In principle, during this step, the following PISTIS components are involved:

- Data Factory Catalogue
- PISTIS Catalogue
- Distributed Query Engine
- Matchmaking Services
- Data Monetisation Bundle
- Asset Description Bundler
- FAIR Data Valuation Service
- Data Investment Planner
- PISTIS Market Insights
- Monetisation Plan Designer
- Data Usage and Intentions Analytics
- On/Off Platform Contract Inspector
- Smart Contract Template Composer
- Smart Contract Execution Engine
- PISTIS Data Ledger

The overall flow is described in the next Figure

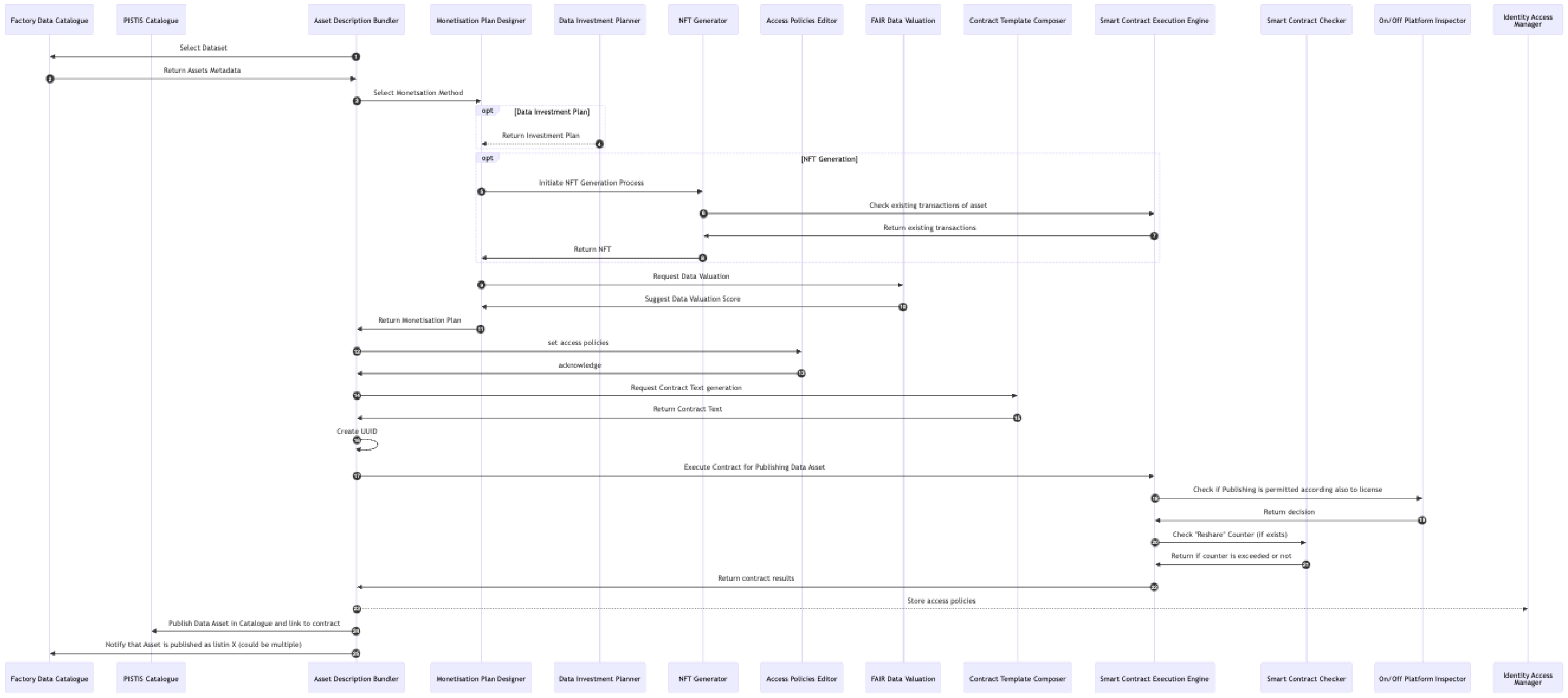

Factory Data Catalogue
PISTIS Catalogue
Asset Description Bundler
Monetisation Plan Designer
Data Investment Planner
NFT Generator
Access Policies Editor
FAIR Data Valuation
Contract Template Composer
Smart Contract Execution Engine
Smart Contract Checker
On/Off Platform Inspector
Identity Access Manager
Select Dataset
Return Assets Metadata
Select Monetsation Method
opt
[Data Investment Plan]
Return Investment Plan
opt
[NFT Generation]
Initiate NFT Generation Process
Check existing transactions of asset
Return existing transactions
Return NFT
Request Data Valuation
Suggest Data Valuation Score
Return Monetisation Plan
set access policies
acknowledge
Request Contract Text generation
Return Contract Text
Create UUID
Execute Contract for Publishing Data Asset
Check if Publishing is permitted according also to license
Return decision
Check "Reshare" Counter (if exists)
Return if counter is exceeded or not
Return contract results
Store access policies
Publish Data Asset in Catalogue and link to contract
Notify that Asset is published as listin X (could be multiple)

Figure 6: Data Publication Flow Diagram

3.2.3 Data Discovery & Acquisition Flow

This flow includes the steps that are relevant to the monetary transactions and the data transfer. As such, this flow commences with a data consumer searching over the catalogue and identifying a dataset and committing to acquire it. The monetary transactions take place after a set of checks, and this transaction is recorded in the IOTA and the PISTS ledgers and the data asset is becoming available to the user. At this point, the Data Factory Connector takes place and after performing the necessary checks to check the validity of a transaction, copies the data payload from the Data Factory of the buyer to that of the seller (or fully transfers it in the case this was an NFT-based transaction).

In principle, during this step, the following PISTIS components are involved:

- Data Catalogue
- Distributed Query Engine
- Matchmaking Services
- Factory Data Storage
- PISTIS Data Factory Connector
- Smart Contract Checker
- PISTIS Digital DLT-FIAT Wallet
- Encryption/Decryption Engine
- Smart Contract Execution Engine
- Data Factory User Wallet
- PISTIS Data Ledger
- PISTIS Monetary Ledger

The overall flow is described in the next Figure.

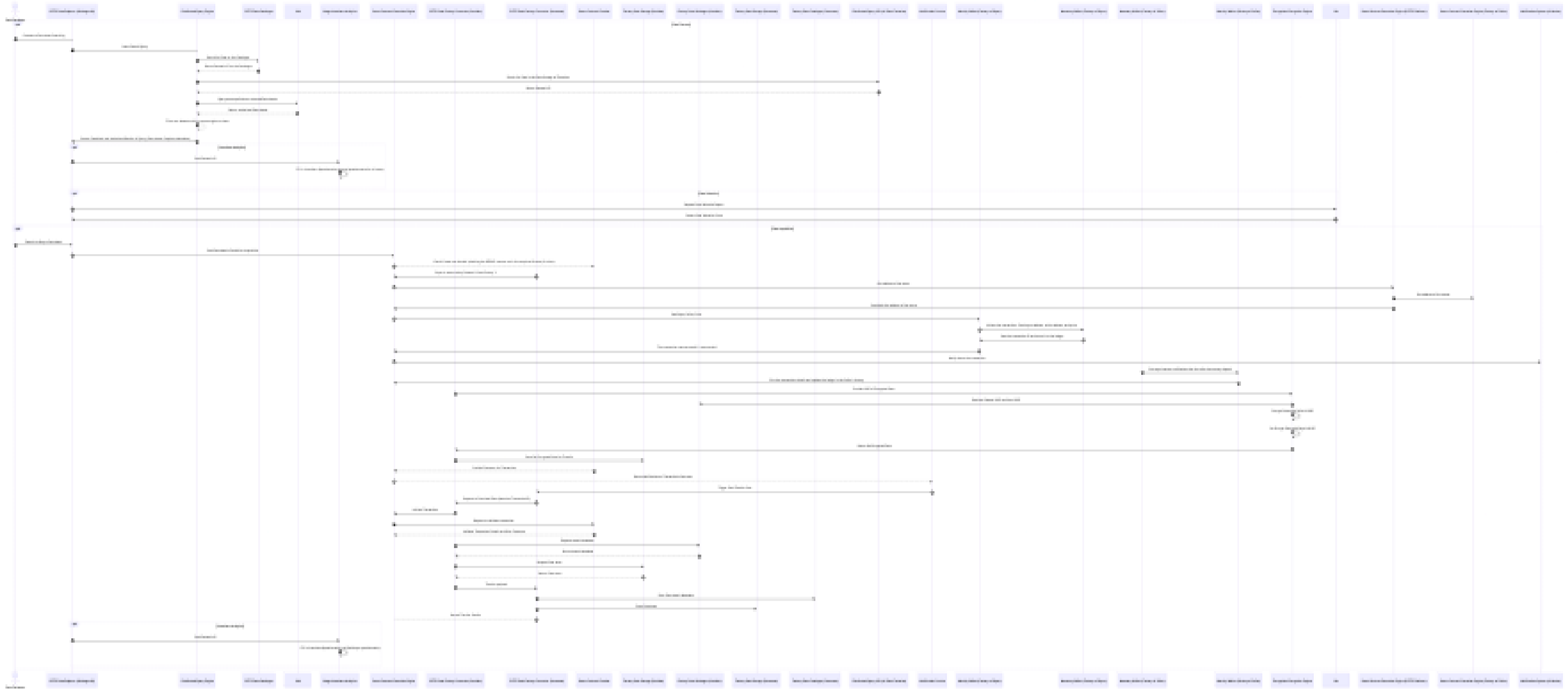


Figure 7: Data Discovery & Acquisition Flow Diagram

3.3 Platform Documentation

Platform user and developer documentation is available only to registered users at: https://docs.pistis-market.eu

4 CONCLUSIONS

This document comes as an accompanying document of deliverable D4.2 constitutes the Alpha version of the PISTIS platform that is delivered in M21 of the project.

The platform, as described above, is the initial prototype version developed by the technical partners and will be used by the demonstrators to experiment with their scenarios and provide feedback to the development team to improve existing features, as well as to add new, or re-rank the user stories available in the backlog document.

During the next period of the project, two major releases are planned (beta and v1.00) that will be the basis of the product to be exploited after the end of the project and will be the main released to be used by the demonstrators. In these releases, the current components of the Alpha version will be improved, while also new features will be added to be able to fully satisfy the needs of the different demonstrators. Furthermore, the resource utilization aspects relevant to each Data Factory will be investigated, to minimize the required resources for setting up the environment.

ANNEXES – UPDATES ON THE D4.1 BUNDLE DESCRIPTIONS

5 DATA MANAGEMENT AND ASSESSMENT BUNDLE

The Data Management and Assessment bundle, that is responsible for the collection of data from existing repositories available to an organisation, the refinement, transformation, and improvement of data, judging also their quality and providing services to improve them and make them interoperable.

This bundle consists of the following components:

- Data Check-In
- Data Transformation
- Job configurator
- Analytics Engine
- Data Enrichment
- Data Quality Assessment
- Data Insights Generator

These are presented in the following sub-sections.

5.1 DATA CHECK-IN

Data Check-in will enable support for data sources that will supply data for the solution workflow, as data sources can come in a variety of forms (repositories, data streaming flows, and so on).

Data Check-In will serve as input for the whole data workflow in the PISTIS platform allowing several ways for data ingestion which must include the followings:

- File upload
- FTP Server
- Data Space connectors:
  - API
  - KAFKA
  - MQTT

Data Check-In offering in terms of functionalities is detailed below:

- *UploadData*: this functionality should allow the end user to provide a data file to be stored in a server to be consumed by the subsequent data processing workflow defined in the PISTIS platform. Some basic verifications could be carried out to check some requirements regarding the data file provided (e.g., size limits, data formats, etc.).
- *GetDataFromFTP*: in case the data to be ingested by the workflow is stored in an FTP server, a method will be provided to retrieve that data. This method should be called

providing all the information needed to get access to the data (I.e., endpoint, path to the file, filename, required credentials, etc.).

- *GetDataFromDataSpace*: in this case, connectors compatible with GAIA-X and IDS could be implemented to retrieve data stored in data spaces, providing a similar functionality to the one of the data check in from an FTP server. It will be required, as in the previous case, to get all the details (as well as credentials when necessary) needed to get access to the required data source.
- *GetDataFromSubscription*: the possibility to subscribe as a client to a Kafka topic[1] is also being evaluated, allowing to get data from this kind of data source (I.e. Kafka or MQTT topics). By means of these, data can be consumed following a given criteria (e.g., defining a time window, a data limit, etc.) and then set for its processing.

5.1.1 Relation to Other Components

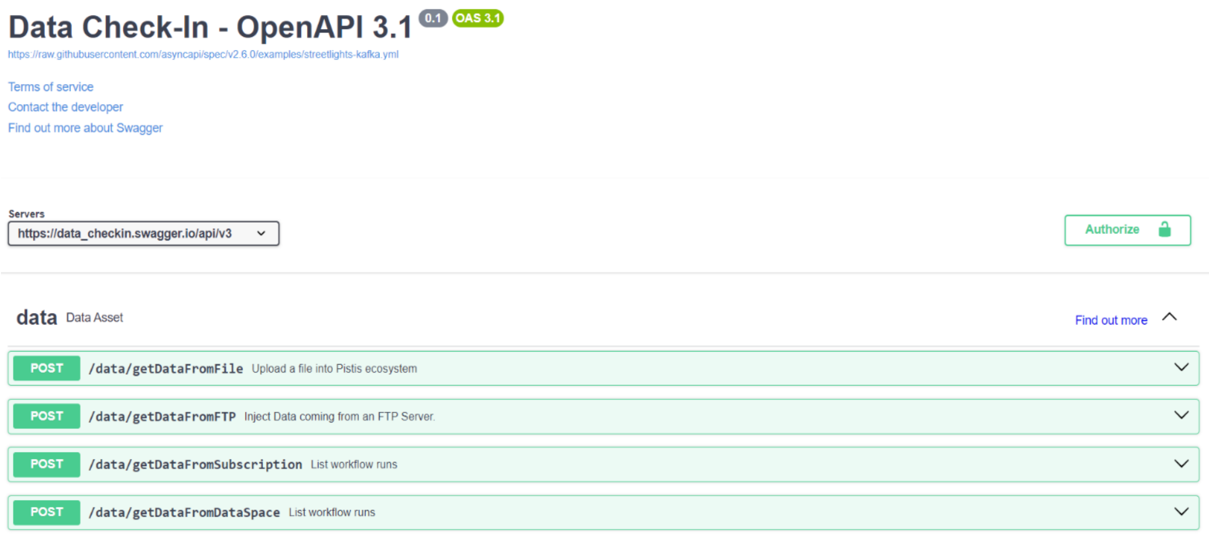
5.1.1.1 Components providing Input to Data Check-In

Input Required	Component providing the Input	Communication Method
DataSet	Job Configurator	API

5.1.1.2 Components to which Data Check-In provides input

Output Served	Component ingesting the output	Communication Method
DataSet	Factory Storage	API
Status	Job Configurator	API

5.1.2 API Calls Descriptions


Data Check-In - OpenAPI 3.1 0.1 OAS 3.1

https://raw.githubusercontent.com/asyncapi/spec/v2.6.0/examples/streetlights-kafka.yml

Terms of service
Contact the developer
Find out more about Swagger

Servers: https://data_checkin.swagger.io/api/v3

data Data Asset — Find out more

- POST /data/getDataFromFile Upload a file into Pistis ecosystem
- POST /data/getDataFromFTP Inject Data coming from an FTP Server.
- POST /data/getDataFromSubscription List workflow runs
- POST /data/getDataFromDataSpace List workflow runs

[1] https://kafka.apache.org/intro - Accessed 29/12/2023

5.1.3 Sequence Diagrams

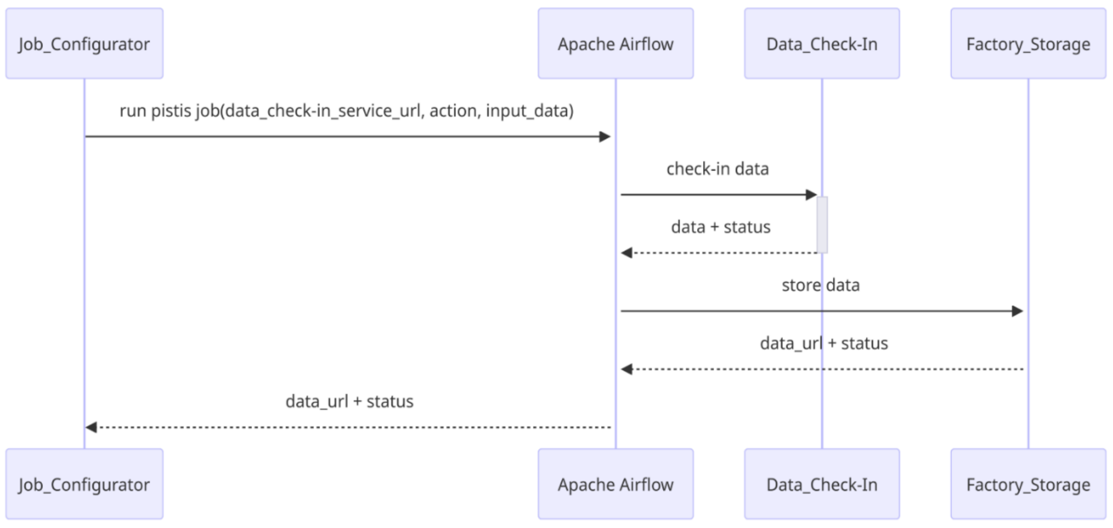


Figure 8: Data Check-In Sequence Diagram

5.2 Data Transformation

Data transformation component aims at providing the possibility of performing some preprocessing tasks on the datasets to be handled by the PISTIS platform. These transformations can be very useful to improve the quality of the dataset, handling some aspects of the dataset that are commonly considered to diminish its value (e.g., missing values, wrong values, unformatted strings, etc.).

To perform that dataset preprocessing, a set of transformations can be defined to be applied over the dataset. Accordingly, the activities to be performed for each transformation to be applied are setting up the chosen transformation, including some specific settings depending on the given transformation, some filtering on the fields or registries to be transformed, etc.

5.2.1 Relation to Other Components

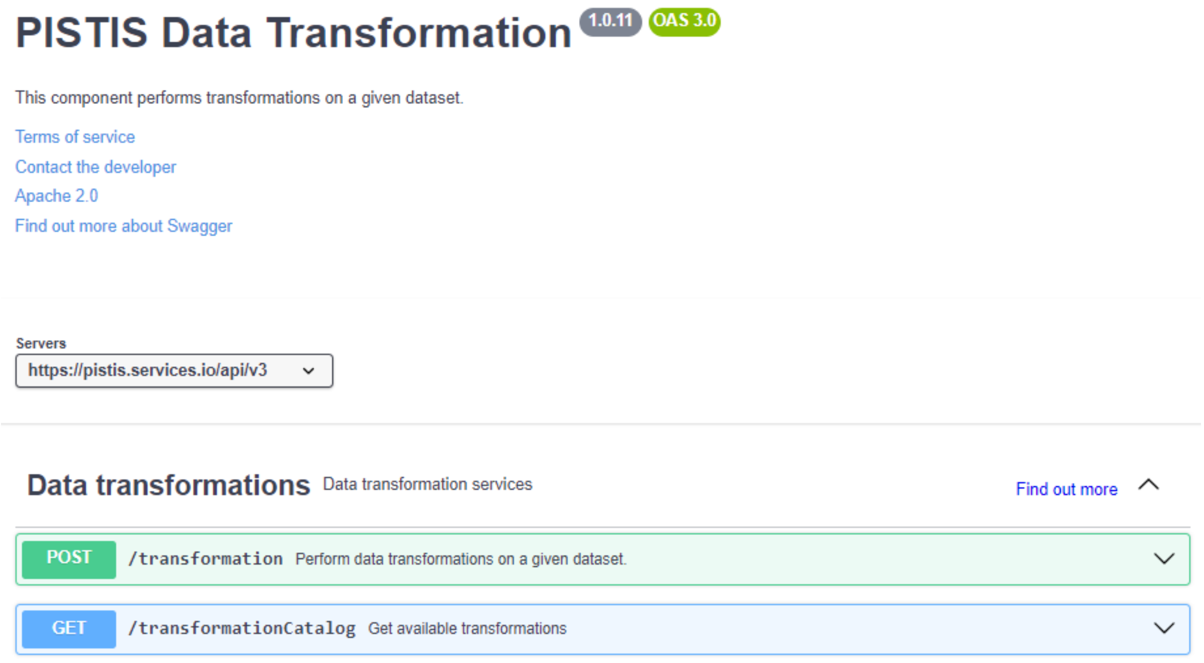
5.2.1.1 Components providing Input to Data Transformation

Input Required	Component providing the Input	Communication Method
Dataset	Job configurator	API
Transformation definition	Job configurator	API

5.2.1.2 Components to which Data Transformation provides input

Output Served	Component ingesting the output	Communication Method
Modified dataset	Job configurator	API

5.2.2 API Calls Descriptions


PISTIS Data Transformation 1.0.11 OAS 3.0

This component performs transformations on a given dataset.

Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger

Servers
https://pistis.services.io/api/v3

Data transformations Data transformation services Find out more

POST /transformation Perform data transformations on a given dataset.

GET /transformationCatalog Get available transformations

5.2.3 Sequence Diagrams

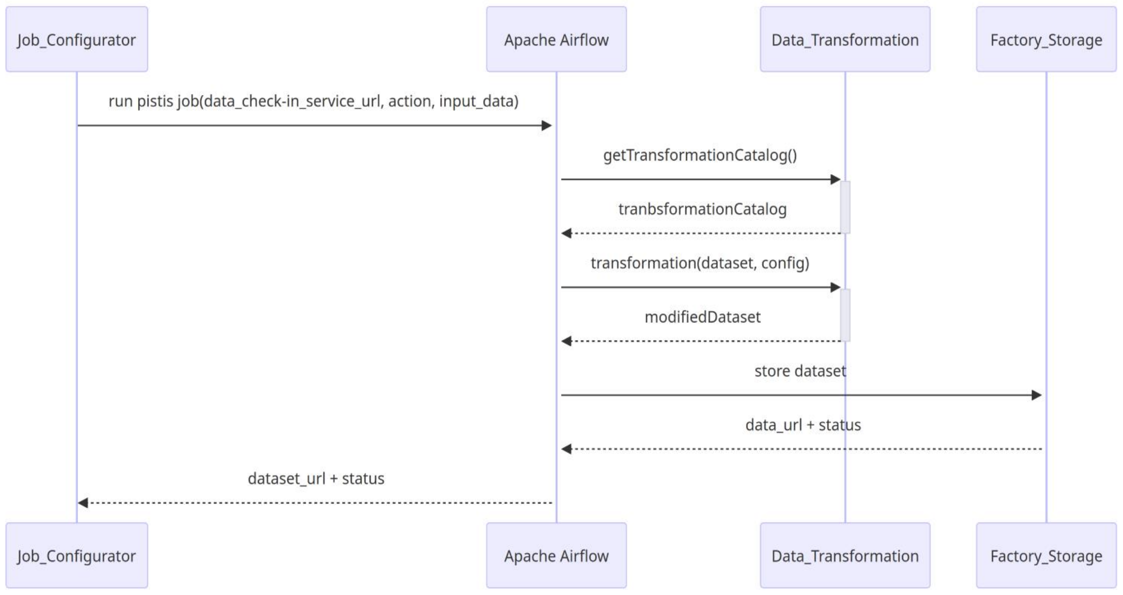


Figure 9: Data Transformation Sequence Diagram

5.3 Job Configurator

The Job Configurator oversees defining templates to support data pipeline jobs, as well as orchestrating them through the development of complicated workflows.

Job Configurator provides a high-level format for defining workflow and jobs to select only supporting those formats accepted by the workflow orchestration tool, which is Apache Airflow.

5.3.1 Relation to Other Components

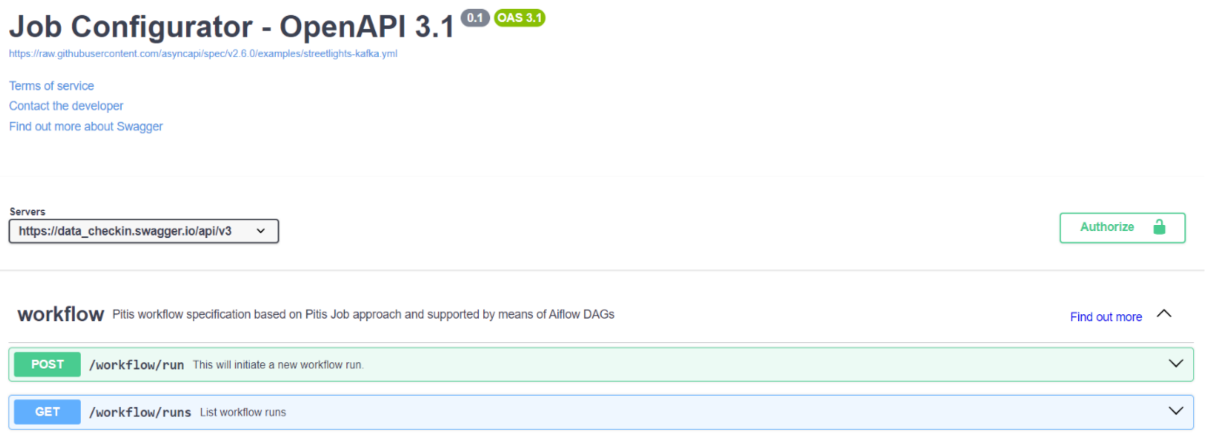
5.3.1.1 *Components providing Input to Job Configurator*

Input Required	Component providing the Input	Communication Method
PISTIS workflow specification	PISTIS Dashboard	API

5.3.1.2 *Components to which Job Configurator provides input*

Output Served	Component ingesting the output	Communication Method
DataSet	Data Factory Storage	API
Data Trace	Linage Tracker	API

5.3.2 API Calls Descriptions


Job Configurator - OpenAPI 3.1 0.1 OAS 3.1

https://raw.githubusercontent.com/asyncapi/spec/v2.6.0/examples/streetlights-kafka.yml

Terms of service
Contact the developer
Find out more about Swagger

Servers
https://data_checkin.swagger.io/api/v3

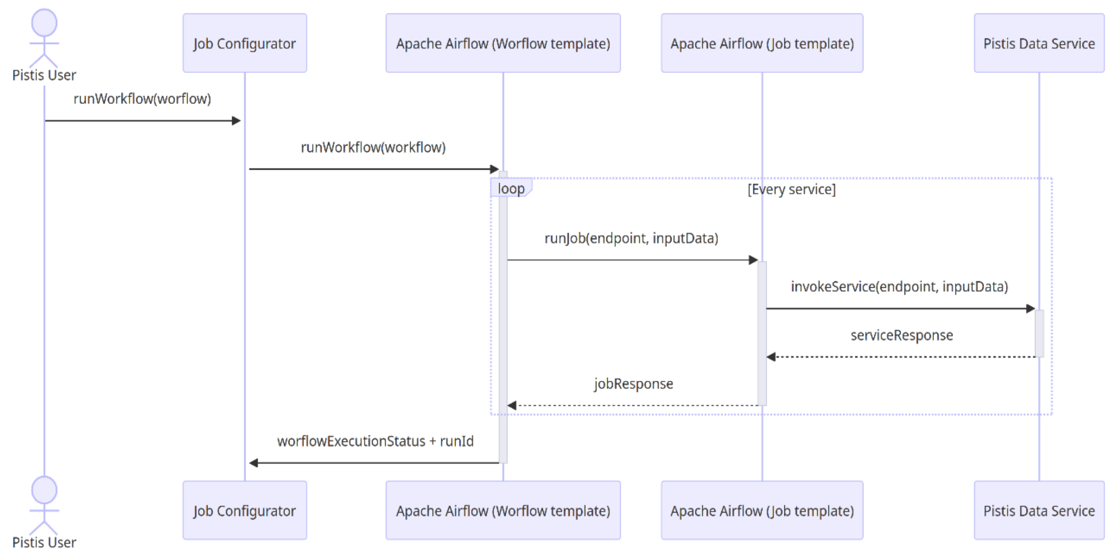
Authorize

workflow Pitis workflow specification based on Pitis Job approach and supported by means of Aiflow DAGs — Find out more

POST `/workflow/run` This will initiate a new workflow run.

GET `/workflow/runs` List workflow runs

5.3.3 Sequence Diagrams



Pistis User
Job Configurator
Apache Airflow (Worflow template)
Apache Airflow (Job template)
Pistis Data Service
runWorkflow(worflow)
runWorkflow(workflow)
loop
[Every service]
runJob(endpoint, inputData)
invokeService(endpoint, inputData)
serviceResponse
jobResponse
worflowExecutionStatus + runId

Figure 10: Job Configurator Sequence Diagram

5.4 Analytics Engine

The Analytics Engine Facilitator is a tool that customizes and sets up analytics using MLFlow for analysis and Apache Superset for visualization. It takes a set of starting datasets and creates a personalized analytic playground. This allows the Data Consumer to easily run analytics on the provided datasets in a separate environment, producing valuable results and metrics.

Furthermore, the presence of an analytics engine will enable other modules of the overall PISTIS environment to benefit from its functionalities, such as enabling automatic data transformations, accommodating ML-based anonymisation activities, and running analyses relevant to the PISTIS market, such as trend identifications, predictions, and so on.

5.4.1 Relation to Other Components

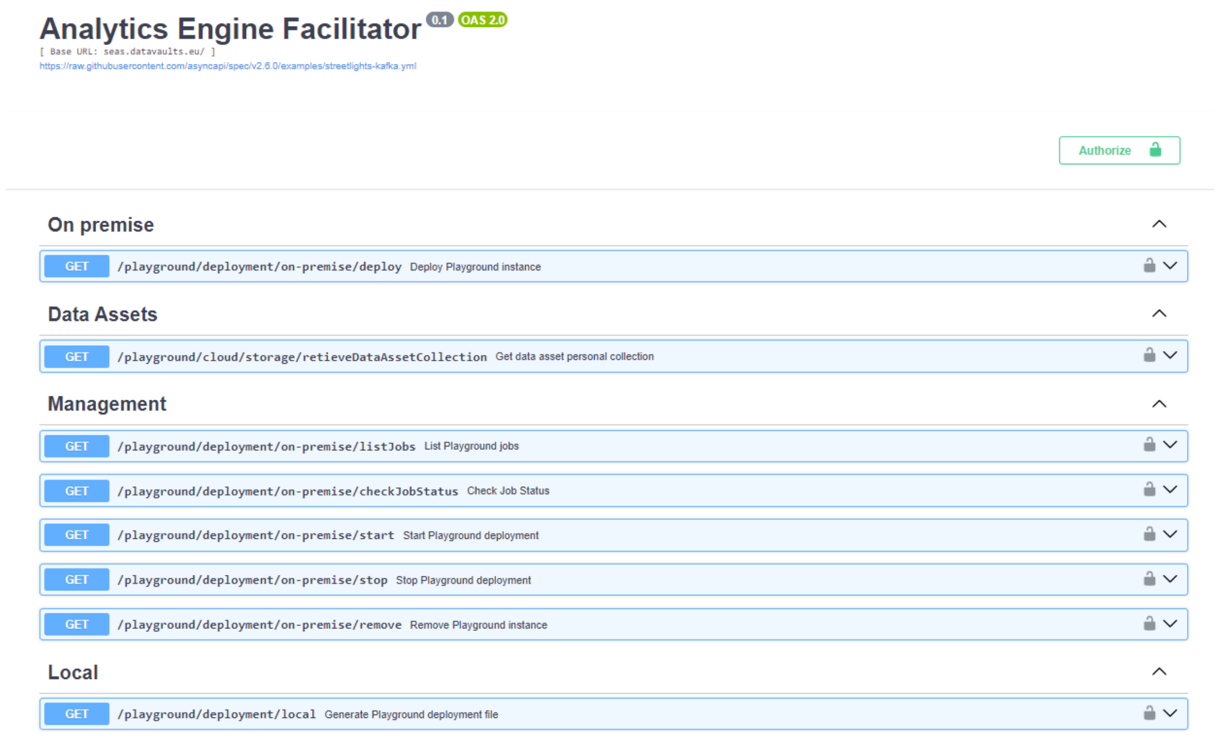
5.4.1.1 Components providing Input to Analytics Engine

Input Required	Component providing the Input	Communication Method
List of Dataset paths	Factory Data Storage	API
Market Insights Execution Query	PISTIS Market Insights	API

5.4.1.2 to which Analytics Engine provides input

Output Served	Component ingesting the output	Communication Method
Market Insights Analysis Results	PISTIS Market Insights	API

5.4.2 API Calls Descriptions



Analytics Engine Facilitator 0.1 OAS 2.0

[Base URL: seas.datavaults.eu/]

https://raw.githubusercontent.com/asyncapi/spec/v2.6.0/examples/streetlights-kafka.yml

On premise

GET `/playground/deployment/on-premise/deploy` Deploy Playground instance

Data Assets

GET `/playground/cloud/storage/retieveDataAssetCollection` Get data asset personal collection

Management

GET `/playground/deployment/on-premise/listJobs` List Playground jobs

GET `/playground/deployment/on-premise/checkJobStatus` Check Job Status

GET `/playground/deployment/on-premise/start` Start Playground deployment

GET `/playground/deployment/on-premise/stop` Stop Playground deployment

GET `/playground/deployment/on-premise/remove` Remove Playground instance

Local

GET `/playground/deployment/local` Generate Playground deployment file

5.4.3 Sequence Diagrams

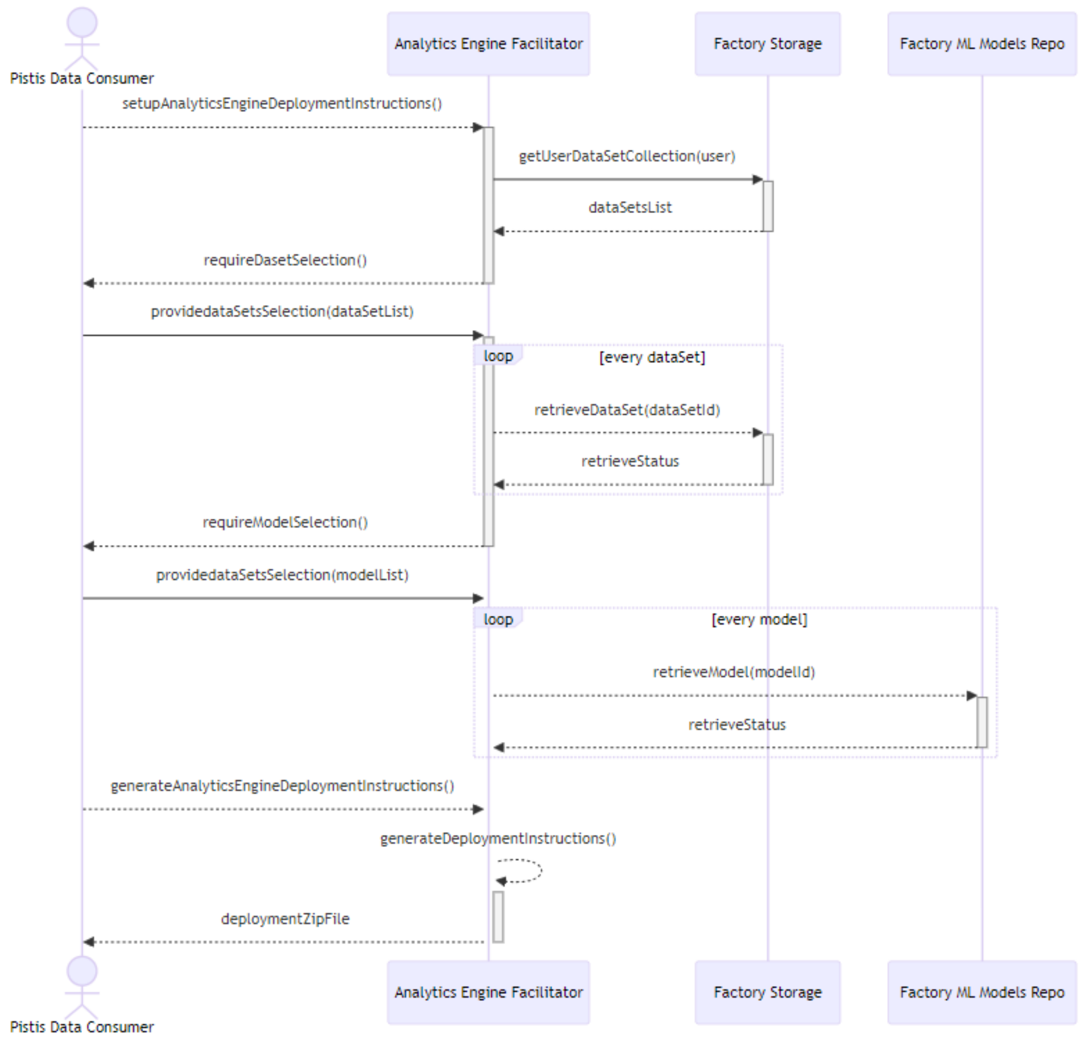

Pistis Data Consumer
Analytics Engine Facilitator
Factory Storage
Factory ML Models Repo
setupAnalyticsEngineDeploymentInstructions()
getUserDataSetCollection(user)
dataSetsList
requireDasetSelection()
providedataSetsSelection(dataSetList)
loop
[every dataSet]
retrieveDataSet(dataSetId)
retrieveStatus
requireModelSelection()
providedataSetsSelection(modelList)
loop
[every model]
retrieveModel(modelId)
retrieveStatus
generateAnalyticsEngineDeploymentInstructions()
generateDeploymentInstructions()
deploymentZipFile

Figure 11: Analytics Engine Facilitator Sequence Diagram

5.5 Data Enrichment

The data enrichment component in PISTIS is included in the Data Ingestion and Transformation module, which is situated in the premises of an organisation. Along with the data transformation component, this component is responsible to transform and enrich any available data assets to increase its value for trading. Data enrichment refines and enhances datasets to add more value and utilisation to the existing data. Typically, data enrichment refers to data harmonisation using additional data sources. It combines information from several data sources into a standardised format for further data analysis. The different source of data could be in different file formats, naming conventions and distinct data sources. The main steps included in data harmonisation are data cleaning, appending, sorting and aggregating.

Enrichment of data can extend beyond data harmonisation using data cleaning and aggregating. Metadata of the data can be involved in this process to extend the actual data. Metadata enrichment is about controlling the onboarding of new data into a standardised data landscape by using domain specific vocabularies. Metadata available in RDF format can be semantically enriched to align with the available semantic data models.

5.5.1 Relation to Other Components

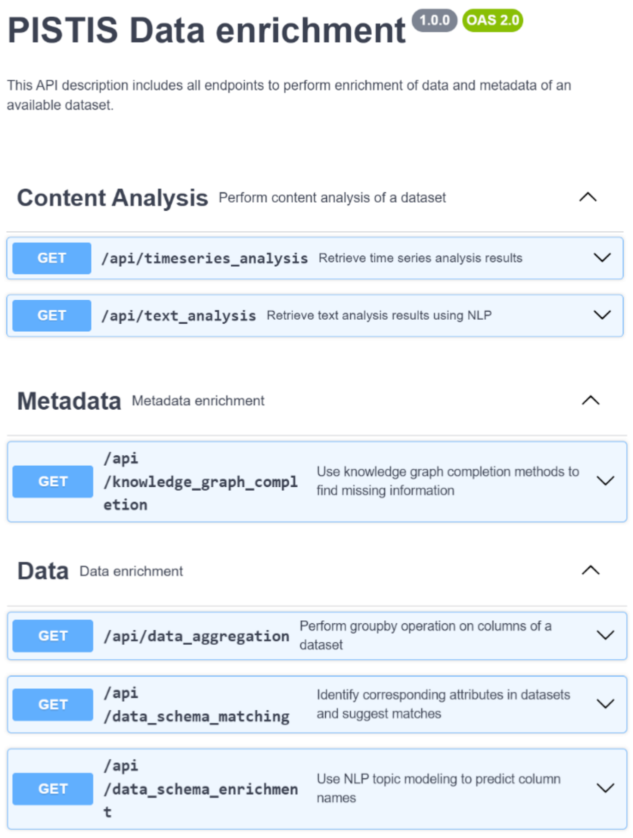
5.5.1.1 Components providing Input to Data Enrichment

Input Required	Component providing the Input	Communication Method
Dataset	Job Configurator	API
Metadata of a dataset	Factory Data Catalogue	API
Statistical analysis results	Analytics Engine	API

5.5.1.2 Components to which Data enrichment provides input

Output Served	Component ingesting the output	Communication Method
Dataset	Job Configurator	API
Metadata	Factory Data Catalogue	API

5.5.2 API Calls Descriptions


PISTIS Data enrichment 1.0.0 OAS 2.0

This API description includes all endpoints to perform enrichment of data and metadata of an available dataset.

Content Analysis Perform content analysis of a dataset

GET /api/timeseries_analysis Retrieve time series analysis results

GET /api/text_analysis Retrieve text analysis results using NLP

Metadata Metadata enrichment

GET /api/knowledge_graph_completion Use knowledge graph completion methods to find missing information

Data Data enrichment

GET /api/data_aggregation Perform groupby operation on columns of a dataset

GET /api/data_schema_matching Identify corresponding attributes in datasets and suggest matches

GET /api/data_schema_enrichment Use NLP topic modeling to predict column names

5.5.3 Sequence Diagrams

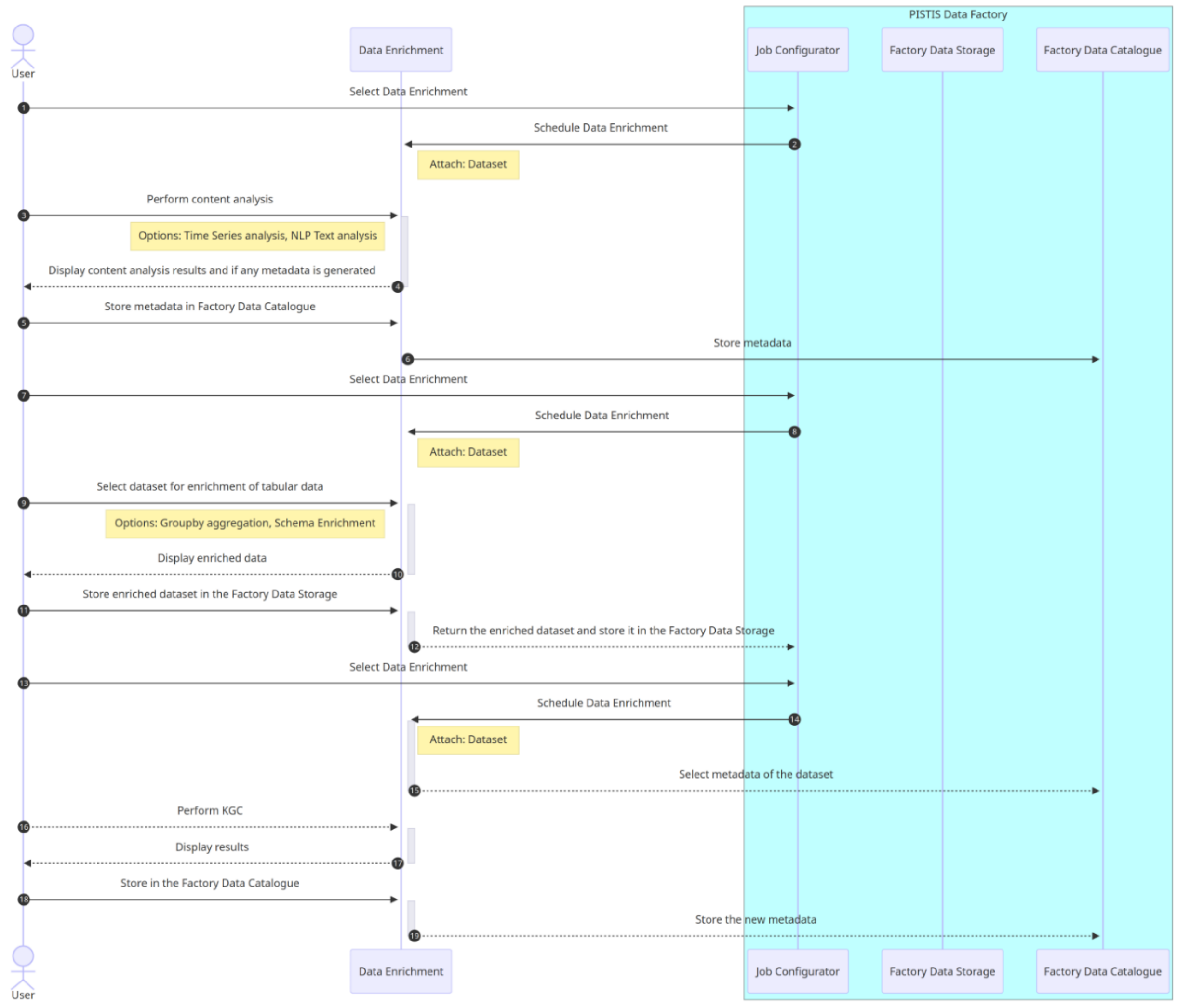

PISTIS Data Factory
User
Data Enrichment
Job Configurator
Factory Data Storage
Factory Data Catalogue
Select Data Enrichment
Schedule Data Enrichment
Attach: Dataset
Perform content analysis
Options: Time Series analysis, NLP Text analysis
Display content analysis results and if any metadata is generated
Store metadata in Factory Data Catalogue
Store metadata
Select Data Enrichment
Schedule Data Enrichment
Attach: Dataset
Select dataset for enrichment of tabular data
Options: Groupby aggregation, Schema Enrichment
Display enriched data
Store enriched dataset in the Factory Data Storage
Return the enriched dataset and store it in the Factory Data Storage
Select Data Enrichment
Schedule Data Enrichment
Attach: Dataset
Select metadata of the dataset
Perform KGC
Display results
Store in the Factory Data Catalogue
Store the new metadata

Figure 12: Data Enrichment Sequence Diagram

5.6 Data Quality Assessment

The Data & Metadata Quality Assessment component ensures the quality and consistency of data and metadata within the PISTIS system.

It provides two main functionalities:

- **Metadata Assessment:** this module checks and validates metadata against the predefined Metadata model and returns the validation result together with a score of the result. It identifies missing metadata, validates data types and formats, and ensures adherence to data standards.
- **Data Assessment:** this module checks and validates structured data against information provided in the metadata and returns the validation result together with

a score of the result. It checks for data consistency, adherence to data quality rules, and identifies potential errors or anomalies. The validation process ensures that data is reliable, accurate, and usable for downstream applications. For this purpose, it uses the “Great Expectations” Python library[2].

The Data & Metadata Quality Assessment component provides APIs for both metadata and data validation, allowing integration with various data management and processing tools. It also supports on-demand and scheduled validation runs, enabling proactive data quality monitoring. User-defined validation rules can be incorporated to address specific data quality requirements.

5.6.1 Relation to Other Components

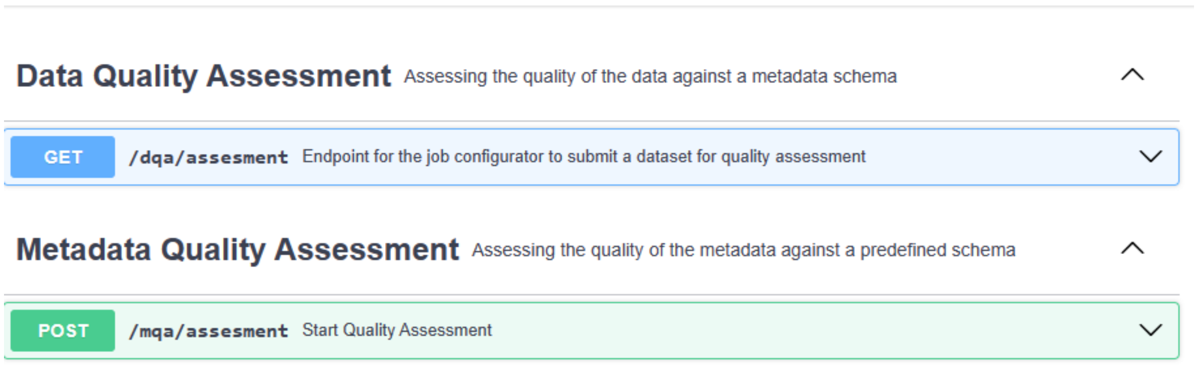
5.6.1.1 *Components providing Input to Data Quality Assessment*

Input Required	Component providing the Input	Communication Method
Metadata	Job Configurator	REST API
Metadata Model	Job Configurator	REST API
Data	Job Configurator	REST API

5.6.1.2 *Components to which Data Quality Assessment provides input*

Output Served	Component ingesting the output	Communication Method
Metadata validation result	Job Configurator	REST API
Data validation Result	Job Configurator	REST API

5.6.2 API Calls Descriptions


Data Quality Assessment Assessing the quality of the data against a metadata schema

GET /dqa/assesment Endpoint for the job configurator to submit a dataset for quality assessment

Metadata Quality Assessment Assessing the quality of the metadata against a predefined schema

POST /mqa/assesment Start Quality Assessment

[2] https://greatexpectations.io/ - Accessed 29/12/2023

5.6.3 Sequence Diagrams

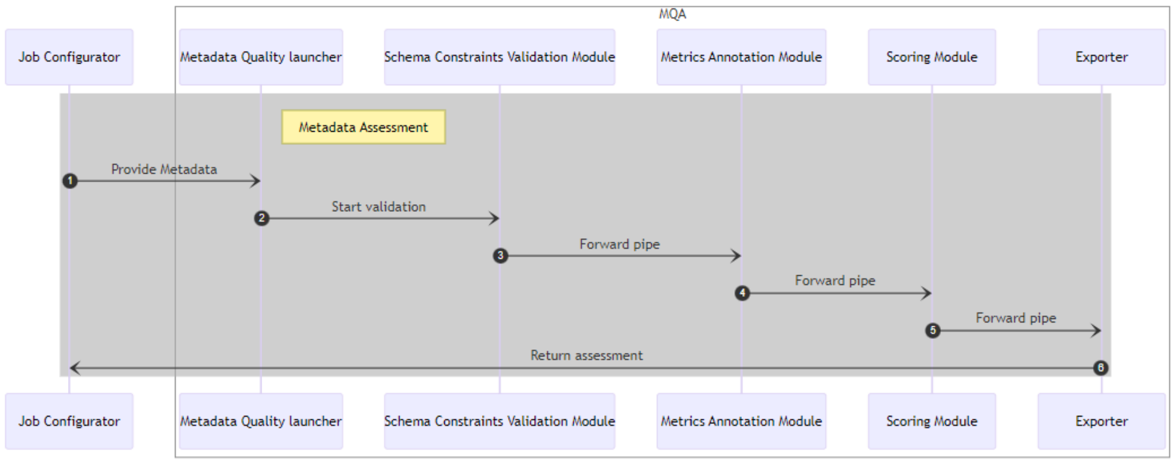


Figure 13: Metadata Quality Assessment Sequence Diagram

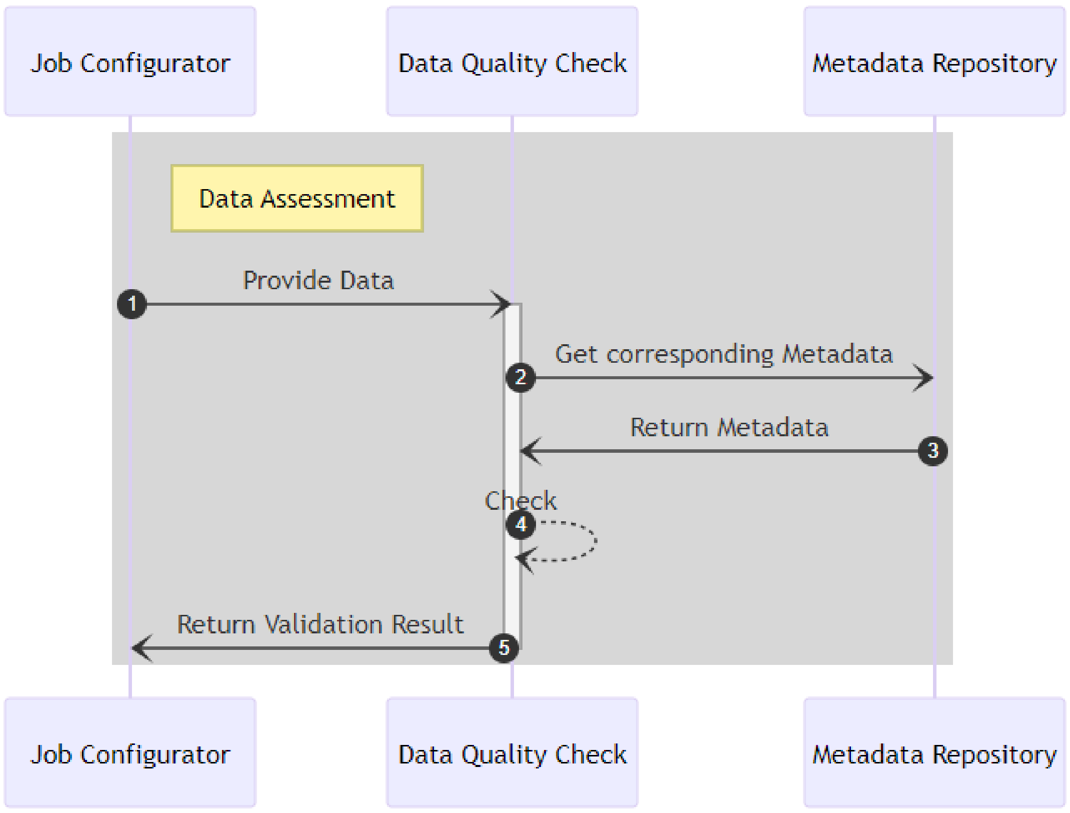


Figure 14: Data Quality Assessment Sequence Diagram

5.7 Data Insights Generator

The Data Insights generator is a component that provides information about the structure and data types of a given dataset to ease the understanding of a dataset for the final user.

The component is expected to receive a given dataset and map it to a python Pandas DataFrame. From that input dataset, a report on the different fields of the dataset is expected to be provided, including some information such as the data type of each field, number of missing elements, different values in categorial values, some statistical analytics on numerical data, etc.

5.7.1 Relation to Other Components

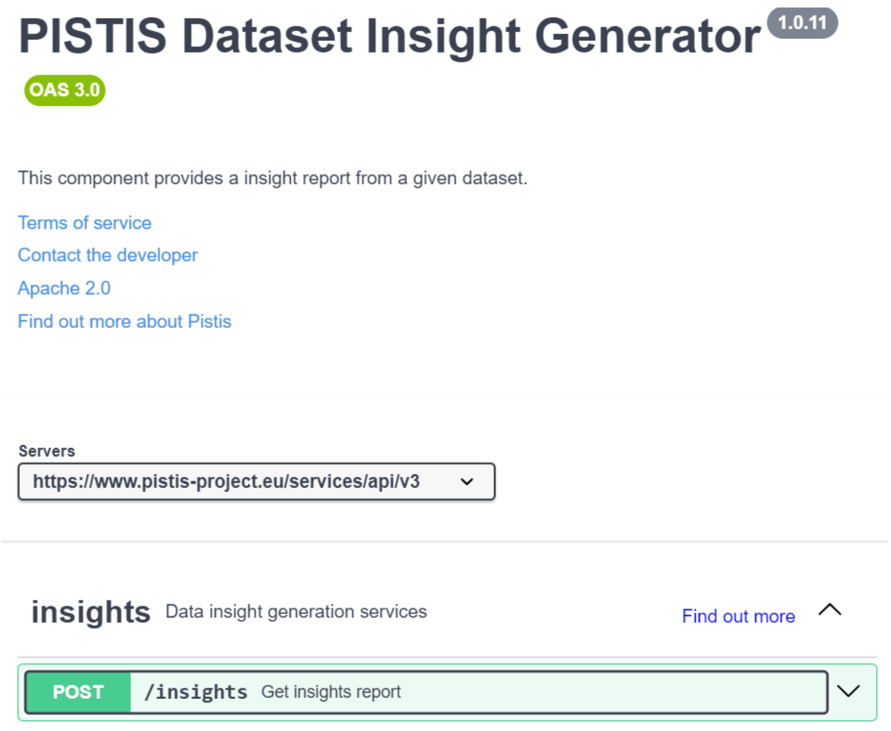
5.7.1.1 Components providing Input to Data Insight Generator

Input Required	Component providing the Input	Communication Method
Dataset	Job configurator	API

5.7.1.2 Components to which Data Insight Generator provides input

Output Served	Component ingesting the output	Communication Method
Insights report	**Job configurator**	API

5.7.2 API Calls Descriptions


PISTIS Dataset Insight Generator 1.0.11

OAS 3.0

This component provides a insight report from a given dataset.

Terms of service
Contact the developer
Apache 2.0
Find out more about Pistis

Servers
https://www.pistis-project.eu/services/api/v3

insights Data insight generation services
Find out more

POST /insights Get insights report

5.7.3 Sequence Diagrams

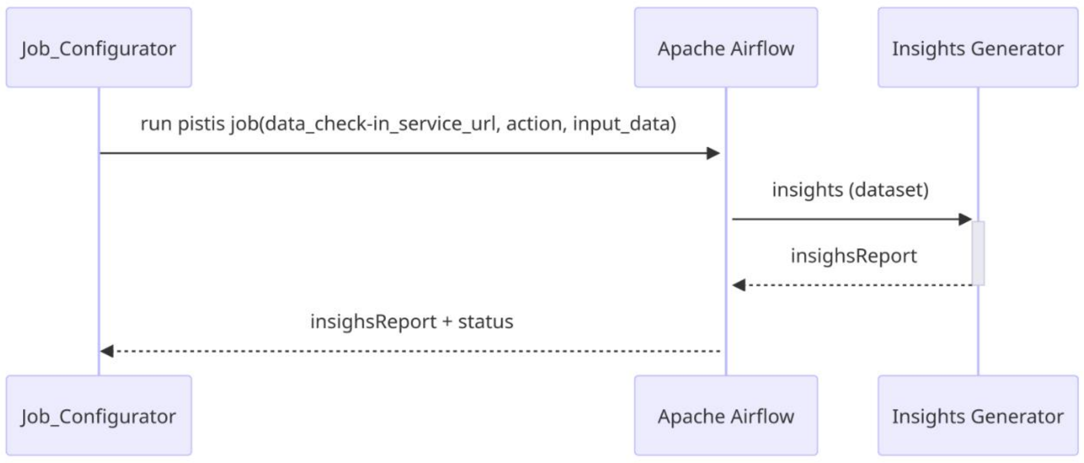


Figure 15: Data Insight Generator Sequence Diagram

6 Data & Metadata Storage Bundle

The Data & Metadata Storage bundle is delivering a catalogue for the data that are made available by each organisation in their own PISTIS Data Factory environment. Moreover, it also concerns those made available as "published" datasets over the whole ecosystem, alongside with the appropriate data storage facilities to hold the data.

This bundle consists of the following components:

- Data Catalogues
- Factory Data Storage
- Indexing Service

These are presented in the following sub-sections.

6.1 Data Catalogues

Under "data catalogues" we refer to the components that offer catalogue features for the data in PISTIS. They constitute the essential components to manage the offerings of data assets and are the following:

- **Factory Data Catalogue**
- **PISTIS Data Catalogue**

The **Factory Data Catalogue** runs within the premises of the data provider and serves as the access point to the organisation's data assets. It provides the means to make the metadata and data available. Each organisation is responsible for maintaining its own catalogue and incorporating their own (meta)data.

The **PISTIS Data Catalogue** is a centralised service within the PISTIS Cloud Platform and aggregates the metadata from all Factory Data Catalogues. It constitutes the central marketplace of PISTIS by allowing to browse all available data assets. The visibility of the data assets is determined by the respective data access policies.

6.1.1 Relation to Other Components

6.1.1.1 Components providing Input to Factory Data Catalogue

Input Required	Component providing the Input	Communication Method
Basic Metadata	Data Check-in	Web Application / API
Metadata about Transformation	Data Transformation	API
Metadata about Enrichment	Data Enrichment	API
Quality Assessment of Data and Metadata	Data Quality Assessment	API
Provenance Metadata	Lineage Tracker	API

6.1.1.2 Components to which Factory Data Catalogue provides input

Output Served	Component ingesting the output	Communication Method
Synced metadata	PISTIS Data Catalogue	API
Entire Metadata[3]	PISTIS Data Factory Connector	API

6.1.1.3 Components providing Input to PISTIS Data Catalogue

Input Required	Component providing the Input	Communication Method
Entire Metadata	Factory Data Catalogue	API
Identity of the User	PISTIS IAM	HTTP / OIDC

6.1.1.4 Components to which PISTIS Data Catalogue provides input

Output Served	Component ingesting the output	Communication Method
Entire Metadata	PISTIS Market Insights	API
Metadata matching a user query	Distributed Query Engine	API

6.1.1.5 Components to which PISTIS Data Catalogue provides input

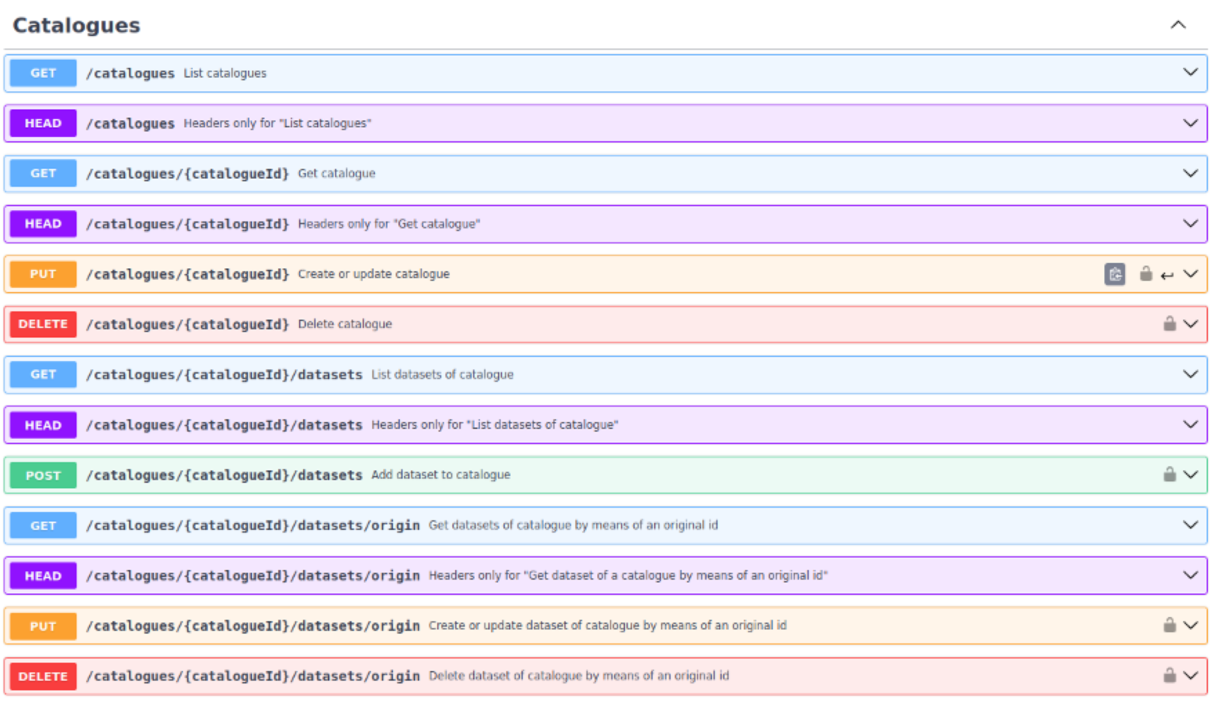
Output Served	**Component ingesting the output**	**Communication Method**
Entire Metadata	PISTIS Market Insights	API
Metadata matching a user query	Distributed Query Engine	API

6.1.2 API CALLS DESCRIPTIONS

These components will have the same API and will use the same software stack as a basis (to be defined in WP2). Conceptionally each entity of the API will represent a different “thing” for the different components.

- Factory Data Catalogue
  - Each Factory will most likely only have one catalogue
  - A Dataset is the metadata about a data asset
  - A Distribution gives details about the actual access to the Factory Store
- PISTIS Data Catalogue
  - A Catalogue represent a single data provider
  - A Dataset is the metadata about a data asset
  - A Distribution gives details about the actual access (e.g. via a Connector)

[3] “Entire Metadata” refers to a complete metadata set for data asset, including the provided metadata and added metadata, e.g., through enrichment and data linage tracker.



Catalogues

GET /catalogues List catalogues

HEAD /catalogues Headers only for "List catalogues"

GET /catalogues/{catalogueId} Get catalogue

HEAD /catalogues/{catalogueId} Headers only for "Get catalogue"

PUT /catalogues/{catalogueId} Create or update catalogue

DELETE /catalogues/{catalogueId} Delete catalogue

GET /catalogues/{catalogueId}/datasets List datasets of catalogue

HEAD /catalogues/{catalogueId}/datasets Headers only for "List datasets of catalogue"

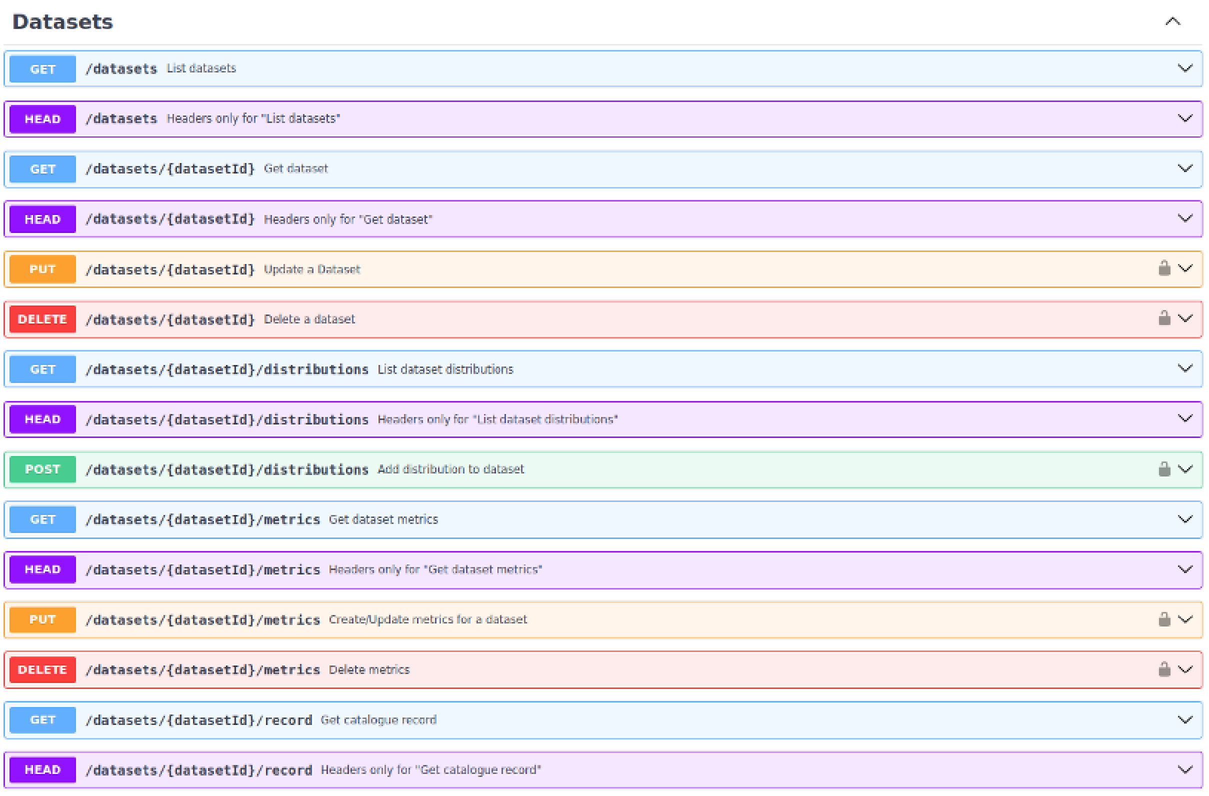
POST /catalogues/{catalogueId}/datasets Add dataset to catalogue

GET /catalogues/{catalogueId}/datasets/origin Get datasets of catalogue by means of an original id

HEAD /catalogues/{catalogueId}/datasets/origin Headers only for "Get dataset of a catalogue by means of an original id"

PUT /catalogues/{catalogueId}/datasets/origin Create or update dataset of catalogue by means of an original id

DELETE /catalogues/{catalogueId}/datasets/origin Delete dataset of catalogue by means of an original id



Datasets

GET /datasets List datasets

HEAD /datasets Headers only for "List datasets"

GET /datasets/{datasetId} Get dataset

HEAD /datasets/{datasetId} Headers only for "Get dataset"

PUT /datasets/{datasetId} Update a Dataset

DELETE /datasets/{datasetId} Delete a dataset

GET /datasets/{datasetId}/distributions List dataset distributions

HEAD /datasets/{datasetId}/distributions Headers only for "List dataset distributions"

POST /datasets/{datasetId}/distributions Add distribution to dataset

GET /datasets/{datasetId}/metrics Get dataset metrics

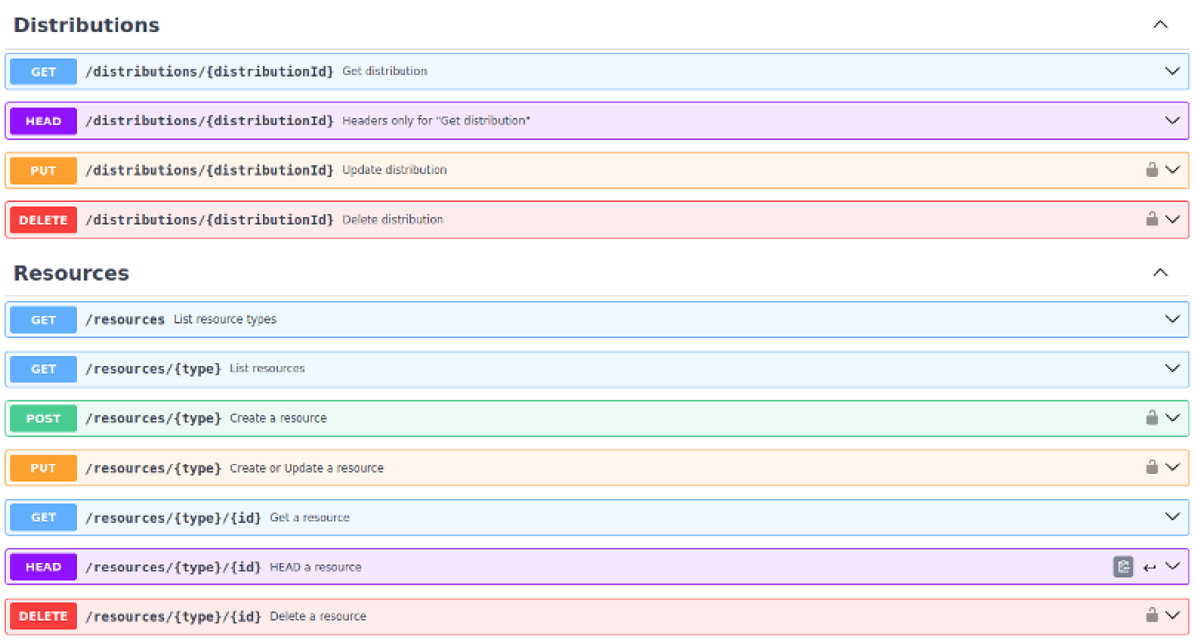
HEAD /datasets/{datasetId}/metrics Headers only for "Get dataset metrics"

PUT /datasets/{datasetId}/metrics Create/Update metrics for a dataset

DELETE /datasets/{datasetId}/metrics Delete metrics

GET /datasets/{datasetId}/record Get catalogue record

HEAD /datasets/{datasetId}/record Headers only for "Get catalogue record"



Distributions

GET /distributions/{distributionId} Get distribution

HEAD /distributions/{distributionId} Headers only for "Get distribution"

PUT /distributions/{distributionId} Update distribution

DELETE /distributions/{distributionId} Delete distribution

Resources

GET /resources List resource types

GET /resources/{type} List resources

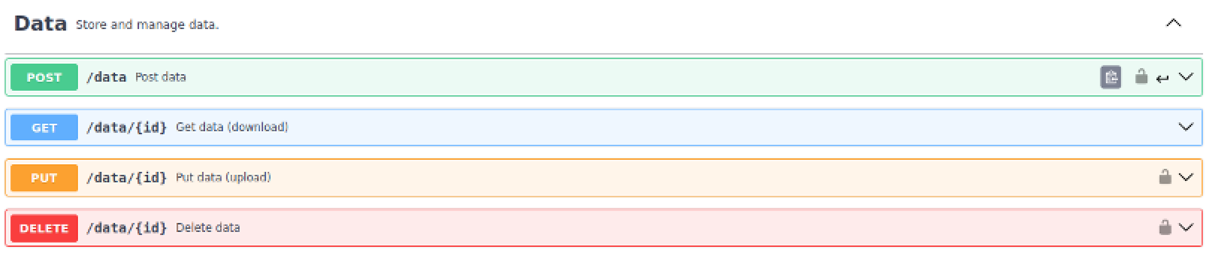
POST /resources/{type} Create a resource

PUT /resources/{type} Create or Update a resource

GET /resources/{type}/{id} Get a resource

HEAD /resources/{type}/{id} HEAD a resource

DELETE /resources/{type}/{id} Delete a resource



Data Store and manage data.

POST /data Post data

GET /data/{id} Get data (download)

PUT /data/{id} Put data (upload)

DELETE /data/{id} Delete data

6.1.3 SEQUENCE DIAGRAMS

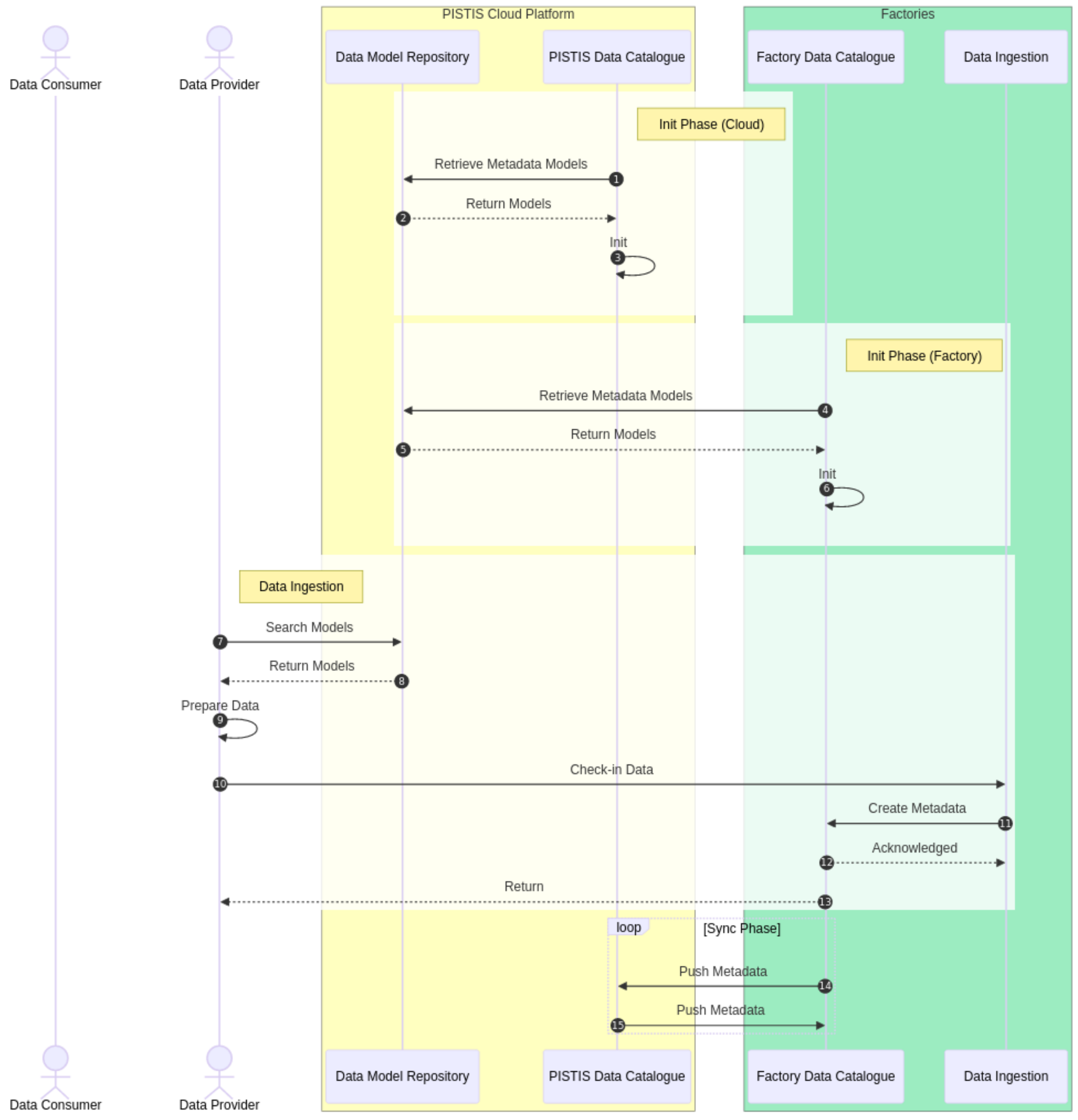


Figure 16: Data Catalogues Sequence Diagram

6.2 FACTORY DATA STORAGE

The Factory Data Storage is a database that is hosted locally by the Data Factory environment which is on the premises of an organisation that is member of the PISTIS ecosystem. Data from each organisation is ingested into the Data Storage by the Data ingestion module. This component comprises of two relational databases to store the datasets in the form of relational tables and files. Access to these databases is provided through a REST API.

The datasets are stored in two separate databases based on their format. If a dataset is in the form of a relational table, with a data schema, they are stored as a table using SQL queries. Each table is assigned a unique identifier that is a UUID, which is used to query the rows and columns of a table. Multiple versions of a table can also be stored in this database. Each version is assigned a version ID and using this version ID and UUID, rows and columns of different versions of a table can be queried.

If a dataset is available in the form of a file (csv, xml), they are stored in another database using a UUID. The database stores the file, name of the file, UUID assigned to it and version ID of the file. Different versions of a file can be downloaded using its UUID and version ID.

The main functionalities of the Data Storage are:

- Storage for tables and files on a relational database
- Access to the database using a REST API
- API endpoints to store, retrieve and delete datasets using a UUID as identifier
- API endpoints to store and retrieve different versions of a dataset using a version ID and UUID
- Possibility to perform SQL queries on tabular data

6.2.1 RELATION TO OTHER COMPONENTS

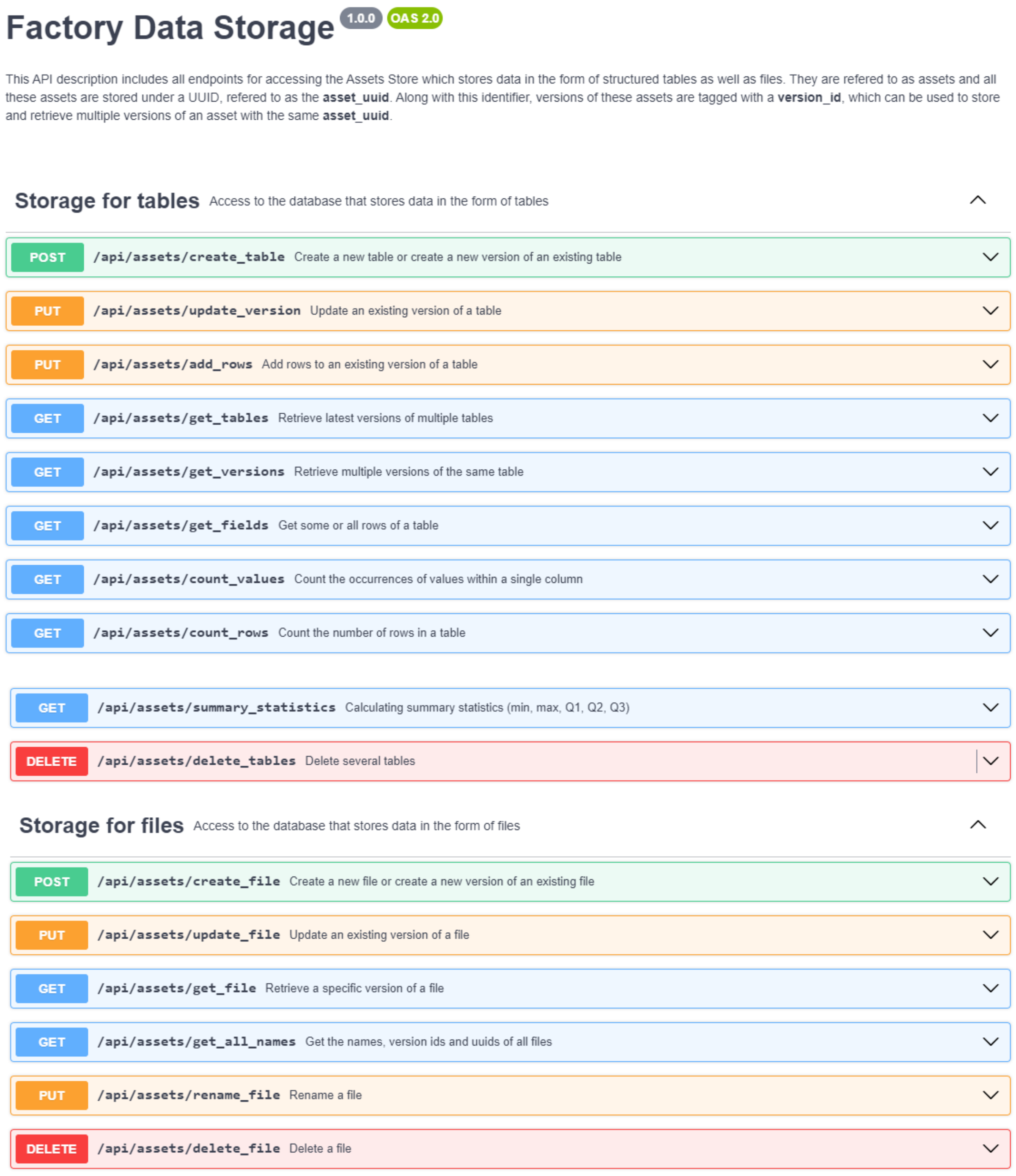
6.2.1.1 Components providing Input to Factory Data Storage

Input Required	Component providing the Input	Communication Method
Data, Data model and some initial metadata (such as name of the dataset)	Job Configurator	API
Data, Data model and some initial metadata (such as name of the dataset)	PISTIS Data Factory Connector	API
Dataset identifier (UUID)	Distributed Query Engine	API
Transaction ID to retrieve encrypted keyword	Searchable Encryption	API

6.2.1.2 Components to which Factory Data Storage provides input

Output Served	Component ingesting the output	Communication Method
Dataset UUID, version ID, user info	Lineage Tracker	API
Metadata (UUID)	Factory Data Catalogue	API
Full Dataset and metadata	PISTIS Data Factory Connector	API
Dataset identifier (UUID), version ID, Dataset	Distributed Query Engine	API
Encrypted Keywords & Transaction IDs	Searchable Encryption	API

6.2.2 API Calls Descriptions



Factory Data Storage 1.0.0 OAS 2.0

This API description includes all endpoints for accessing the Assets Store which stores data in the form of structured tables as well as files. They are refered to as assets and all these assets are stored under a UUID, refered to as the **asset_uuid**. Along with this identifier, versions of these assets are tagged with a **version_id**, which can be used to store and retrieve multiple versions of an asset with the same **asset_uuid**.

Storage for tables Access to the database that stores data in the form of tables

POST `/api/assets/create_table` Create a new table or create a new version of an existing table

PUT `/api/assets/update_version` Update an existing version of a table

PUT `/api/assets/add_rows` Add rows to an existing version of a table

GET `/api/assets/get_tables` Retrieve latest versions of multiple tables

GET `/api/assets/get_versions` Retrieve multiple versions of the same table

GET `/api/assets/get_fields` Get some or all rows of a table

GET `/api/assets/count_values` Count the occurrences of values within a single column

GET `/api/assets/count_rows` Count the number of rows in a table

GET `/api/assets/summary_statistics` Calculating summary statistics (min, max, Q1, Q2, Q3)

DELETE `/api/assets/delete_tables` Delete several tables

Storage for files Access to the database that stores data in the form of files

POST `/api/assets/create_file` Create a new file or create a new version of an existing file

PUT `/api/assets/update_file` Update an existing version of a file

GET `/api/assets/get_file` Retrieve a specific version of a file

GET `/api/assets/get_all_names` Get the names, version ids and uuids of all files

PUT `/api/assets/rename_file` Rename a file

DELETE `/api/assets/delete_file` Delete a file

6.2.3 SEQUENCE DIAGRAMS

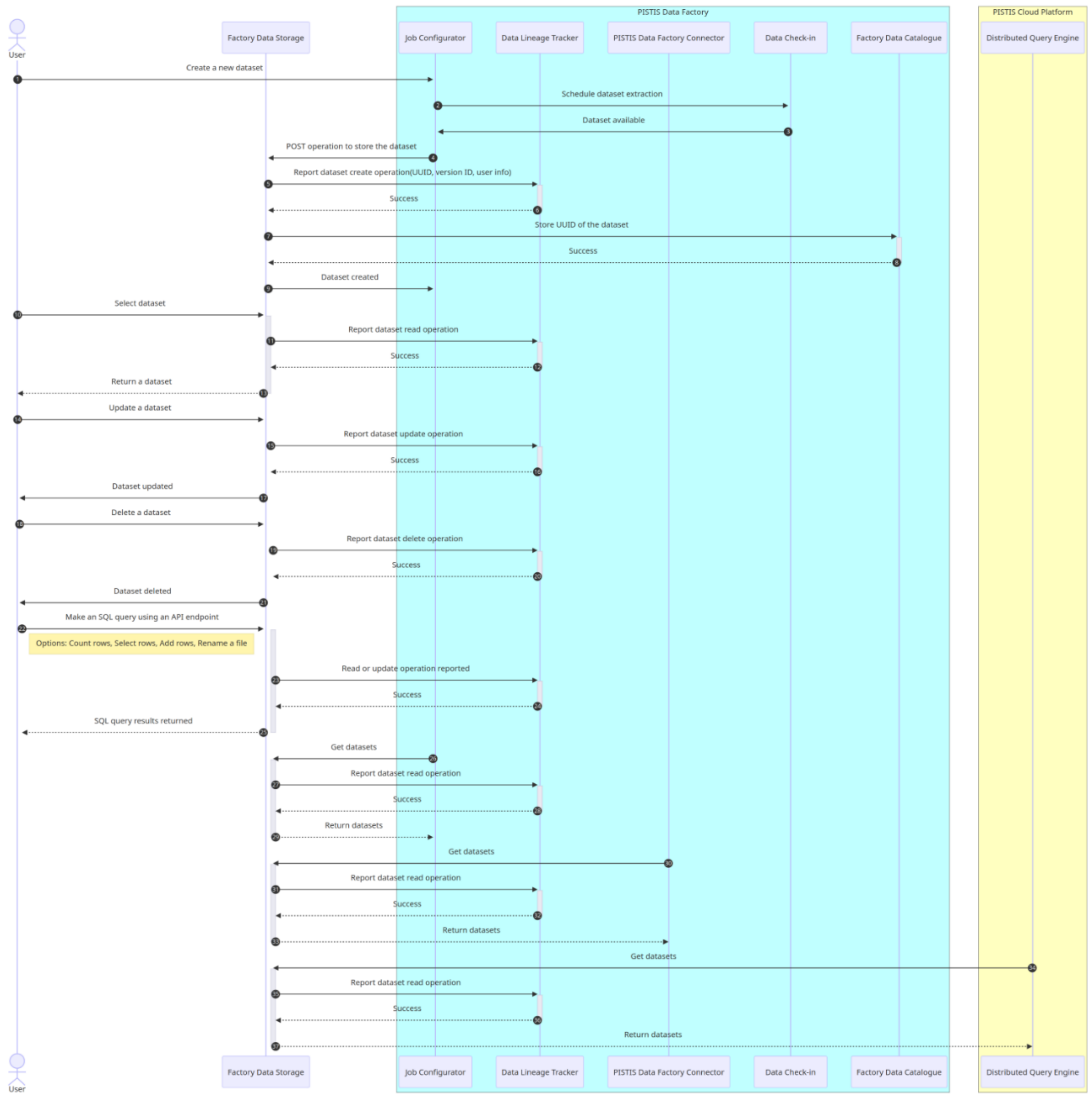

PISTIS Data Factory
PISTIS Cloud Platform
User
Factory Data Storage
Job Configurator
Data Lineage Tracker
PISTIS Data Factory Connector
Data Check-in
Factory Data Catalogue
Distributed Query Engine
Create a new dataset
Schedule dataset extraction
Dataset available
POST operation to store the dataset
Report dataset create operation(UUID, version ID, user info)
Success
Store UUID of the dataset
Success
Dataset created
Select dataset
Report dataset read operation
Success
Return a dataset
Update a dataset
Report dataset update operation
Success
Dataset updated
Delete a dataset
Report dataset delete operation
Success
Dataset deleted
Make an SQL query using an API endpoint
Options: Count rows, Select rows, Add rows, Rename a file
Read or update operation reported
Success
SQL query results returned
Get datasets
Report dataset read operation
Success
Return datasets
Get datasets
Report dataset read operation
Success
Return datasets
Get datasets
Report dataset read operation
Success
Return datasets

Figure 17: Factory Data Storage Sequence Diagram

7 Data Discovery Bundle

The Data Discovery bundle provides services for searching and discovering the available data assets that might be of interest for a Data Consumer.

This bundle consists of the following components:

- Distributed Query Engine
- Matchmaking Services

These are presented in the following sub-sections.

7.1 Distributed Query Engine

The main purpose of this component is to query directly the unstructured or semi-structured data to discover datasets that cannot be retrieved by querying their metadata on the Distributed Data Catalogue.

However, the volume of the data stored in the Data Factories does not allow extensive search approaches to be used. Therefore, Locality Sensitive Hashing techniques will be employed to quickly obtain a list of matches. Subsequently, the list of potential matches yielded by the LSH methods will be further evaluated and combined with those returned by the Distributed Data Catalogue.

Finally, the merged list will be cross-checked with the Policy Engine that will be part of the Keycloak to decide if the users have access rights to the results. Subsequently, this list will be fed to a pretrained ML-model that will re-rank it to give prominence to the most relevant matches.

7.1.1 Relation to Other Components

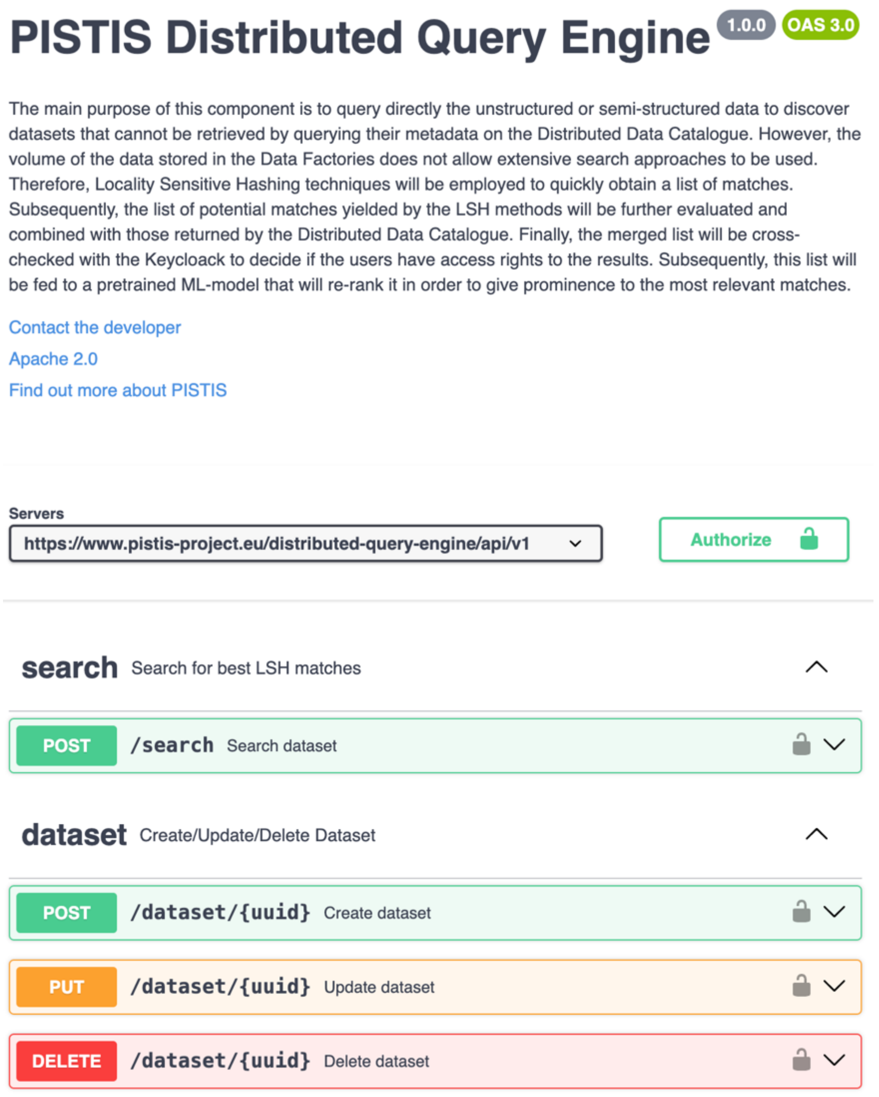
7.1.1.1 Components providing Input to Distributed Query Engine

Input Required	Component providing the Input	Communication Method
Results for queries on metadata	Distributed Data Catalogue	API
User access rights on Dataset	Keycloak	API
Notification for data altering action (Insert/Update/Delete) on Dataset	Data Storage	API
Dataset’s raw data for indexing	Data Storage	API

7.1.1.2 Components to which Distributed Query Engine provides input

Output Served	Component ingesting the output	Communication Method
N/A		

7.1.2 API Calls Descriptions



PISTIS Distributed Query Engine 1.0.0 OAS 3.0

The main purpose of this component is to query directly the unstructured or semi-structured data to discover datasets that cannot be retrieved by querying their metadata on the Distributed Data Catalogue. However, the volume of the data stored in the Data Factories does not allow extensive search approaches to be used. Therefore, Locality Sensitive Hashing techniques will be employed to quickly obtain a list of matches. Subsequently, the list of potential matches yielded by the LSH methods will be further evaluated and combined with those returned by the Distributed Data Catalogue. Finally, the merged list will be cross-checked with the Keycloack to decide if the users have access rights to the results. Subsequently, this list will be fed to a pretrained ML-model that will re-rank it in order to give prominence to the most relevant matches.

Contact the developer
Apache 2.0
Find out more about PISTIS

Servers
https://www.pistis-project.eu/distributed-query-engine/api/v1

Authorize

search Search for best LSH matches

POST `/search` Search dataset

dataset Create/Update/Delete Dataset

POST `/dataset/{uuid}` Create dataset

PUT `/dataset/{uuid}` Update dataset

DELETE `/dataset/{uuid}` Delete dataset

7.1.3 Sequence Diagrams

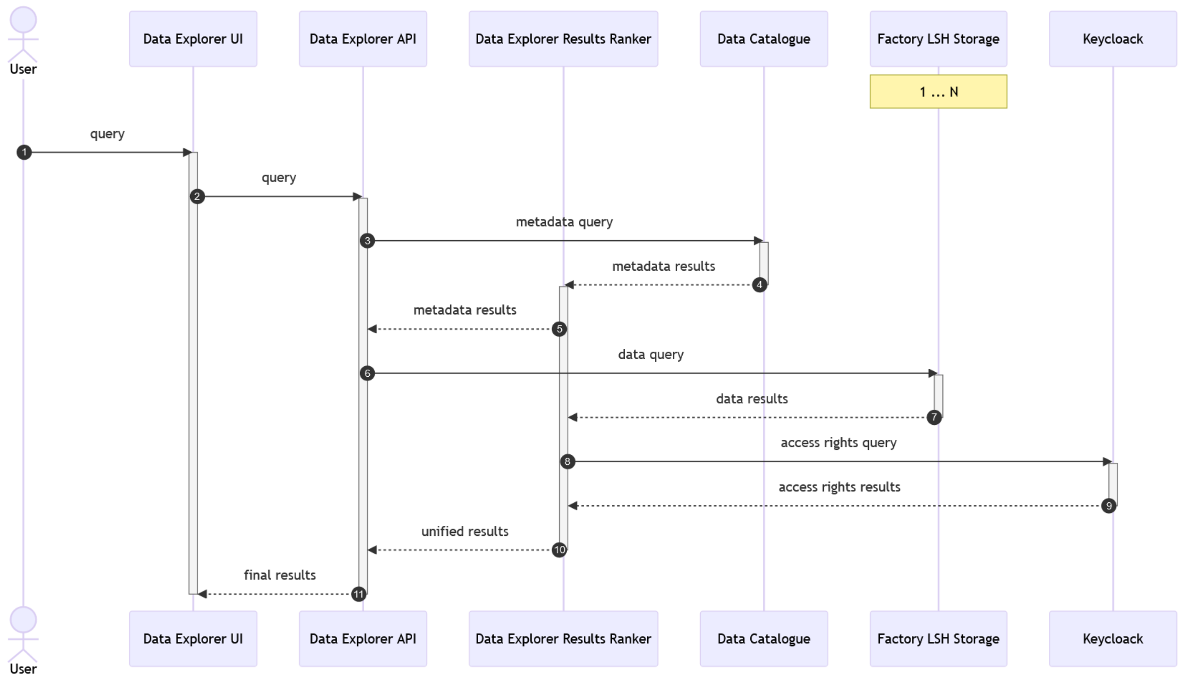


Figure 18: Querying Data Explorer Sequence Diagram

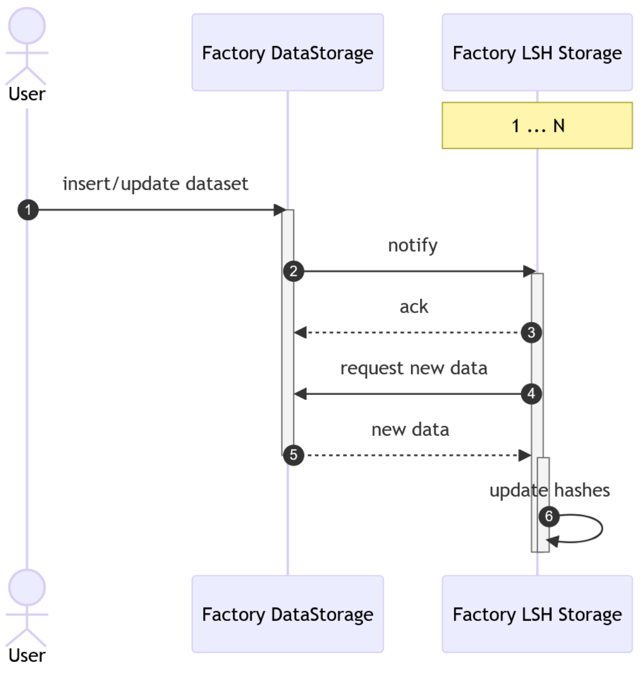


Figure 19: Insert/Update Dataset in LSH Storage Sequence Diagram

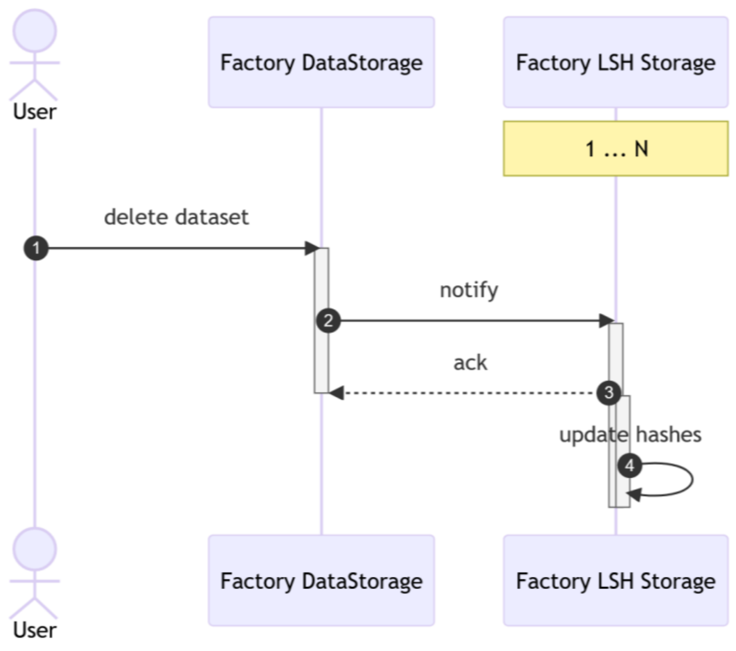


Figure 20: Delete Dataset from LSH Storage Sequence Diagram

7.2 Matchmaking Services

Matchmaking services are employed to link, as proactively as possible, Data Providers to Data Consumers, based on the latter's interest in data assets (e.g., application domain), as well as complementary data assets, which can result in an increase of data value. Think of it like a matchmaking problem, where the 'items' to be matched are datasets.

The main functionalities of the Matchmaking Service are:

1. Recommends data assets to Consumers based on their previous purchases and interactions with the Platform.
2. Connects Consumers with Owners of data assets, based on the platform interactions and interests of the former and the properties of the data assets of the latter.
3. Provides interpretability for the user-asset match.

The component will achieve these functionalities by implementing state-of-the-art recommendation algorithms: k-Nearest Neighbour (kNN), collaborative filtering (CF), deep neural networks (DNN) or graph neural networks (GNN). The minimum required data are user-assets interactions, where the concept of "interaction" can refer to "user views an asset", "user enquires about an asset", "user makes a query with certain parameters", "user purchases an asset". The choice of the algorithm will be heavily influenced by the amount of interaction data available for training, with an initial preference for kNN and probably CF, as DNN and GNN-based recommenders require large amounts to train.

7.2.1 Relation to Other Components

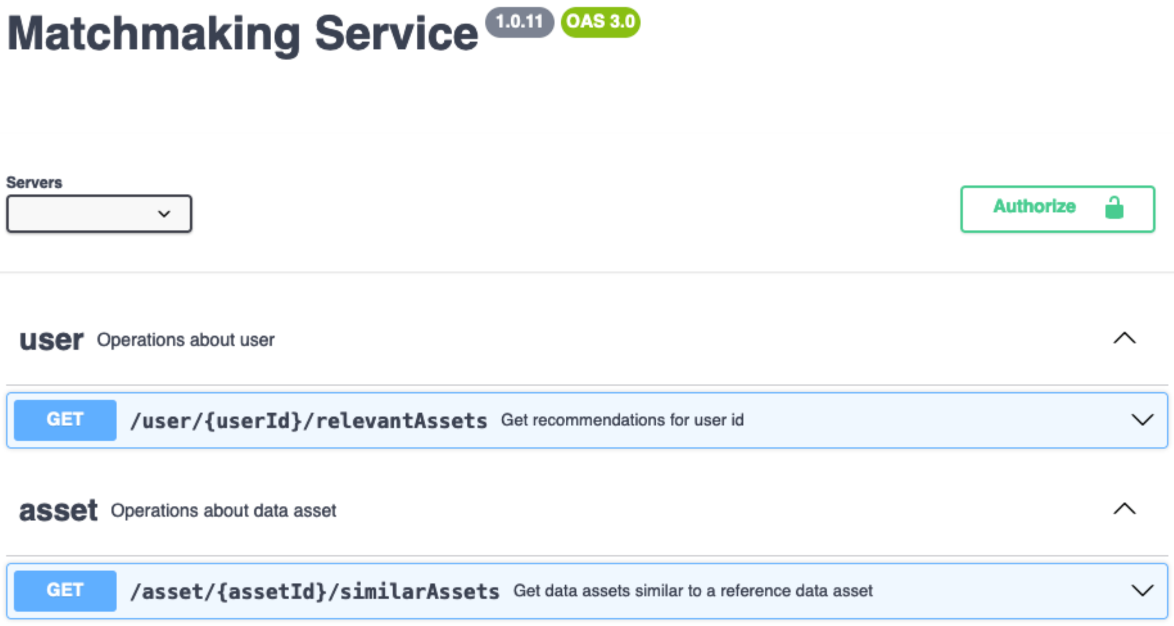
7.2.1.1 Components providing Input to the Matchmaking Service

Input Required	Component providing the Input	Communication Method
Data assets and metadata	PISTIS Data Catalogue	API
User purchases an asset (only Consumer ID and Asset ID)	Smart Contract Execution Engine	API
User views a dataset (only Consumer ID and Asset ID)	UI logs	API
User enquires about a dataset (only Consumer ID and Asset ID)	Distributed Query Engine	API

7.2.1.2 Components to which the Matchmaking Service provides input

Output Served	Component ingesting the output	Communication Method
User-asset recommendations	Distributed Query Engine (frontend)	API

7.2.2 API Calls Descriptions



user Operations about user

GET `/user/{userId}/relevantAssets` Get recommendations for user id

asset Operations about data asset

GET `/asset/{assetId}/similarAssets` Get data assets similar to a reference data asset

7.2.3 Sequence Diagrams

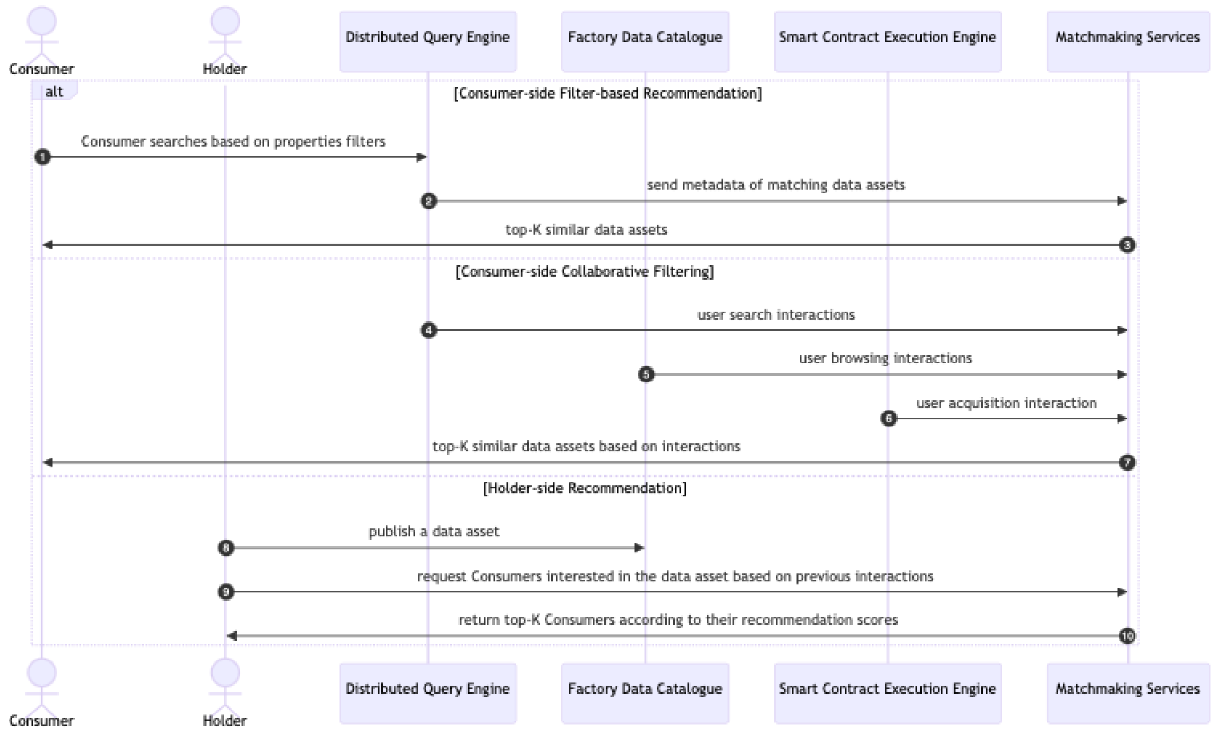

Consumer
Holder
Distributed Query Engine
Factory Data Catalogue
Smart Contract Execution Engine
Matchmaking Services
alt
[Consumer-side Filter-based Recommendation]
1 Consumer searches based on properties filters
2 send metadata of matching data assets
3 top-K similar data assets
[Consumer-side Collaborative Filtering]
4 user search interactions
5 user browsing interactions
6 user acquisition interaction
7 top-K similar data assets based on interactions
[Holder-side Recommendation]
8 publish a data asset
9 request Consumers interested in the data asset based on previous interactions
10 return top-K Consumers according to their recommendation scores

Figure 21: Matchmaking Services Sequence Diagram

8 DATA EXCHANGE BUNDLE

The Data Exchange bundle facilitates the peer-to-peer exchange of the data assets between a Data Provider and a Data Consumer, adhering to the terms of the contract that has been signed to govern the overall transaction.

This bundle consists of the following components:

- PISTIS Data Factory Connector
- Smart Contract Checker

These are presented in the following sub-sections.

8.1 PISTIS DATA FACTORY CONNECTOR

The transfer of data between different PISTIS users (e.g., Data Providers and Data Consumers) is the logical termination point of a monetary or otherwise exchange agreement flow, where, following the establishment of an electronic contract, the data set that is part of the agreement must reach the Data Consumer.

The overall transfer in PISTIS is facilitated by the PISTIS Data Factory Connector (or else, the PISTIS Connector), which is a software component that is tasked, once a data transfer contract needs to be executed, to fetch the data stored in the PISTIS Data Factory of the Data Provider and pass it to the PISTIS Data Factory of the Data Consumer.

This transfer is to be performed following the appropriate checks at smart contract level that will govern such exchanges (based on license, usage and permission attributes stored in the ledger), and the result will be the deposition of the data asset purchased by the Data Consumer in their own, local data storage.

8.1.1 RELATION TO OTHER COMPONENTS

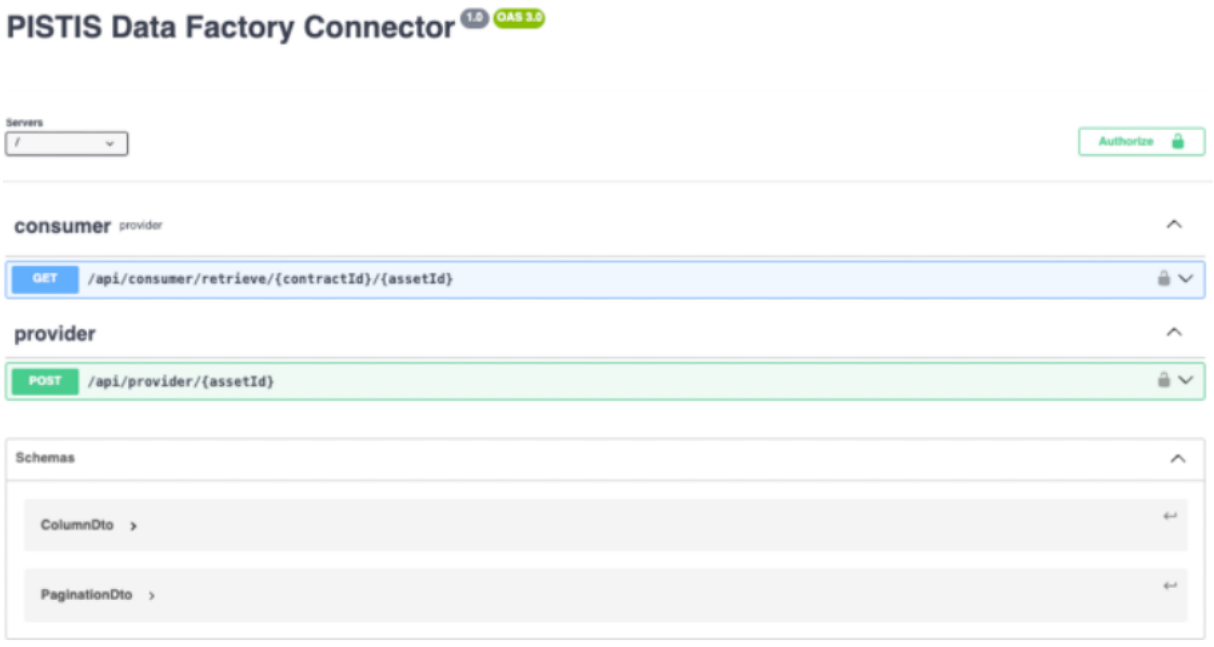
8.1.1.1 Components providing Input to Peer-to-Peer Data Transfer Gateway

Input Required	Component providing the Input	Communication Method
Information about executing the transaction and its terms	Smart Contract Execution Engine	API
Data Payload to be transferred to the Data Consumer	PISTIS Data Storage	API
Metadata to be transferred to the Data Consumer	Factory Data Catalogue	API
Address of the PISTIS Factory of the Data Consumer	Data Factories Registrant	API

8.1.1.2 Components to which Peer-to-Peer Data Transfer Gateway provides input

Input Required	Component providing the Input	Communication Method
Information about the result of the transaction	Smart Contract Execution Engine	API
Transferred Data Payload	PISTIS Data Storage	API
Transferred Metadata	Factory Data Catalogue	API

8.1.2 API Calls Descriptions


PISTIS Data Factory Connector 1.0 OAS 3.0

Servers
/

Authorize

consumer provider

GET /api/consumer/retrieve/{contractId}/{assetId}

provider

POST /api/provider/{assetId}

Schemas

ColumnDto

PaginationDto

8.1.3 SEQUENCE DIAGRAMS

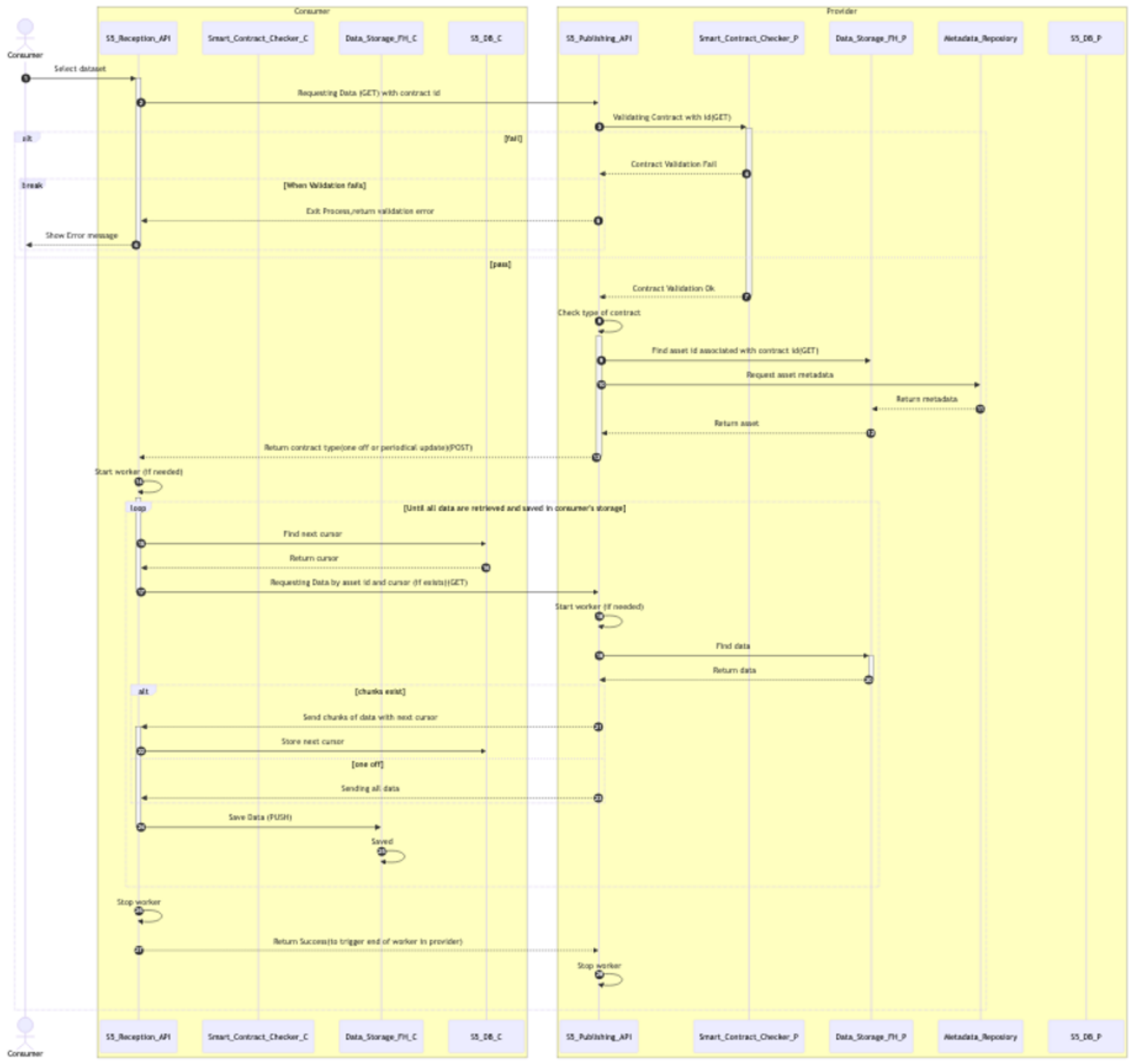


Figure 22: PISTIS Data Factory Connector Sequence Diagram

8.2 SMART CONTRACT CHECKER

The Smart Contract Checker within the PISTIS platform is a sophisticated component designed to ensure the integrity and validity of transactions related to data transfers. Its policy is based on two pillars:

- **Authenticity Check**

  For the Alpha version, the Smart Contract Checker verifies the source component calling it. Every component in PISTIS (including the SC Checker) must authenticate the calling component by checking its JWT using the public key. This authentication process applies to all PISTIS components as an **orthogonal functionality** across the platform and is not exclusive to the SC Checker.

In future Beta versions, the SC Checker will also validate users through their ID wallet to ensure that the user belongs to the correct organisation. **This user-level authentication will not be part of the Alpha release.**

- **Violation Check**

  The SC Checker ensures there is no violation in executing future data transactions, including:

  - **Allowed to Transfer Times:** Check if the number of permitted transfers (NoOfTimes) has been exceeded.
  - **Transfer Frequency:** Validate if the transfer frequency (e.g., per hour, day, week) is within allowed limits.
  - **Transfer Until:** Verify that transactions occur before the specified timestamp.
  - **ReShare Times Limit:** Check if the limit on the number of times the asset can be reshared has been breached.
  - **ReSell Times Limit:** Check if the limit on the number of times the asset can be resold has been breached.

The Smart Contract Checker is an integral component of the PISTIS platform, instilling confidence in the platform's transactions. By rigorously validating user authenticity and transaction logic, it upholds the platform's security and compliance standards. This comprehensive approach to transaction validation ensures that PISTIS remains a secure, trustworthy, and user-compliant platform, capable of handling complex data and financial transactions with utmost integrity. Figure 18 presents the high-level architecture of the Smart Contract Checker including the internal components of the tool.

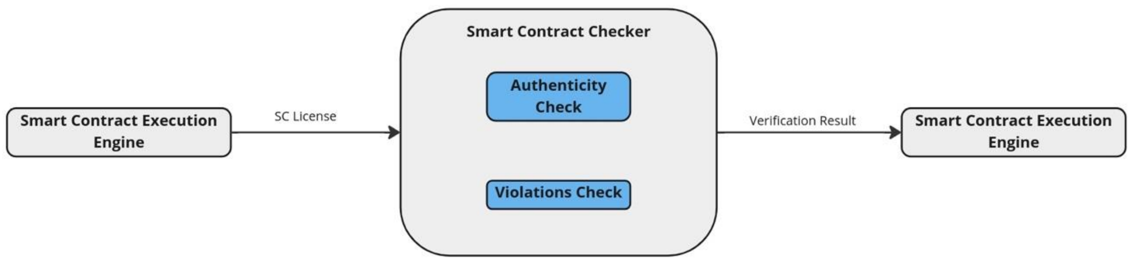


Figure 23: Smart Contract Checker High-level Architecture

8.2.1 RELATION TO OTHER COMPONENTS

8.2.1.1 Components providing Input to Smart Contract Checker component

Input Required	Component providing the Input	Communication Method
User credentials for Smart Contract Verification (Beta Version)	Data Factory User Wallet	API
Smart Contract to be checked and its functional logic	Smart Contract Execution Engine	IPC

8.2.1.2 Components to which Smart Contract Checker component provides input

Output Served	Component ingesting the output	Communication Method
Smart Contract violations and validity checked	Smart Contract Execution Engine	IPC

8.2.2 API CALLS DESCRIPTIONS

The Smart Contract Checker communicates **only** with the Smart Contract Execution Engine via internal communication using IPC (Inter-Process Communication). It does not expose any APIs to external components or systems outside the PISTIS environment. This ensures that interactions remain isolated within the platform.

8.2.3 SEQUENCE DIAGRAMS

The next figure depicts the sequence of actions for the Smart Contract Checker component.

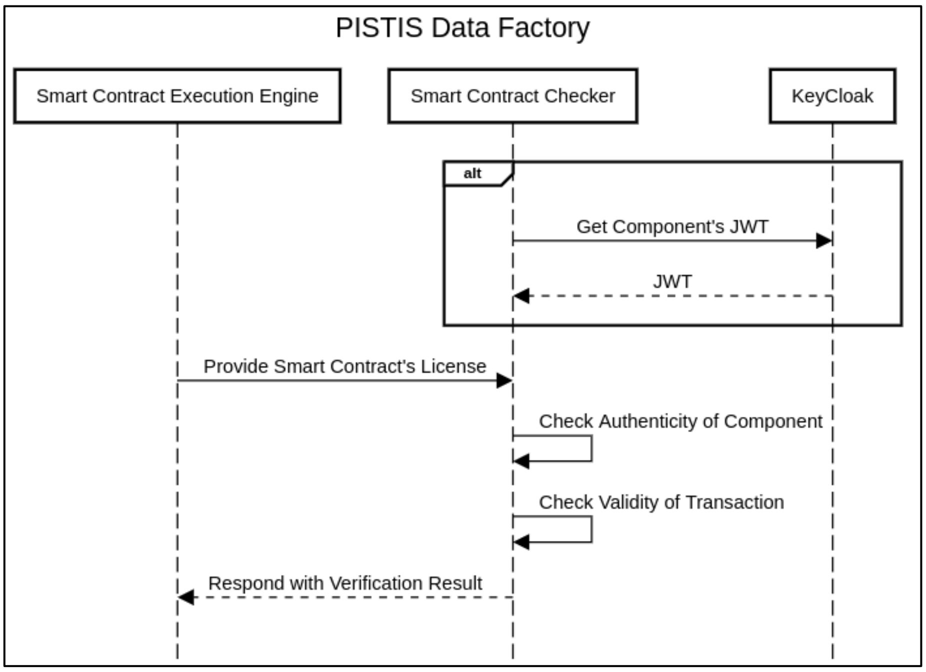

PISTIS Data Factory
Smart Contract Execution Engine
Smart Contract Checker
KeyCloak
alt
Get Component's JWT
JWT
Provide Smart Contract's License
Check Authenticity of Component
Check Validity of Transaction
Respond with Verification Result

Figure 24: Smart Contract Checker Sequence Diagram

9 DATA MONETISATION BUNDLE

The Data Monetisation bundle provides different services for the Data provider to understand the value of its data within the market environment of the platform. Furthermore, it supports the Data Provider to place its/her data on the PISTIS Data Catalogue using different methods to monetise, by setting up and upon the interest of a Data Consumer engaging in transactions performed with the aid of blockchain technology.

This bundle consists of the following components:

- Access Policy Editor
- Asset Description Bundler
- FAIR Data Valuation Service
- PISTIS Digital DLT-FIAT Wallet
- Data Investment Planner
- PISTIS Market Insights
- NFT Generator
- Monetisation Plan Designer
- Data Usage and Intentions Analytics

These are presented in the following sub-sections.

9.1 ACCESS POLICY EDITOR

The Access Policy Editor, a critical component integrated into the robust Keycloak identity and access management platform within PISTIS, serves as a centralised tool empowering PISTIS Data Factory Administrators to define and apply the access policies for the data to be placed over the PISTIS Catalogue.

The primary goal is to simplify the definition of scope-based policies through an intuitive web-based UI editor. By doing so, administrators gain the ability to tailor access controls for specific use cases, encompassing user roles, and access to targeted resources within the PISTIS ecosystem. This strategic functionality ensures that the access policies align with the unique organisational structure and requirements so as data in the PISTIS Data Catalogue are findable with proper access to other organisations; either internal or external PISTIS ecosystem.

Access policies generated by the Keycloak-based Access Policy Editor provide a multifaceted approach to access control. Firstly, they precisely dictate who has the privilege to access distinct PISTIS Organisation resources, ranging from specific features within the PISTIS Platform to exclusive datasets owned by the organisation. Secondly, these policies define the scope or rights associated with accessible PISTIS Organisation resources. This extends beyond conventional permissions, including Create, Read, Update, Delete, and Admin, to incorporate PISTIS-specific policies such as Trading, Transformation, Pricing, and more. Additionally, the editor allows administrators to finely tune access on nested objects or attributes within a specific PISTIS Organisation's resource. For example, an administrator can grant read access to an entire data stream while restricting update permissions to a specific child attribute, providing granular control over resource accessibility.

Internally, the Access Policy Editor relies on Keycloak's robust infrastructure. Keycloak employs a secure and scalable database system to store and retrieve the intricate policies defined by administrators. This ensures that policy information is organised, quickly accessible, and securely managed. Furthermore, Keycloak leverages its advanced indexing service to optimize the efficiency of policy enforcement during runtime. The indexing service plays a pivotal role in accelerating the retrieval of policies, contributing to the overall responsiveness and performance of the Access Policy Editor within the PISTIS platform. These internal components work seamlessly to provide a dynamic, responsive, and secure access control mechanism tailored to the specific needs of organisations utilizing the Keycloak-based Access Policy Editor in the PISTIS environment.

Access Policy Editor within PISTIS also provides the access policies as part of the blockchain, as they will be part of the Asset Description bundle; namely as information related to Smart Contracts.

9.1.1 Relation to Other Components

9.1.1.1 Components providing Input to Access Policy Editor

Input Required	Component providing the Input	Communication Method
Users, Roles, Groups, Scopes, Resources	Identity Manager	HTTP Request

9.1.1.2 Components to which Access Policy Editor provides input

Output Served	Component ingesting the output	Communication Method
Access Policies	Access Policies Engine	HTTP Request
Access Policies on Data Assets	Asset Description Bundler	HTTP Request

9.1.2 API Calls Descriptions

N/A - This is a frontend service not providing services to other components.

9.1.3 Sequence Diagrams

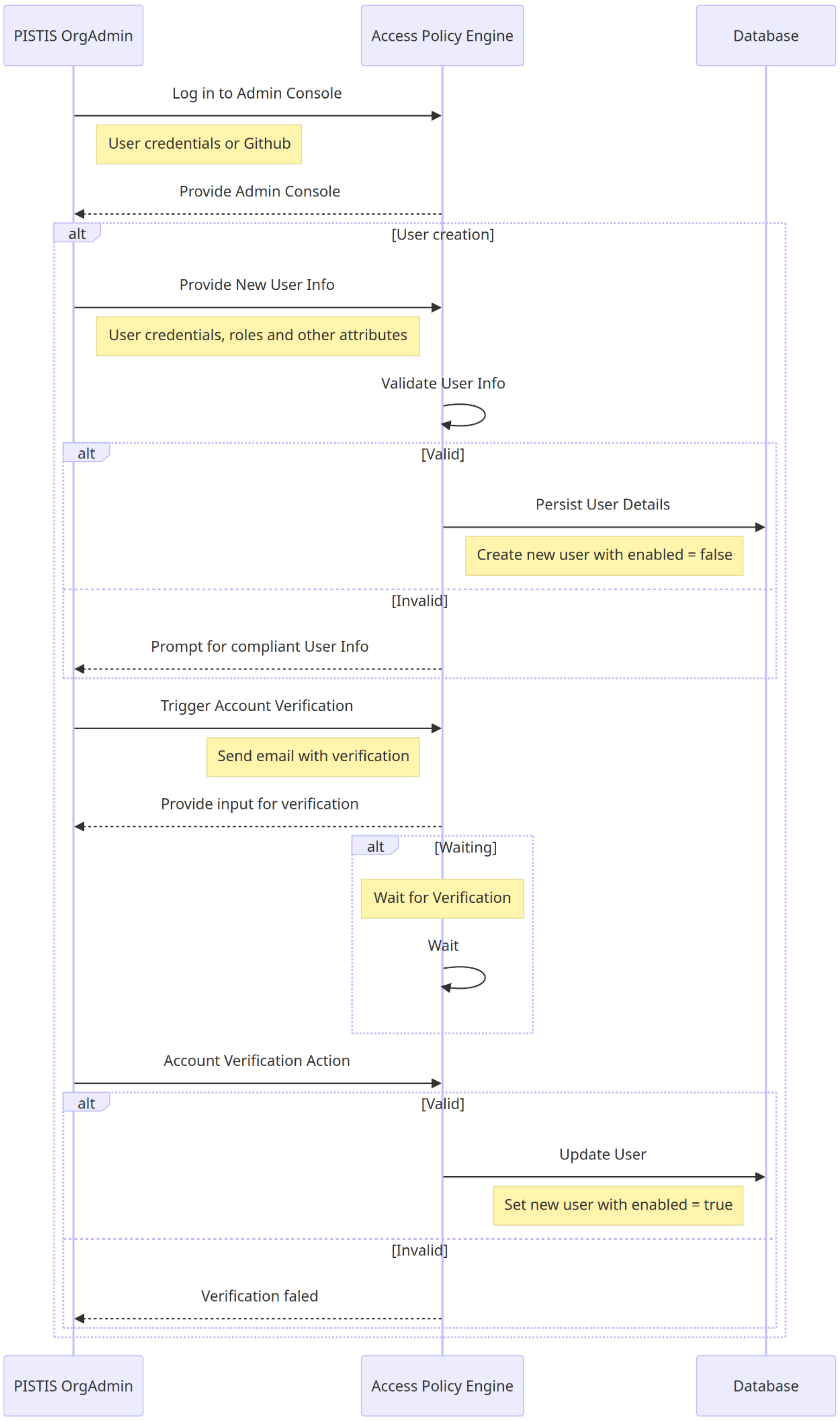


Figure 25: Access Policy Editor Sequence Diagram #1 – User Verification

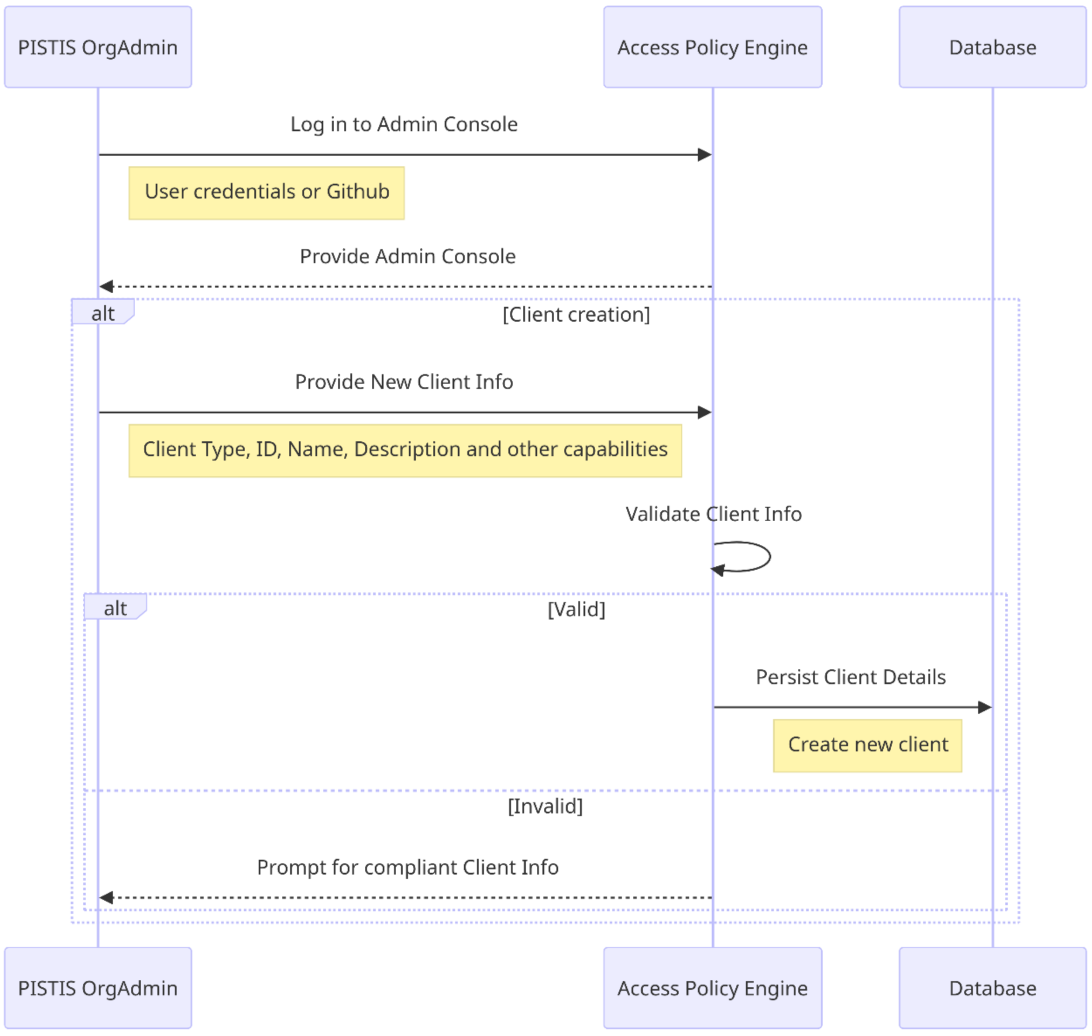


Figure 26: Access Policy Editor Sequence Diagram #2 – Client Creation

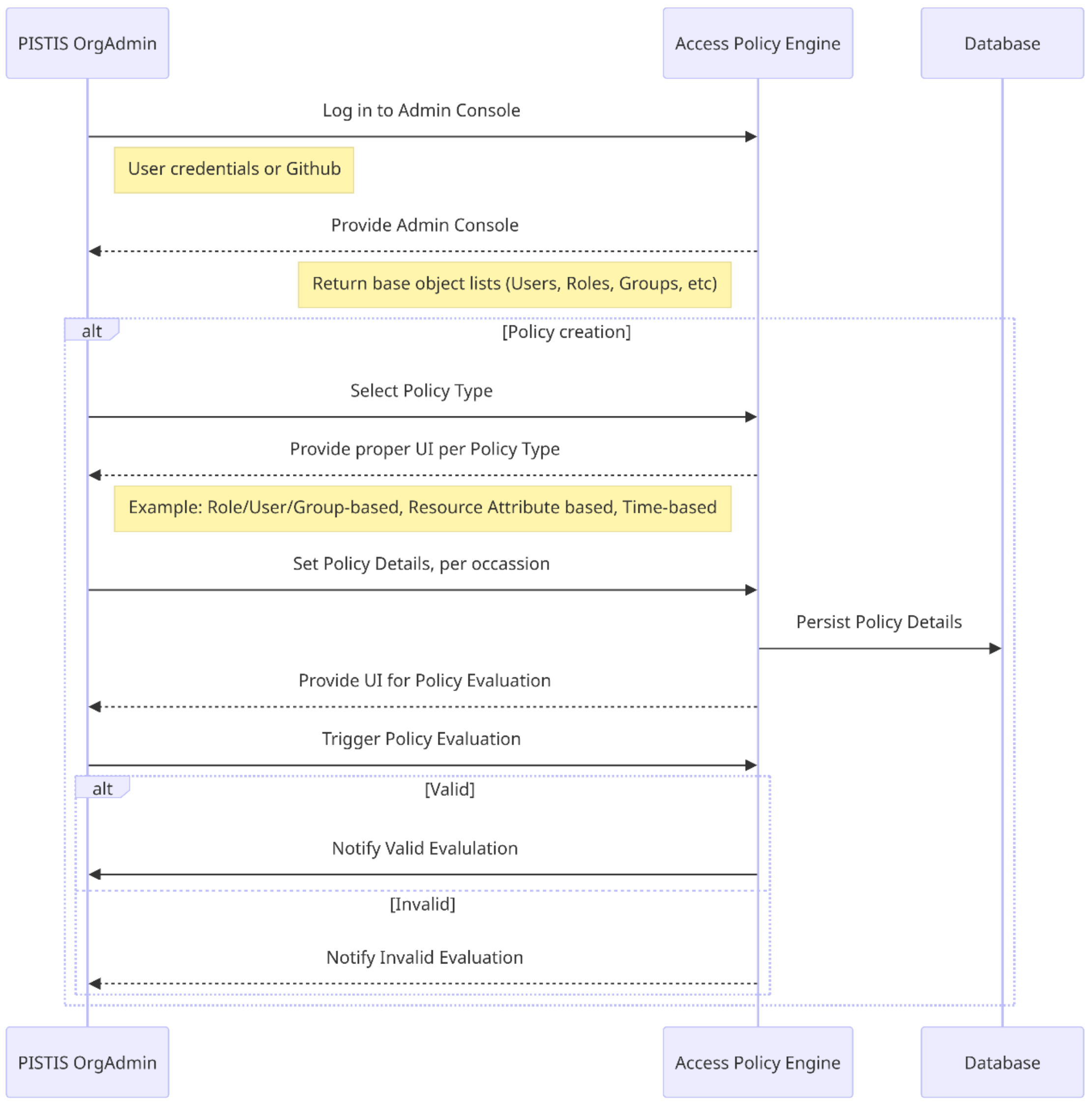

PISTIS OrgAdmin
Access Policy Engine
Database
Log in to Admin Console
User credentials or Github
Provide Admin Console
Return base object lists (Users, Roles, Groups, etc)
alt
[Policy creation]
Select Policy Type
Provide proper UI per Policy Type
Example: Role/User/Group-based, Resource Attribute based, Time-based
Set Policy Details, per occassion
Persist Policy Details
Provide UI for Policy Evaluation
Trigger Policy Evaluation
alt
[Valid]
Notify Valid Evaluation
[Invalid]
Notify Invalid Evaluation
PISTIS OrgAdmin
Access Policy Engine
Database

Figure 27: Access Policy Editor Sequence Diagram #3 – Policy Setting and Evaluation

9.2 Asset Description Bundler

The Asset Description Bundler (ADB) puts together a package which fully describes the data asset that is placed for trading. The foundation of the bundle is the original metadata, upon which the ADB gathers metadata related to the value of the data asset. Lastly, the ADB gathers metadata related to the value of the data asset (in the form of data value dimensions) and references information related to the smart contract (access policies, monetisation option, smart contract ID, execution status).

The ADB is the intermediate layer facilitating the communication between other components and the catalogues.

The functionalities of the ADB are:

1. Provides a semantic model to encapsulate the information needed for data asset trading.

2. Collects and updates basic asset metadata.
3. Collects and updates metadata about data value dimensions.
4. Collects and updates information related to smart contract execution: access policies, monetisation option, smart contract ID, execution status.

The metadata bundled by the ADB:

1. *Basic metadata available upon Data Check-In*:
    a. provenance
    b. license
    c. lineage
    d. format
    e. accessibility
    f. etc.
2. *Data value dimensions*:
    a. valuation context (purpose, weights, business value)
    b. data quality metrics
    c. functional data utility (optional) metrics
    d. bias assessment metrics
    e. anonymisation metrics OR deanonymisation risk
    f. GDPR check
    g. aggregate data value score
3. *Information related to smart contracts*:
    a. access policies
    b. monetisation option
    c. smart contract ID
    d. contract execution status

The ADB will be built with the help of a semantic model suited for the description and exchange of the information previously presented. This will be achieved using a standard for key-value descriptions (JSON, YAML) or an RDF-based solution.

9.2.1 Relation to Other Components

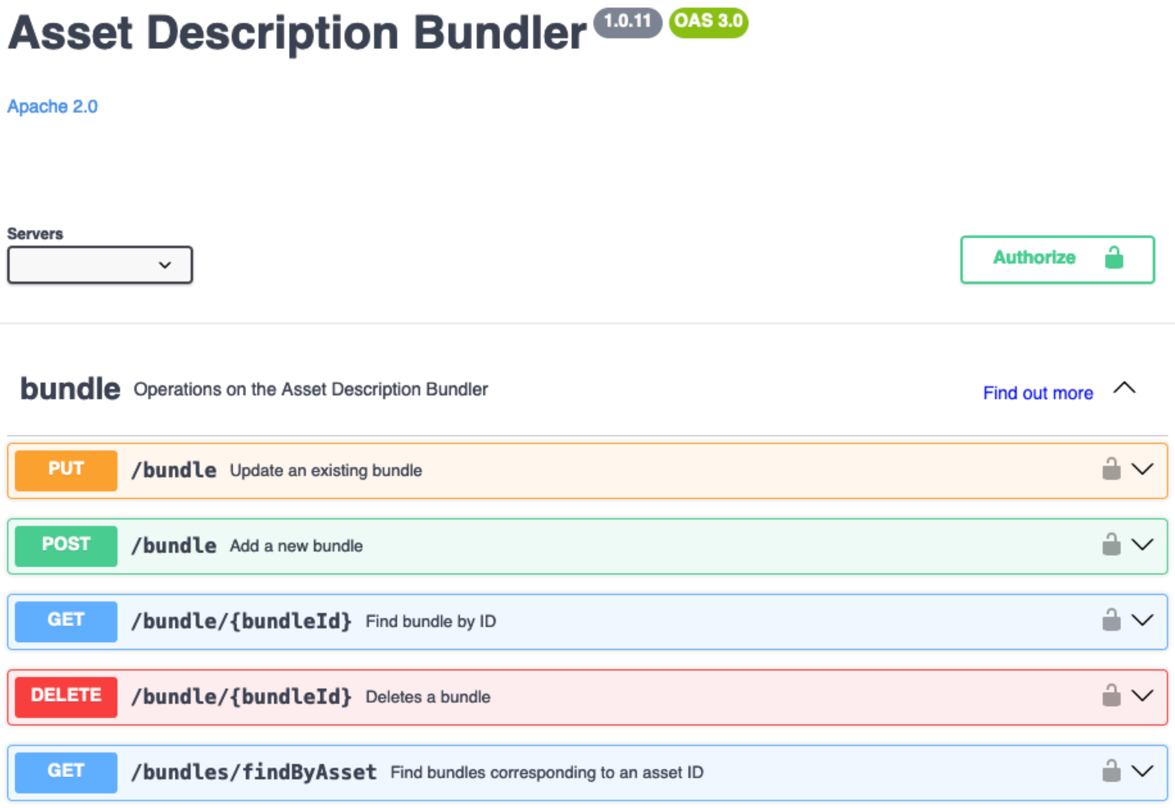
9.2.1.1 Components providing Input to the Asset Description Bundler

Input Required	Component providing the Input	Communication Method
Basic metadata	PISTIS Data Catalogue	API
Basic metadata (if asset not published by Owner)	Factory Data Catalogue	API
Data value dimensions (see Section 1.1) and aggregated data value score	FAIR Data Valuation Service	API
Selected access policy	Access Policy Editor	API
Selected monetisation option	Monetisation Plan Designer	API
Smart contract ID	Smart Contract Template Composer	API
Smart contract execution status	Smart Contract Execution Engine	API

9.2.1.2 *Components to which the Asset Description Bundler provides input*

Output Served	Component ingesting the output	Communication Method
Data asset bundle	PISTIS Data Catalogue	API
Data asset bundle	Smart Contract Template Composer	API

9.2.2 API Calls Descriptions


Asset Description Bundler 1.0.11 OAS 3.0

Apache 2.0

Servers

Authorize

bundle Operations on the Asset Description Bundler Find out more

PUT /bundle Update an existing bundle

POST /bundle Add a new bundle

GET /bundle/{bundleId} Find bundle by ID

DELETE /bundle/{bundleId} Deletes a bundle

GET /bundles/findByAsset Find bundles corresponding to an asset ID

9.2.3 Sequence Diagrams

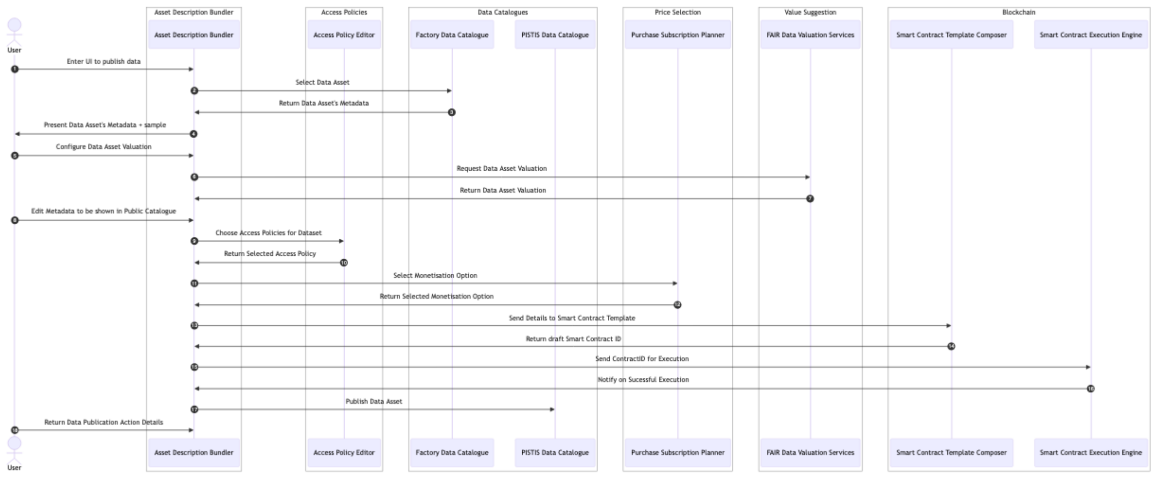

Asset Description Bundler
Access Policies
Data Catalogues
Price Selection
Value Suggestion
Blockchain
User
Asset Description Bundler
Access Policy Editor
Factory Data Catalogue
PISTIS Data Catalogue
Purchase Subscription Planner
FAIR Data Valuation Services
Smart Contract Template Composer
Smart Contract Execution Engine
Enter UI to publish data
Select Data Asset
Return Data Asset's Metadata
Present Data Asset's Metadata + sample
Configure Data Asset Valuation
Request Data Asset Valuation
Return Data Asset Valuation
Edit Metadata to be shown in Public Catalogue
Choose Access Policies for Dataset
Return Selected Access Policy
Select Monetisation Option
Return Selected Monetisation Option
Send Details to Smart Contract Template
Return draft Smart Contract ID
Send ContractID for Execution
Notify on Sucessful Execution
Publish Data Asset
Return Data Publication Action Details

Figure 28: Asset Description Bundler Sequence Diagram

9.3 FAIR Data Valuation Service

The FAIR Data Valuation Service (DVS) comprises of a set of methods that support the task of data valuation – assigning a quantitative value to a data asset. It is a complex, context-dependent, and multi-dimensional process, built upon several other processes: data and metadata quality assessment, functional data utility assessment, feature bias assessment, legal and ethical assessment, privacy analysis. While the service will mainly support a dimensional approach to data valuation, our ambition is for the component to integrate and/or support market-based and economic models.

The main functionalities of the FAIR Data Valuation Service are:

1. Allow for the User to define the data valuation context: purpose, relevant dimensions, business value/goals, IP requirements. This will be achieved by designing a dedicated UI.
2. Retrieves metadata relevant to the data value dimensions (DVDs) of a selected data asset: basic metadata (provenance, access policy), usage and intentions, data quality metrics, anonymisation metrics, bias metrics, GDPR compliance, market insights. If these are not available, it will query other components' APIs trying to retrieve them (Usage & Intentions Analytics, Data Quality Assessment, Repo for Pre-trained AI Models, Market Insights, Anonymisation, GDPR Checker, Lineage Tracker).
3. Functional utility checks to assess the contextual value of each data point in a structured dataset – currently available for classification or regression problems.
4. Implements a Data Valuation Methodology to aggregate the quantification of each DVD into an aggregate Data Value Score.
5. Reports an aggregate Data Value Score, together with a breakdown along the DVDs.
6. Communicates the results of the data valuation process to instances of the Asset Description Bundler (ADB).

9.3.1 Relation to Other Components

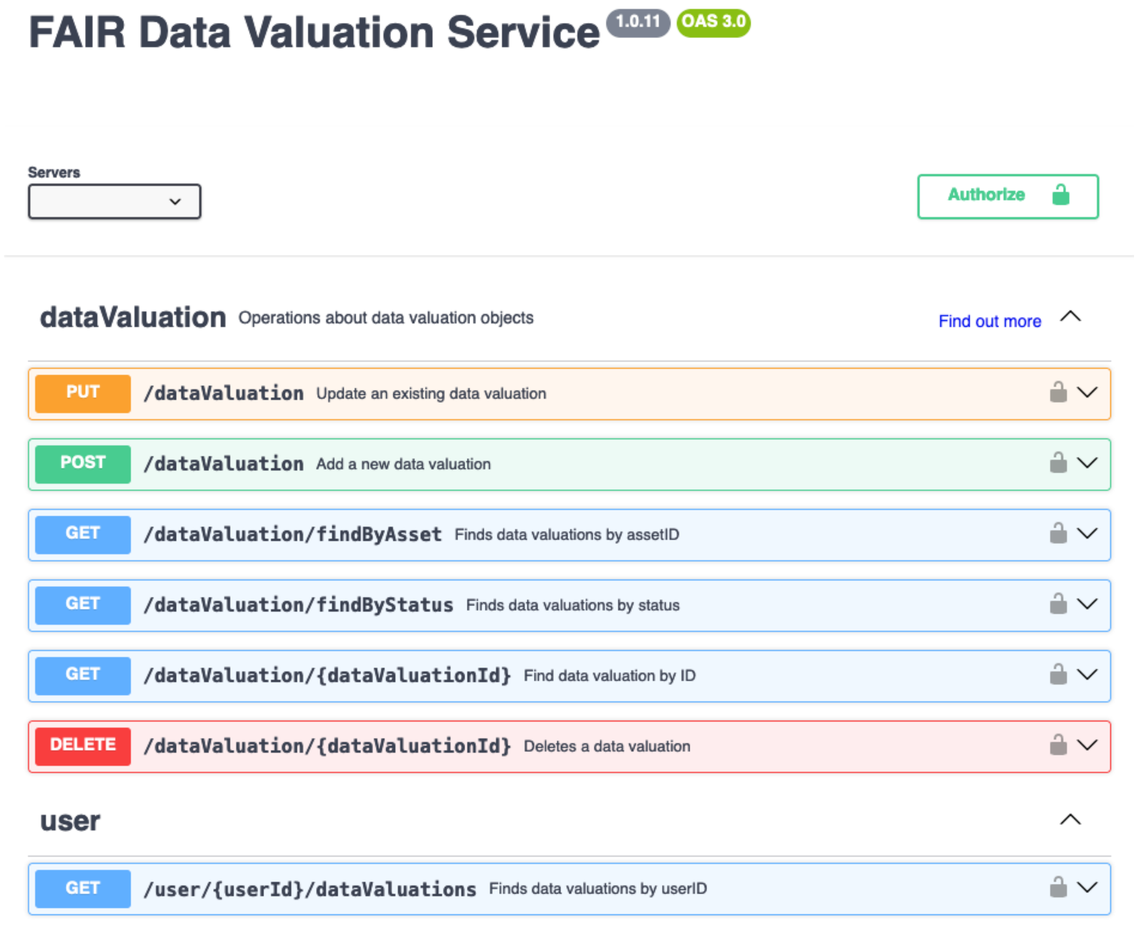
9.3.1.1 Components providing Input to the FAIR Data Valuation Service

Input Required	Component providing the Input	Communication Method
Metadata (provenance, access policy, lineage, purpose, data quality, GDPR, anonymisation, market insights)	PISTIS Data Catalogue	API
Profile and purpose	Analytics/Data Insights Generator	API
Previous uses and transformations	Lineage Tracker	API
Access policy / License	Access Policy Engine	API
Data quality metrics	Data Quality Assessment	API
Anonymisation metrics	Anonymisation	API
GDPR compliance	GDPR Checker	API
Interest in this and similar datasets	Market Insights	API
ML models	Pre-trained AI models repo	API

9.3.1.2 Components to which the FAIR Data Valuation Service provides input

Output Served	Component ingesting the output	Communication Method
Data value dimensions and data valuation results	Asset Description Bundler	API

9.3.2 API Calls Descriptions



FAIR Data Valuation Service 1.0.11 OAS 3.0

Servers

Authorize

dataValuation Operations about data valuation objects — Find out more

Method	Path	Description
PUT	`/dataValuation`	Update an existing data valuation
POST	`/dataValuation`	Add a new data valuation
GET	`/dataValuation/findByAsset`	Finds data valuations by assetID
GET	`/dataValuation/findByStatus`	Finds data valuations by status
GET	`/dataValuation/{dataValuationId}`	Find data valuation by ID
DELETE	`/dataValuation/{dataValuationId}`	Deletes a data valuation

user

Method	Path	Description
GET	`/user/{userId}/dataValuations`	Finds data valuations by userID

9.3.3 Sequence Diagrams

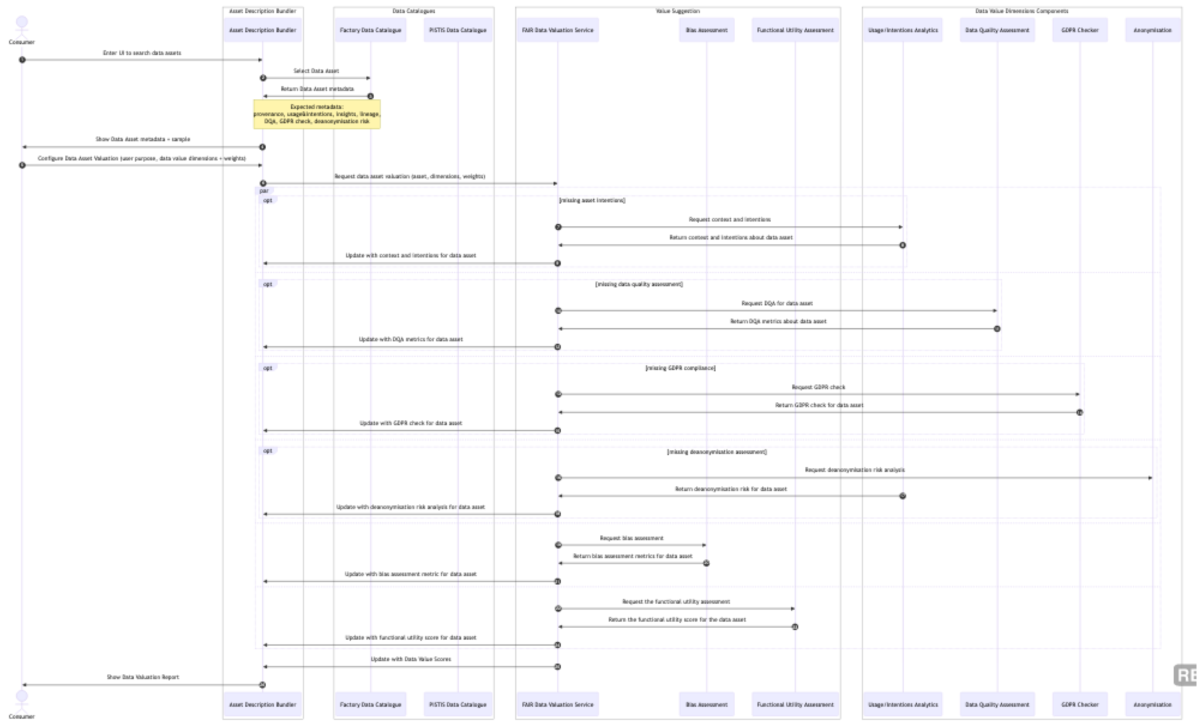

Consumer
Asset Description Bundler
Data Catalogues
Factory Data Catalogue
PISTIS Data Catalogue
Value Suggestion
FAIR Data Valuation Service
Bias Assessment
Functional Utility Assessment
Data Value Dimensions Components
Usage/Intentions Analytics
Data Quality Assessment
GDPR Checker
Anonymisation
Enter UI to search data assets
Select Data Asset
Return Data Asset metadata
Expected metadata: provenance, usage&intentions, insights, lineage, DQA, GDPR check, deanonymisation risk
Show Data Asset metadata + sample
Configure Data Asset Valuation (user purpose, data value dimensions + weights)
Request data asset valuation (asset, dimensions, weights)
par
opt
[missing asset intentions]
Request context and intentions
Return context and intentions about data asset
Update with context and intentions for data asset
opt
[missing data quality assessment]
Request DQA for data asset
Return DQA metrics about data asset
Update with DQA metrics for data asset
opt
[missing GDPR compliance]
Request GDPR check
Return GDPR check for data asset
Update with GDPR check for data asset
opt
[missing deanonymisation assessment]
Request deanonymisation risk analysis
Return deanonymisation risk for data asset
Update with deanonymisation risk analysis for data asset
Request bias assessment
Return bias assessment metrics for data asset
Update with bias assessment metric for data asset
Request the functional utility assessment
Return the functional utility score for the data asset
Update with functional utility score for data asset
Update with Data Value Scores
Show Data Valuation Report

Figure 29: FAIR Data Valuation Service Sequence Diagram

9.4 PISTIS Digital DLT-FIAT Wallet

This component is responsible for facilitating value transactions using cryptocurrencies within the public IOTA Shimmer network, guaranteeing quick and reliable exchanges. It also plays a pivotal role in interfacing with traditional FIAT banking accounts[4], enabling seamless fund transfers within the system and facilitating conversion between FIAT and cryptocurrencies. Cryptocurrency transactions are processed within the DLT network, maintaining a 1:1 exchange ratio between FIAT money and PISTIS crypto coins, without the need for a liquidity mechanism. In the alpha version, a mocking mechanism has been implemented to emulate interactions with open banking systems (PSD2).

9.4.1 Relation to Other Components

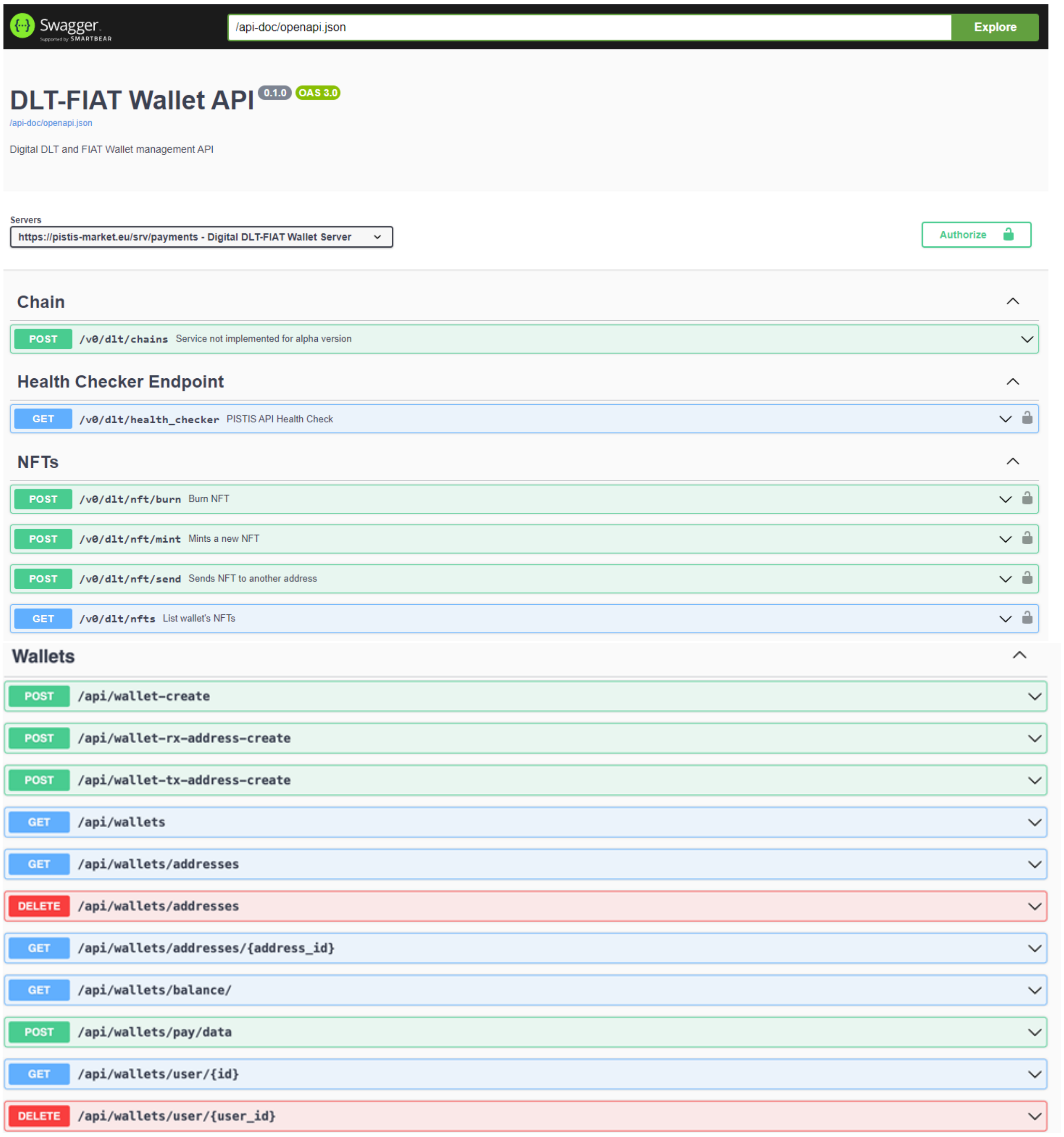
9.4.1.1 Components providing Input to Digital DLT-FIAT Wallet

Input Required	Component providing the Input	Communication Method
User Authentication Token	Data Factory User Wallet	API
Component's authentication token (acquisition and refresh)	Identity Provider	API
Currency amount to be converted from FIAT to PISTIS (or the other way around)	Data Factory User Wallet	API
Information about DLT value transaction related actions	Data Factory User Wallet	API

9.4.1.2 Components to which Digital DLT-FIAT Wallet provides input

Output Served	Component ingesting the output	Communication Method
Result of DLT wallet creation	Data Factory User Wallet	API
Information about DLT wallet balance and digital assets	Data Factory User Wallet	API
Information about DLT wallet history/previous actions	Data Factory User Wallet	API
Currency conversion result	Data Factory User Wallet	API
Information about specific DLT wallet's address	Data Factory User Wallet	API
Result of executed DLT value transaction	Data Factory User Wallet	API
DLT Value transaction outcome notification	Data Factory User Wallet, Transaction Auditor	API

9.4.2 API Calls Descriptions



DLT-FIAT Wallet API 0.1.0 OAS 3.0

/api-doc/openapi.json

Digital DLT and FIAT Wallet management API

Servers

https://pistis-market.eu/srv/payments - Digital DLT-FIAT Wallet Server

Chain

POST /v0/dlt/chains Service not implemented for alpha version

Health Checker Endpoint

GET /v0/dlt/health_checker PISTIS API Health Check

NFTs

POST /v0/dlt/nft/burn Burn NFT

POST /v0/dlt/nft/mint Mints a new NFT

POST /v0/dlt/nft/send Sends NFT to another address

GET /v0/dlt/nfts List wallet's NFTs

Wallets

POST /api/wallet-create

POST /api/wallet-rx-address-create

POST /api/wallet-tx-address-create

GET /api/wallets

GET /api/wallets/addresses

DELETE /api/wallets/addresses

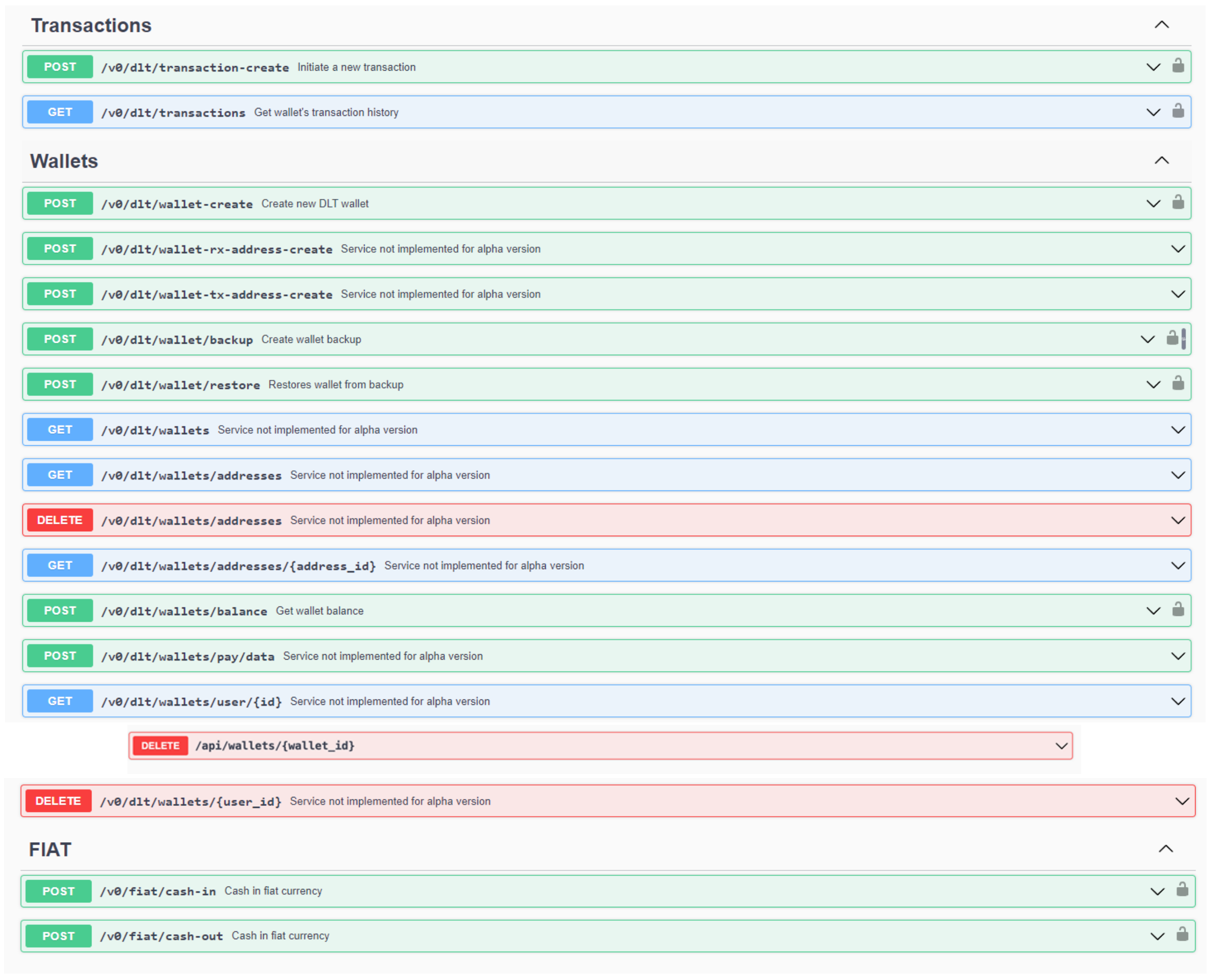
GET /api/wallets/addresses/{address_id}

GET /api/wallets/balance/

POST /api/wallets/pay/data

GET /api/wallets/user/{id}

DELETE /api/wallets/user/{user_id}



Transactions

POST /v0/dlt/transaction-create Initiate a new transaction

GET /v0/dlt/transactions Get wallet's transaction history

Wallets

POST /v0/dlt/wallet-create Create new DLT wallet

POST /v0/dlt/wallet-rx-address-create Service not implemented for alpha version

POST /v0/dlt/wallet-tx-address-create Service not implemented for alpha version

POST /v0/dlt/wallet/backup Create wallet backup

POST /v0/dlt/wallet/restore Restores wallet from backup

GET /v0/dlt/wallets Service not implemented for alpha version

GET /v0/dlt/wallets/addresses Service not implemented for alpha version

DELETE /v0/dlt/wallets/addresses Service not implemented for alpha version

GET /v0/dlt/wallets/addresses/{address_id} Service not implemented for alpha version

POST /v0/dlt/wallets/balance Get wallet balance

POST /v0/dlt/wallets/pay/data Service not implemented for alpha version

GET /v0/dlt/wallets/user/{id} Service not implemented for alpha version

DELETE /api/wallets/{wallet_id}

DELETE /v0/dlt/wallets/{user_id} Service not implemented for alpha version

FIAT

POST /v0/fiat/cash-in Cash in fiat currency

POST /v0/fiat/cash-out Cash in fiat currency

9.4.3 SEQUENCE DIAGRAMS

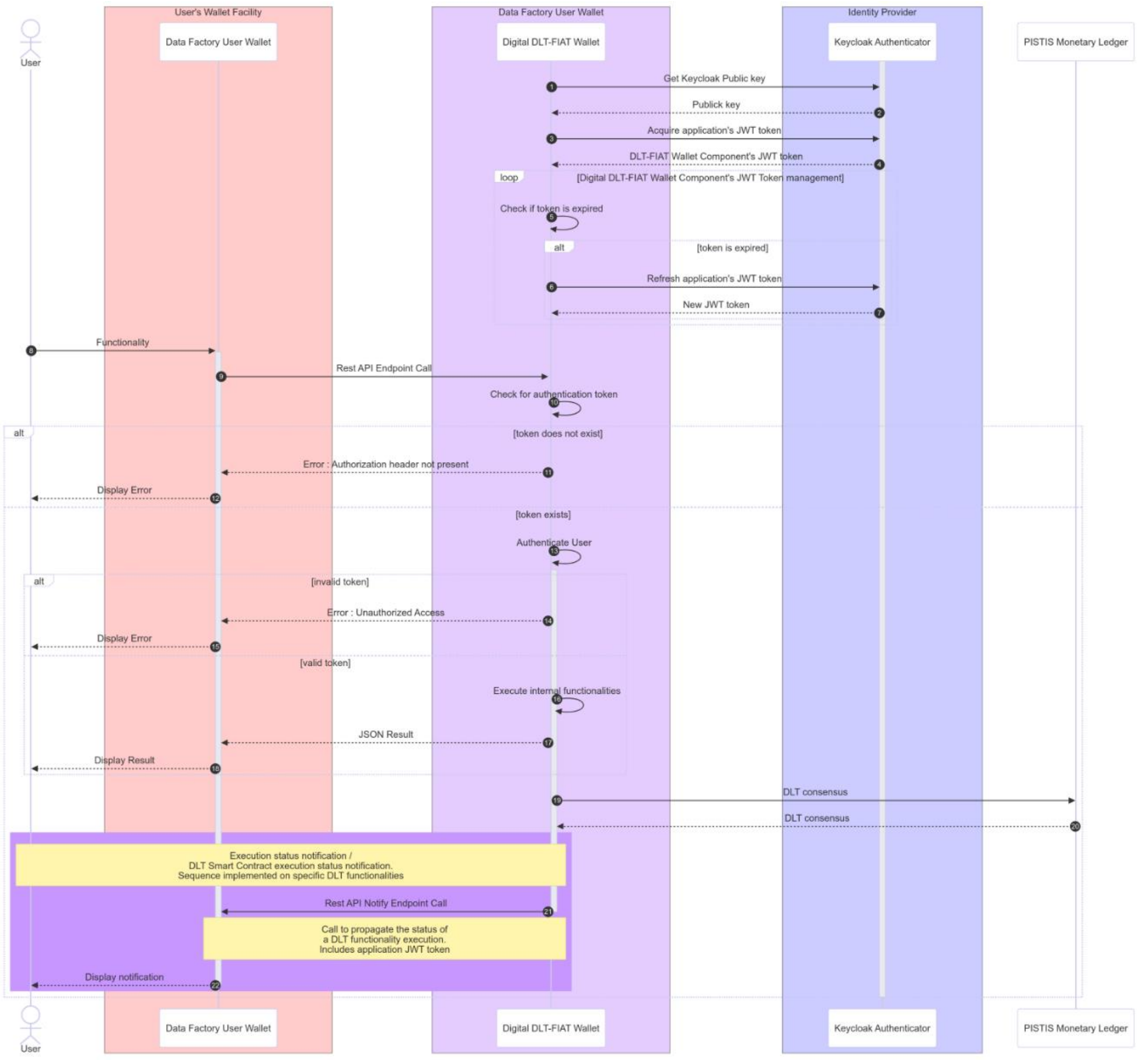

User
User's Wallet Facility
Data Factory User Wallet
Data Factory User Wallet
Digital DLT-FIAT Wallet
Identity Provider
Keycloak Authenticator
PISTIS Monetary Ledger
Get Keycloak Public key
Publick key
Acquire application's JWT token
DLT-FIAT Wallet Component's JWT token
loop
[Digital DLT-FIAT Wallet Component's JWT Token management]
Check if token is expired
alt
[token is expired]
Refresh application's JWT token
New JWT token
Functionality
Rest API Endpoint Call
Check for authentication token
alt
[token does not exist]
Error : Authorization header not present
Display Error
[token exists]
Authenticate User
alt
[invalid token]
Error : Unauthorized Access
Display Error
[valid token]
Execute internal functionalities
JSON Result
Display Result
DLT consensus
DLT consensus
Execution status notification / DLT Smart Contract execution status notification. Sequence implemented on specific DLT functionalities
Rest API Notify Endpoint Call
Call to propagate the status of a DLT functionality execution. Includes application JWT token
Display notification
User
Data Factory User Wallet
Digital DLT-FIAT Wallet
Keycloak Authenticator
PISTIS Monetary Ledger

Figure 30: Digital DLT-FIAT Wallet Sequence Diagram - #1

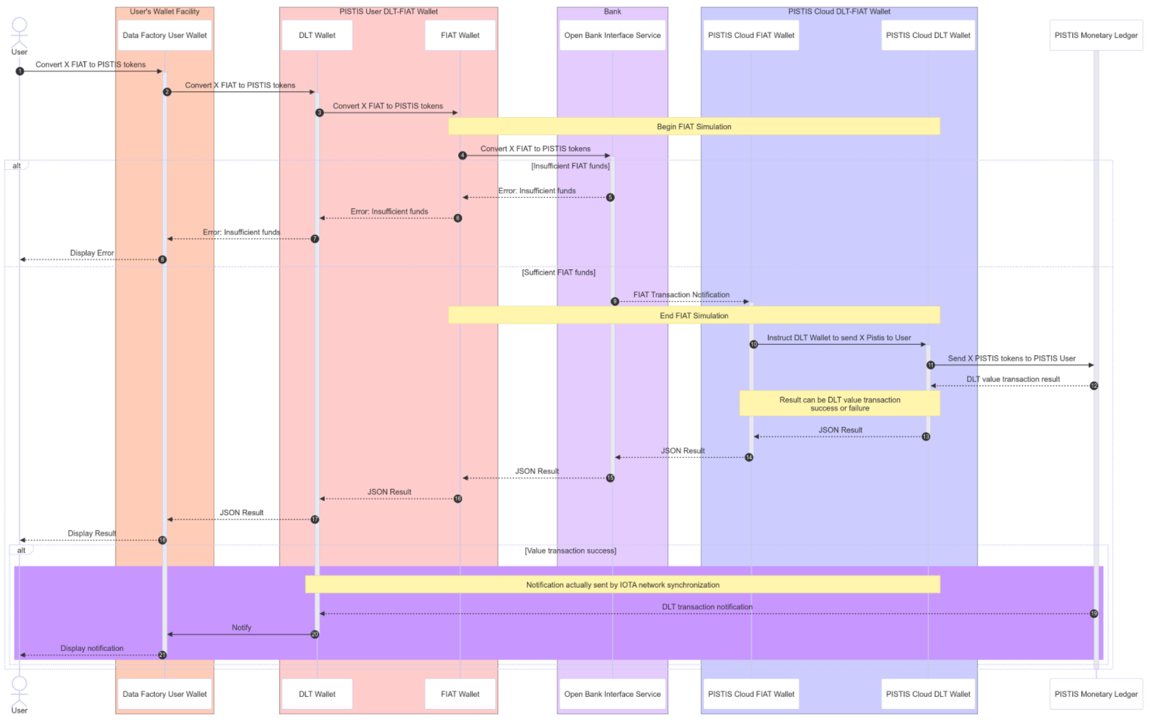

User's Wallet Facility
PISTIS User DLT-FIAT Wallet
Bank
PISTIS Cloud DLT-FIAT Wallet
User
Data Factory User Wallet
DLT Wallet
FIAT Wallet
Open Bank Interface Service
PISTIS Cloud FIAT Wallet
PISTIS Cloud DLT Wallet
PISTIS Monetary Ledger
Convert X FIAT to PISTIS tokens
Convert X FIAT to PISTIS tokens
Convert X FIAT to PISTIS tokens
Begin FIAT Simulation
Convert X FIAT to PISTIS tokens
alt
[Insufficient FIAT funds]
Error: Insufficient funds
Error: Insufficient funds
Error: Insufficient funds
Display Error
[Sufficient FIAT funds]
FIAT Transaction Notification
End FIAT Simulation
Instruct DLT Wallet to send X Pistis to User
Send X PISTIS tokens to PISTIS User
DLT value transaction result
Result can be DLT value transaction success or failure
JSON Result
JSON Result
JSON Result
JSON Result
JSON Result
Display Result
alt
[Value transaction success]
Notification actually sent by IOTA network synchronization
DLT transaction notification
Notify
Display notification

Figure 31: Digital DLT-FIAT Wallet Sequence Diagram - #2

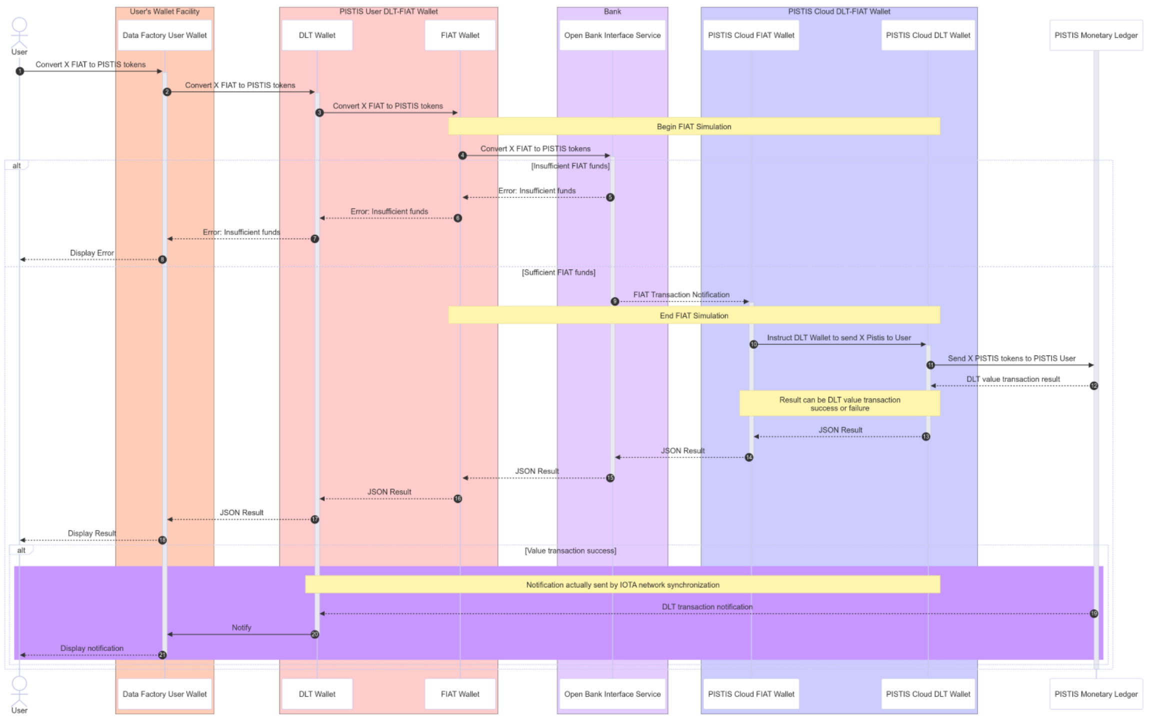

User's Wallet Facility
PISTIS User DLT-FIAT Wallet
Bank
PISTIS Cloud DLT-FIAT Wallet
User
Data Factory User Wallet
DLT Wallet
FIAT Wallet
Open Bank Interface Service
PISTIS Cloud FIAT Wallet
PISTIS Cloud DLT Wallet
PISTIS Monetary Ledger
Convert X FIAT to PISTIS tokens
Convert X FIAT to PISTIS tokens
Convert X FIAT to PISTIS tokens
Begin FIAT Simulation
Convert X FIAT to PISTIS tokens
alt
[Insufficient FIAT funds]
Error: Insufficient funds
Error: Insufficient funds
Error: Insufficient funds
Display Error
[Sufficient FIAT funds]
FIAT Transaction Notification
End FIAT Simulation
Instruct DLT Wallet to send X Pistis to User
Send X PISTIS tokens to PISTIS User
DLT value transaction result
Result can be DLT value transaction success or failure
JSON Result
JSON Result
JSON Result
JSON Result
JSON Result
Display Result
alt
[Value transaction success]
Notification actually sent by IOTA network synchronization
DLT transaction notification
Notify
Display notification

Figure 32: Digital DLT-FIAT Wallet Sequence Diagram - #3

9.5 Data Investment Planner

The Data Investment Planner represents an innovative approach poised to transform the way the financing of data set management and trading is approached. By enabling participants in the supply chain to create an entirely different financing framework by sharing capital, data owners and providers are allowed to raise investment by offering a share of the potential profits generated from their data.

The Data Investment Planner is responsible for designing an investment plan related to a specific dataset of a data provider. Once a dataset is registered in PISTIS, the data provider enters the Monetisation Plan Designer and inserts information about the minimum and total equity percentage, the equity price and the maximum number of investors.

The information is sent to the Data Investment Planner which, after analysing this information, generates a suggested investment plan to the data provider that can be followed as-is, or can be used as a basis for further investment configurations by the data provider towards trading his/her dataset and obtain monetary gains.

9.5.1 Relation to Other Components

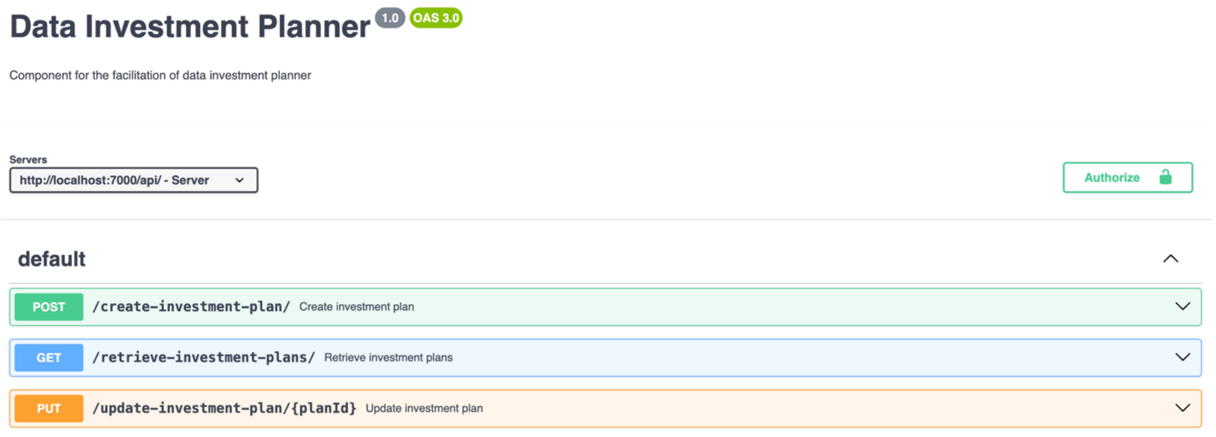
9.5.1.1 *Components providing Input to Data Investment Planner*

Input Required	Component providing the Input	Communication Method
Information for investment	Monetisation Plan Designer	API

9.5.1.2 *Components to which Data Investment Planner provides input*

Output Served	Component ingesting the output	Communication Method
Generated investment plan	Monetisation Plan Designer	API

9.5.2 API Calls Descriptions


Data Investment Planner 1.0 OAS 3.0

Component for the facilitation of data investment planner

Servers
http://localhost:7000/api/ - Server

Authorize

default

POST /create-investment-plan/ Create investment plan

GET /retrieve-investment-plans/ Retrieve investment plans

PUT /update-investment-plan/{planId} Update investment plan

9.5.3 SEQUENCE DIAGRAMS

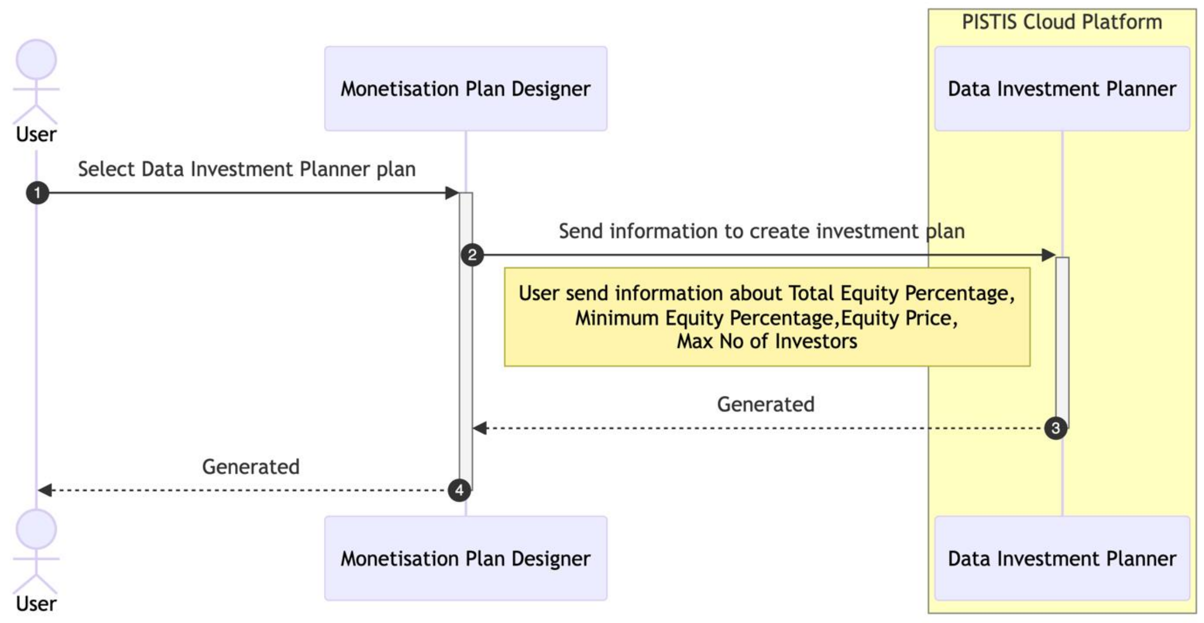

PISTIS Cloud Platform
User
Monetisation Plan Designer
Data Investment Planner
Select Data Investment Planner plan
Send information to create investment plan
User send information about Total Equity Percentage, Minimum Equity Percentage,Equity Price, Max No of Investors
Generated
Generated
Monetisation Plan Designer
Data Investment Planner
User

Figure 33: Data Investment Planner Sequence Diagram

9.6 PISTIS MARKET INSIGHTS

The PISTIS Market Insights component is responsible for offering insights about the transactions performed in the PISTIS Marketplace, by running designated analytics on the available transaction metadata residing in the different protected data ledgers of PISTIS stakeholders.

The PISTIS Market Insights components provides to the users with predefined dashboards for exploring market relevant information (e.g., the PISTIS ecosystem transactions). The retrieved related transactions are indexed and specific models from the PISTIS Models Repository are exploited for the execution of analytics, based on the user query, while the analytics results are displayed in dashboards, allowing the user to understand in depth the specificities of transactions in PISTIS.

9.6.1 RELATION TO OTHER COMPONENTS

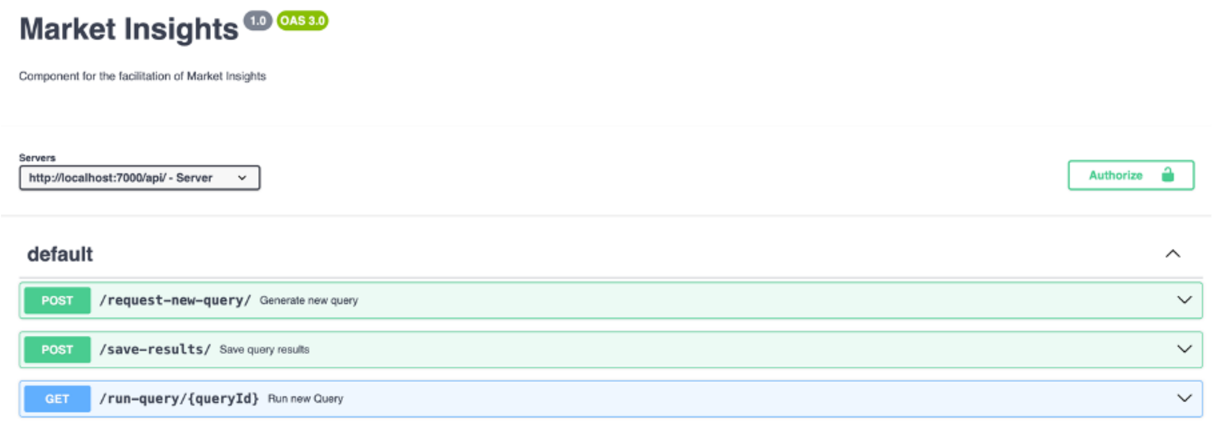
9.6.1.1 Components providing Input to PISTIS Market Insights

Input Required	Component providing the Input	Communication Method
Relevant Transactions	Data Ledger	API
Analytics Results	Analytics Engine	API

9.6.1.2 Components to which PISTIS Market Insights provides input

Output Served	Component ingesting the output	Communication Method
N/A		

9.6.2 API Calls Descriptions


Market Insights 1.0 OAS 3.0

Component for the facilitation of Market Insights

Servers
http://localhost:7000/api/ - Server

Authorize

default

POST /request-new-query/ Generate new query

POST /save-results/ Save query results

GET /run-query/{queryId} Run new Query

9.6.3 Sequence Diagrams

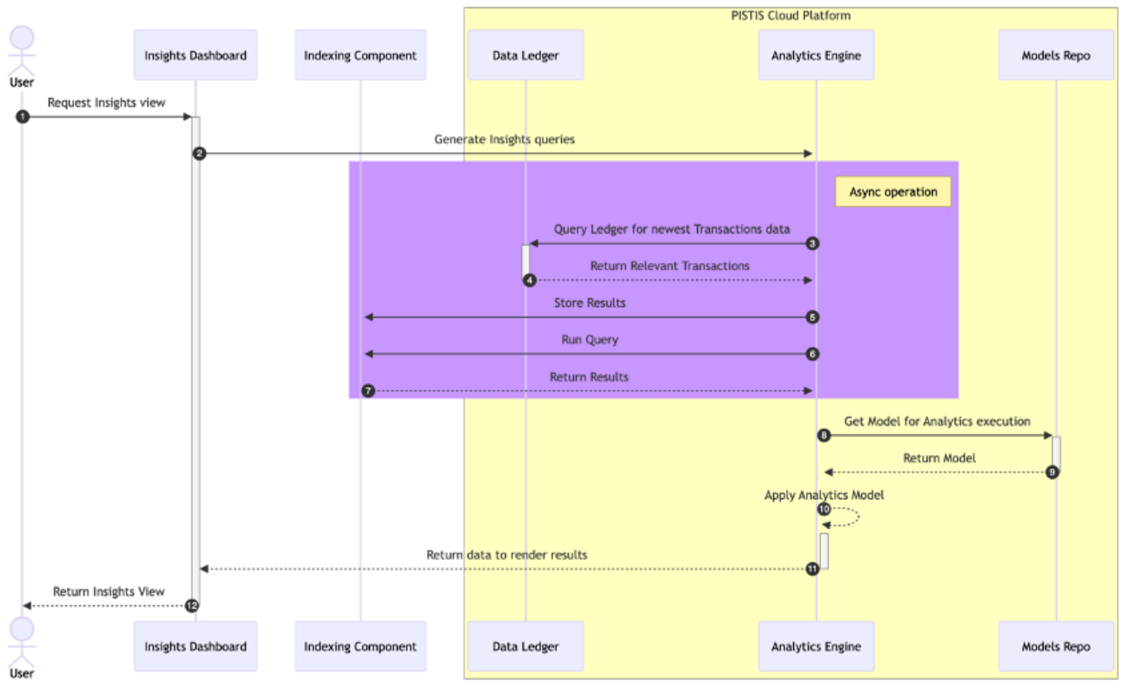


Figure 34: Market Insights Sequence Diagram

9.7 NFT Generator

The NFT Generator lies within the Data Monetisation bundle of the PISTIS Cloud platform and is responsible for generating the appropriate Data NFT, once the users select it as his/her preferred monetisation method.

This component is triggered by the Monetisation Plan Designer once the user selects to sell an owned dataset through NFT licensing. The Monetisation Plan Designer provides all the required information to the NTF Generator for generating an NFT, and the latter is using the

IOTA Layer 1 DLT core protocol to generate the NFT and place it on the ledger, thus making it available for exchange.

Currently the NFT Generator is not provided as a distinct component. All the related NFT functionalities (including NFT exchange transactions and NFT burning) are provided as part of the Digital DLT FIAT Wallet instance in the PISTIS Platform cloud cluster factory. A stand-alone dedicated NFT Generator component will be provided in the Beta version of the PISTIS Platform.

9.7.1 Relation to Other Components

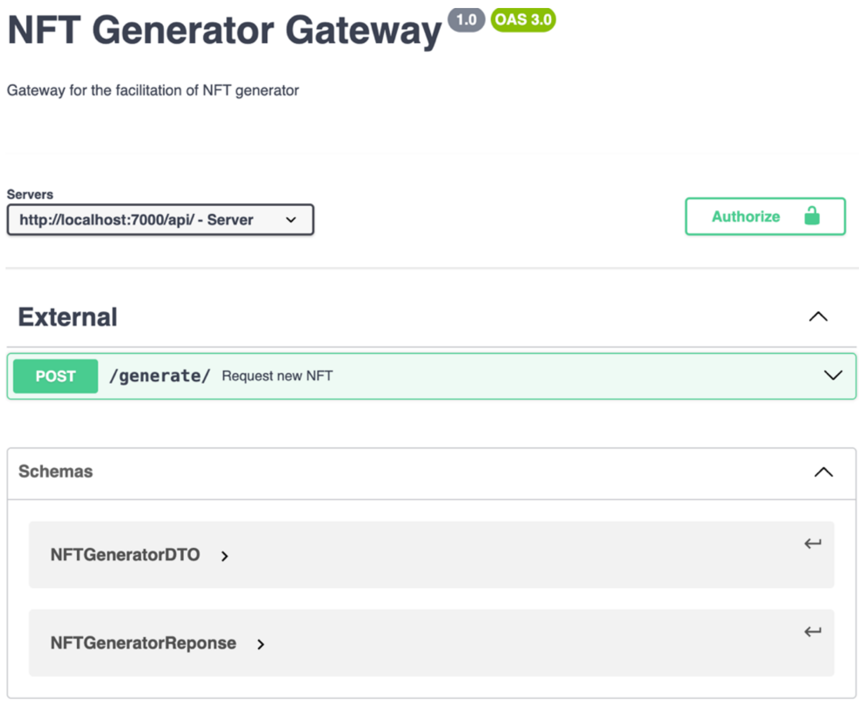
9.7.1.1 Components providing Input to the NFT Generator

Input Required	Component providing the Input	Communication Method
Information to generate an NFT	Monetisation Plan Designer	API
NFT Token id returned upon execution	Smart Contract Execution Engine	API

9.7.1.2 Components to which the NFT Generator provides input

Output Served	Component ingesting the output	Communication Method
Dataset's Information to mint the NFT	Smart Contract Execution Engine	API
Generated NFT	Monetisation Plan Designer	REST API

9.7.2 API Calls Descriptions



9.7.3 Sequence Diagrams

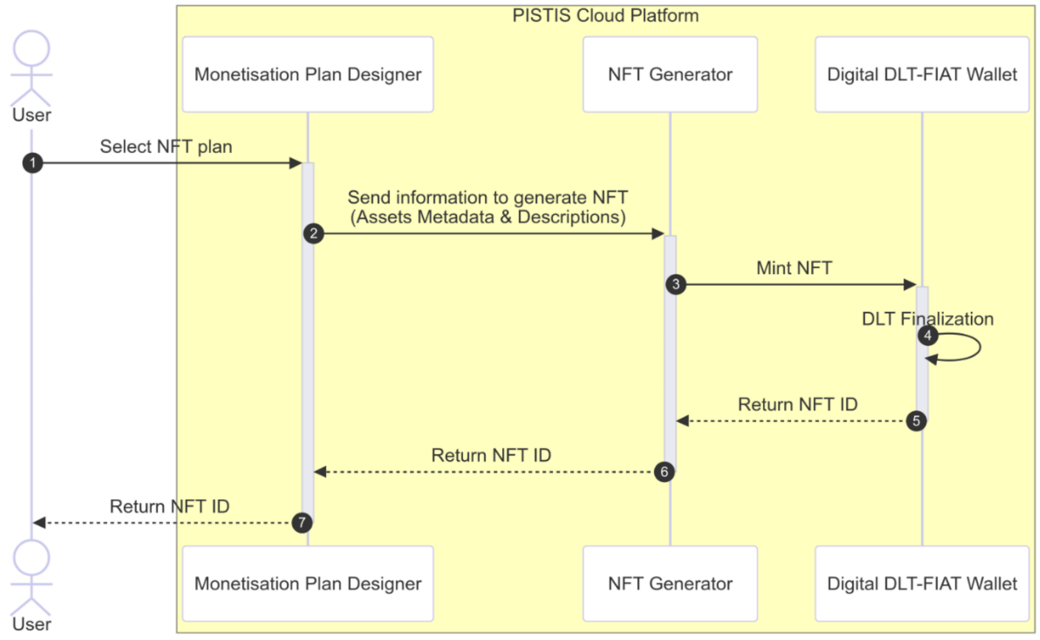


Figure 35: NFT Generator- Sequence Diagram

9.8 Monetisation Plan Designer

The Monetisation Plan Designer, is an integral component of the Data Value Contract Composer, located on the PISTIS Cloud Platform that is essential for seamless data interactions within the PISTIS ecosystem. It essentially enables data providers to put their datasets into the market and specify their desired monetisation method, with the ultimate scope to simplify the complexities of the data transactions creating a streamlined and effective system that lays the foundation for a thriving data-centric ecosystem.

This component embodies a sophisticated mechanism that orchestrates two key options for data dissemination:

I. *One-off purchase*, which enables instantaneous availability (i.e., on-demand access) of specific data assets or services, catering to immediate requirements of users without the need for a long-term commitment.

II. *Subscription Plans*, which involves the provision of data or data-related services through tailored subscription plans (i.e., through a predefined subscription fee). This option aims at ensuring a flexible and scalable approach to data accessibility. By offering various subscription models, data providers are empowered to structure their offerings with granularity, meeting the diverse needs of data stakeholders.

Two additional monetisation methods are also provided to the data providers, including:

- the *selling of the NFT of their assets*, where the data provider specifies the price of the NFT; and
- *the creation of a Data Investment plan for the asset*, where users shall specify the max number of investors, the total equity (%), the minimum equity (%) and the equity price.

To fulfil its functionalities the Monetisation Plan Designer communicates with a variety of data monetisation, trading services and value design components, depending on the users preferred monetisation route, including: the Asset Description Bundler where user can select the dataset for which he/she will define the monetisation method and where the final user's monetisation plan details will be stored; the data FAIR Data valuation Service towards receiving data valuation suggestions. In the case of an NFT plan the component sends information to the NFT Generator to generate the relevant NFT; while in the case of an Investment Plan the component communicates with the Data Investment Planner towards sending the required information for the configuration of the appropriate Investment plan.

9.8.1 Relation to Other Components

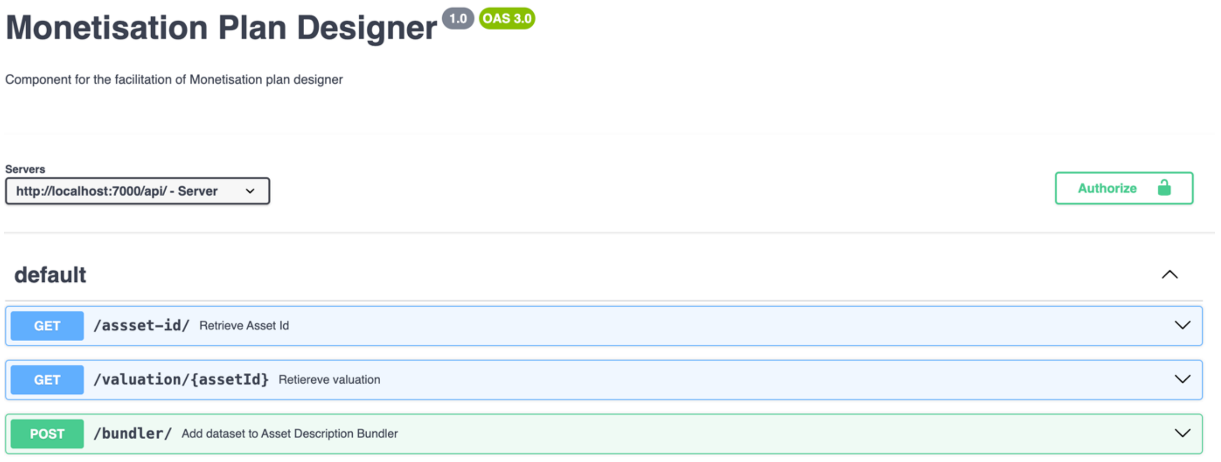
9.8.1.1 Components providing Input to the Monetisation Plan Designer

Input Required	Component providing the Input	Communication Method
Dataset selection	Asset Description Bundler	API
Data valuation suggestion	FAIR Data Valuation Service	API
Generate NFT plan	NFT Generator	API
Generation of Investment Plan	Data Investment Planner	API

9.8.1.2 Components to which the Monetisation Plan Designer provides input

Output Served	Component ingesting the output	Communication Method
Data monetisation plan details	Asset Description Bundler	API

9.8.2 API Calls Descriptions


Monetisation Plan Designer 1.0 OAS 3.0

Component for the facilitation of Monetisation plan designer

Servers
http://localhost:7000/api/ - Server

default

GET /asset-id/ Retrieve Asset Id

GET /valuation/{assetId} Retiereve valuation

POST /bundler/ Add dataset to Asset Description Bundler

9.8.3 SEQUENCE DIAGRAMS

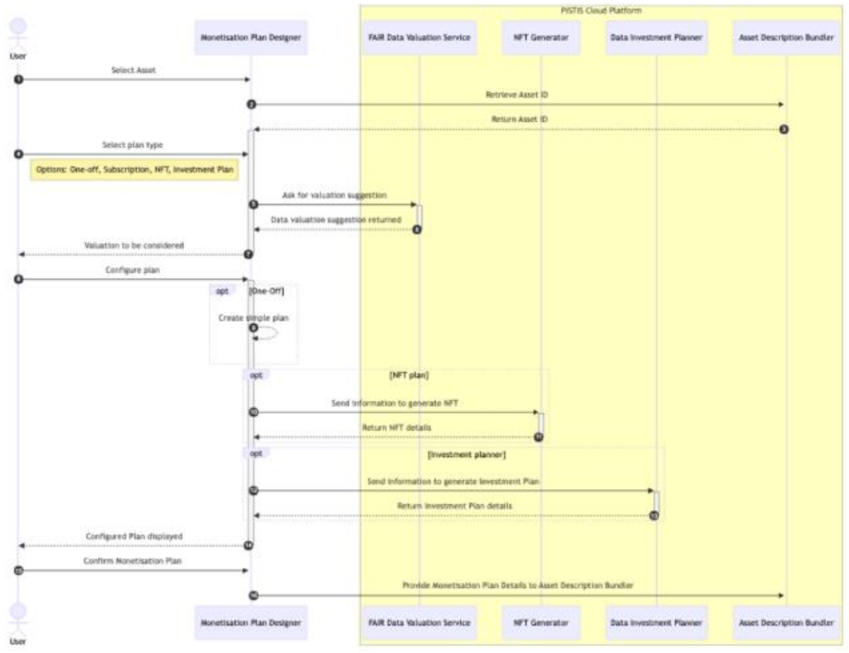

PISTIS Cloud Platform
User
Monetisation Plan Designer
FAIR Data Valuation Service
NFT Generator
Data Investment Planner
Asset Description Bundler
Select Asset
Retrieve Asset ID
Return Asset ID
Select plan type
Options: One-off, Subscription, NFT, Investment Plan
Ask for valuation suggestion
Data valuation suggestion returned
Valuation to be considered
Configure plan
opt
[One-Off]
Create simple plan
opt
[NFT plan]
Send information to generate NFT
Return NFT details
opt
[Investment planner]
Send information to generate Investment Plan
Return Investment Plan details
Configured Plan displayed
Confirm Monetisation Plan
Provide Monetisation Plan Details to Asset Description Bundler

Figure 36: Monetisation Plan Designer - Sequence Diagram

9.9 DATA USAGE AND INTENTIONS ANALYTICS

The Data Usage and Intentions Analytics located within the Monetisation XAI Engine of the PISTIS cloud platform is designed to unravel the complexities of data utilisation within an organisation by delving into the underlying intentions of data owners and users.

By examining data usage patterns, the duration of usage and associated online activities, this component offers a comprehensive dashboard enabling data owners to clearly understand their data assets and the contextual landscape in which these assets may be employed.

The insights gained from these visualisations empower data owners to make efficient decisions not only about how their data is processed but also about the way it is shared, based on a comprehensive understanding of their data landscape; and altogether contributes to enhancing data quality and security.

The Data Usage and Intentions Analytics communicates with the PISTIS Data Catalogue for the retrieval of datasets and their metadata. In addition, the component communicates with the Data Ledger to retrieve the relevant datasets' lineage, and any recorded transactions made on similar datasets. By processing all this input, the Data Usage and Intentions Analytics presents to users' optimal methods for data exploitation.

9.9.1 Relation to Other Components

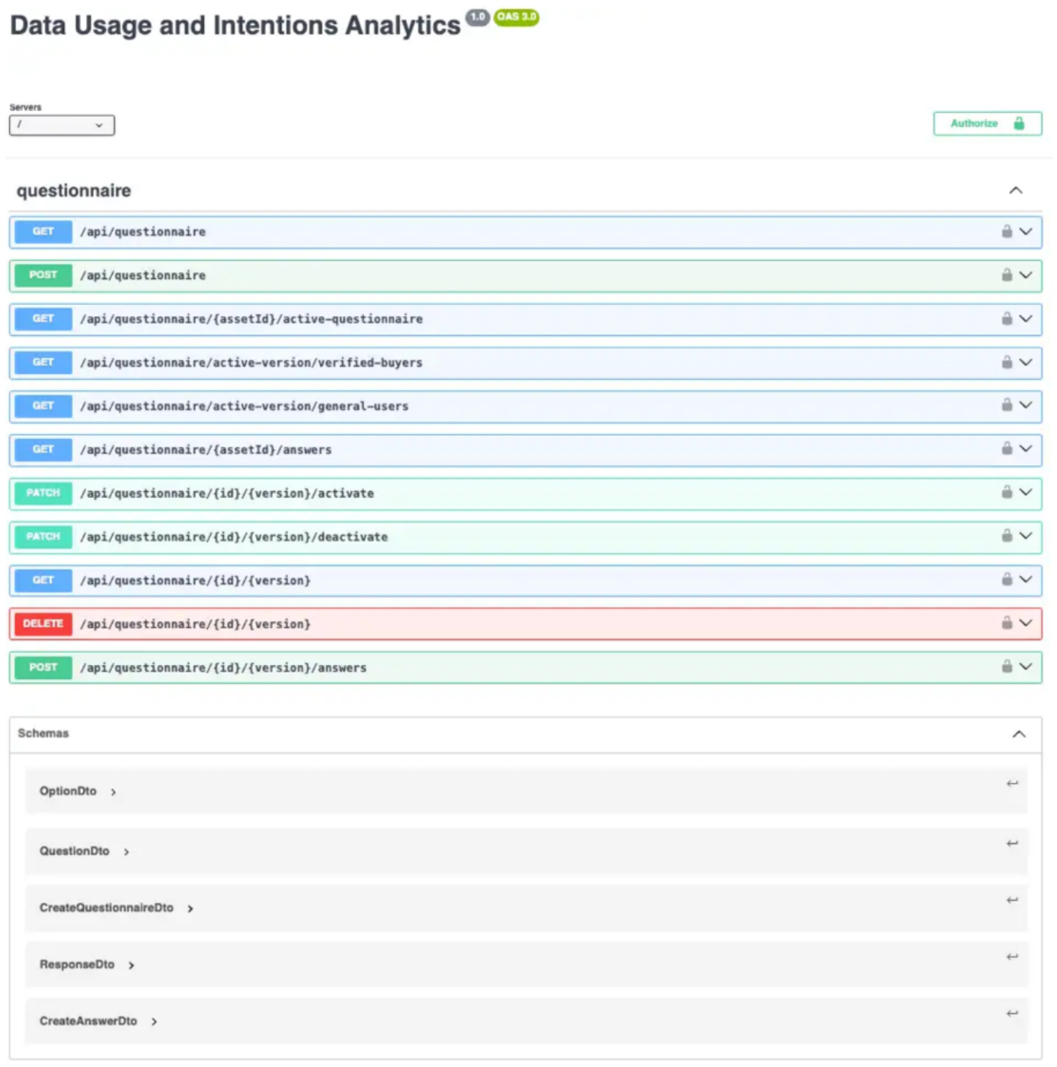
9.9.1.1 Components providing Input to the Data Usage and Intentions Analytics

Input Required	Component providing the Input	Communication Method
Selected dataset to be shown to the data provider	PISTIS Data Catalogue	API
Dataset's Metadata to be shown to the data provider	PISTIS Data Catalogue	API
Dataset's Lineage to be shown to the data provider	Data Ledger	API
Transaction of similar datasets to be shown to the data provider	Data Ledger	API

9.9.1.2 Components to which the Data Usage and Intentions Analytics provides input

Output Served	Component ingesting the output	Communication Method
Information regarding the derived data usage insights	FAIR Data Valuation Services	API

9.9.2 API Calls Descriptions


Data Usage and Intentions Analytics 1.0 OAS 3.0

questionnaire

```
GET     /api/questionnaire
POST    /api/questionnaire
GET     /api/questionnaire/{assetId}/active-questionnaire
GET     /api/questionnaire/active-version/verified-buyers
GET     /api/questionnaire/active-version/general-users
GET     /api/questionnaire/{assetId}/answers
PATCH   /api/questionnaire/{id}/{version}/activate
PATCH   /api/questionnaire/{id}/{version}/deactivate
GET     /api/questionnaire/{id}/{version}
DELETE  /api/questionnaire/{id}/{version}
POST    /api/questionnaire/{id}/{version}/answers
```

Schemas

- OptionDto
- QuestionDto
- CreateQuestionnaireDto
- ResponseDto
- CreateAnswerDto

9.9.3 SEQUENCE DIAGRAMS

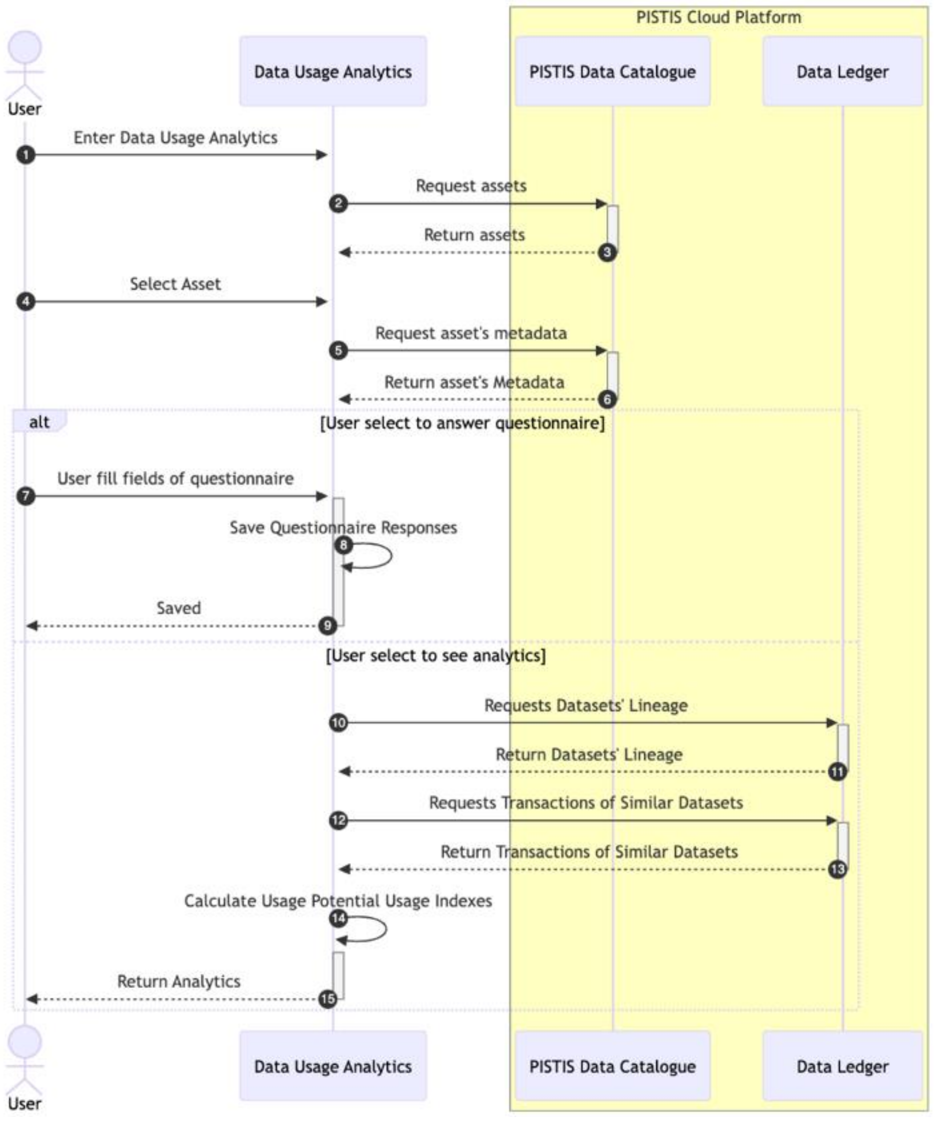

PISTIS Cloud Platform
User
Data Usage Analytics
PISTIS Data Catalogue
Data Ledger
1 Enter Data Usage Analytics
2 Request assets
3 Return assets
4 Select Asset
5 Request asset's metadata
6 Return asset's Metadata
alt
[User select to answer questionnaire]
7 User fill fields of questionnaire
8 Save Questionnaire Responses
9 Saved
[User select to see analytics]
10 Requests Datasets' Lineage
11 Return Datasets' Lineage
12 Requests Transactions of Similar Datasets
13 Return Transactions of Similar Datasets
14 Calculate Usage Potential Usage Indexes
15 Return Analytics

Figure 37: Data Usage and Intentions Analytics- Sequence Diagram

10 Security, Trust & Privacy Preservation Bundle

The Security, Trust & Privacy Preservation bundle offers services for strengthening data security and privacy.

This bundle consists of the following components:

- Anonymizer
- Lineage Tracker
- GDPR checker
- Searchable Encryption
- Encryption/Decryption Engine

These are presented in the following sub-sections.

10.1 Anonymizer

The anonymizer is a component responsible for preserving data privacy. It alters data in such a way that it will preserve its usefulness but hide the original data. With these modifications, it cannot be traced back to the individuals the data was taken from.

The anonymizer can take a dataset and obfuscating the contained data by replacing it with values that represent the original data in a way that is non-identifying (e.g., an age of 29 may be replaced with [20-30] or a name Darren Smith may be replaced with Darren *****). This is known as data masking.

Via the frontend interface, users will be able to configure the anonymisation process by selecting different anonymisation pre-sets, which can be applied to a column in their dataset, or they can use advanced settings to allow for more configurability in their anonymisation. To see how their choices will impact the result, a preview button is available. Upon clicking this button, users will see a subset of their dataset with their current anonymisation options applied to it. This will help users understand the impact of their choices.

The PseudoID generator is a smaller component, capable of producing a unique ID for a user who wishes to share their data as an anonymous user. This ID may then be used for the purposes of communication with the data owner whilst preserving their anonymity.

The Anonymizer also supports 'Location Privacy'. 'Location Privacy' is defined as "*the ability of an individual to move in public space with the expectation that under normal circumstances their location will not be systematically and secretly recorded for later use*."

The existence of location databases stripped of identifying tags can lead to information leaks. For instance, if I know that Vera is the sole resident of Dead End Lane, the data that someone used a location-based service on Dead End Lane can be reasonably linked to Vera.

Since location privacy definition and requirements differ depending on the scenario, no single technique can address the requirements of all location privacy categories. Therefore, in the past, the research community, focusing on providing solutions for the protection of location privacy of users, has defined techniques that can be divided into three main classes:

anonymity-based, obfuscation-based, and *policy-based* techniques. These classes of techniques are partially overlapped in scope and could be potentially suitable to cover requirements coming from one or more of the categories of location privacy.

It is easy to see that anonymity-based and obfuscation-based techniques can be considered dual categories. Anonymity-based techniques have been primarily defined to protect identity privacy and are not suitable for protecting position privacy, whereas obfuscation-based techniques are well suited for position protection and not appropriate for identity protection. Anonymity-based and obfuscation-based techniques could also be both exploited for protecting path privacy. Policy-based techniques are in general suitable for all location privacy categories, although they are often difficult to understand and manage for end users.

It is important to consider the notion of utility within the context of anonymizing location data. If the data seeker aims to understand the mobility patterns of different groups to inform, for example, public transport planning, accurate location data over time at scale is imperative. Therefore, supporting location privacy must also consider the impact on the utility of that data. It will affect the monetary value of the data if the necessary insights can no longer be reliably derived from it.

Location Privacy as a result is supported through a mechanism to generate new **Synthetic Data** that has the **same format and statistical properties** as the original location data.

Synthetic data can then be used to supplement, augment, and in some cases replace real data when training Machine Learning models. Additionally, it enables the testing of Machine Learning or other data dependent software systems without the risk of exposure that comes with data disclosure.

Finally, the anonymizer supports the use of differential privacy to allow for generalised insights to be derived from data in such a way that reverse engineering to re-identify individuals within a dataset is not possible.

10.1.1 Relation to Other Components

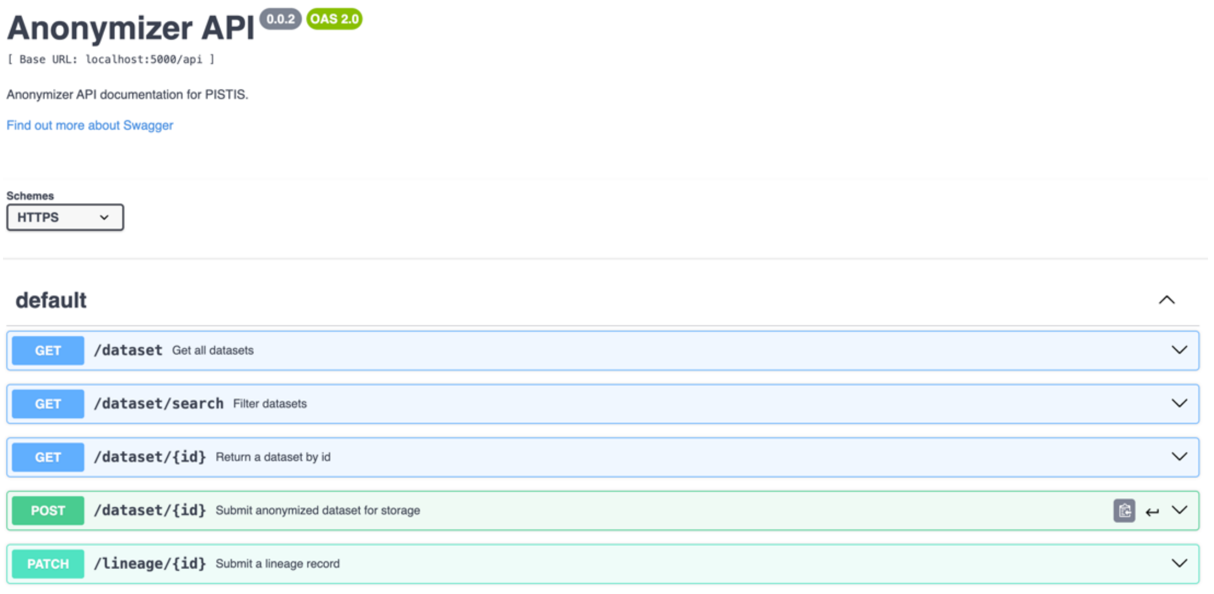
10.1.1.1 Components providing Input to Anonymizer

Input Required	Component providing the Input	Communication Method
Metadata	Factory Data Catalogue	API
Datasets	Factory Data Storage	API

10.1.1.2 Components to which Anonymizer provides input

Output Served	Component ingesting the output	Communication Method
Anonymised Datasets	Factory Data Storage	API
Updates to Metadata	Factory Data Catalogue	API
Indicators of operations carried out on Data	Lineage Tracker	API

10.1.2 API Calls Descriptions



Anonymizer API 0.0.2 OAS 2.0

[Base URL: localhost:5000/api]

Anonymizer API documentation for PISTIS.

Find out more about Swagger

Schemes: HTTPS

default

GET	`/dataset`	Get all datasets
GET	`/dataset/search`	Filter datasets
GET	`/dataset/{id}`	Return a dataset by id
POST	`/dataset/{id}`	Submit anonymized dataset for storage
PATCH	`/lineage/{id}`	Submit a lineage record

10.1.3 Sequence Diagrams

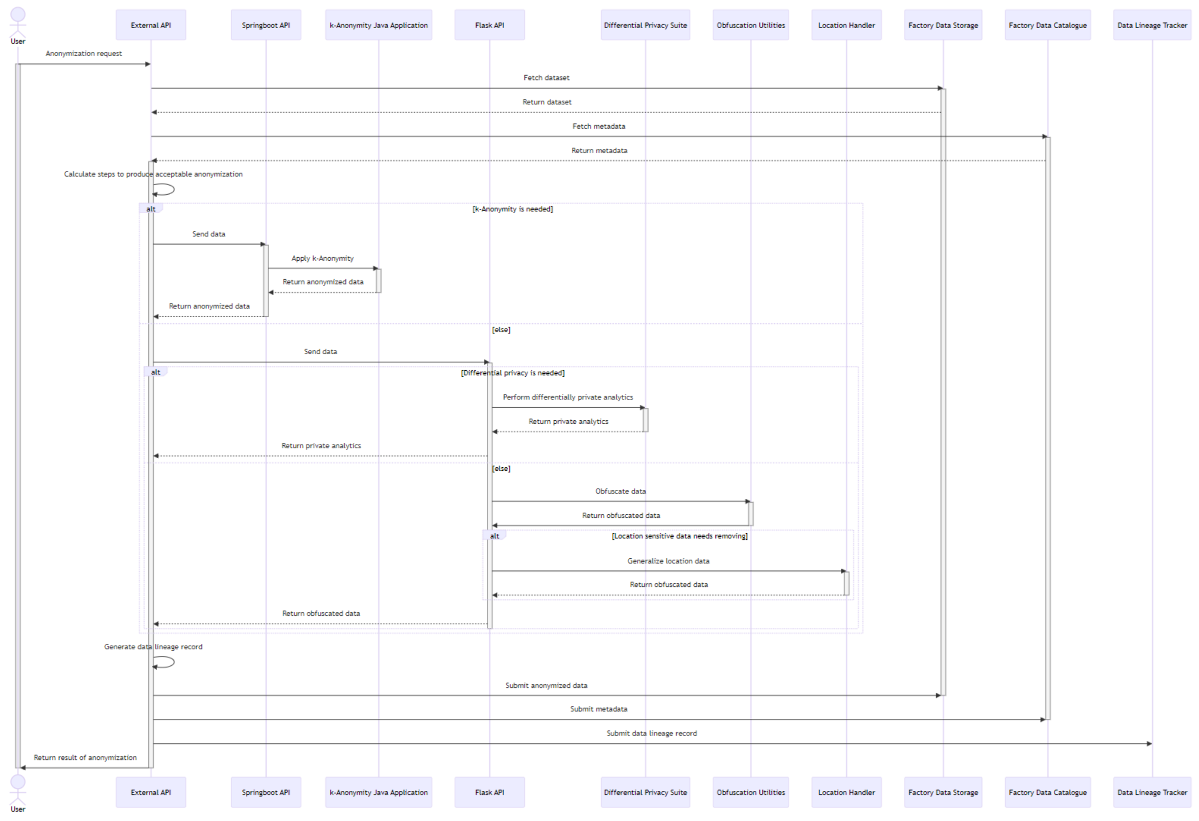


Figure 38: Anonymiser Sequence Diagram

10.2 Lineage Tracker

The Lineage Tracker documents the operations performed on a data asset, including details about who performed the operation and when it was conducted. In doing so, it constructs the asset's lineage, representing its version history and providing transparency to the user, giving them access to previous versions.

The Lineage Tracker consists of a Provenance Engine, which exposes an API offering endpoints to document the operations performed on a specific asset as well as to receive the respective provenance information in structured format. Subsequently, this information is converted into RDF-format (Linked Data) according to the W3C-published PROV-Ontology and saved in a triple store, where the lineage graph associated with each asset is being continuously extended with every operation.

In addition, it will offer a UI visualizing the data lineage, giving the user direct access to previous versions.

10.2.1 Relation to Other Components

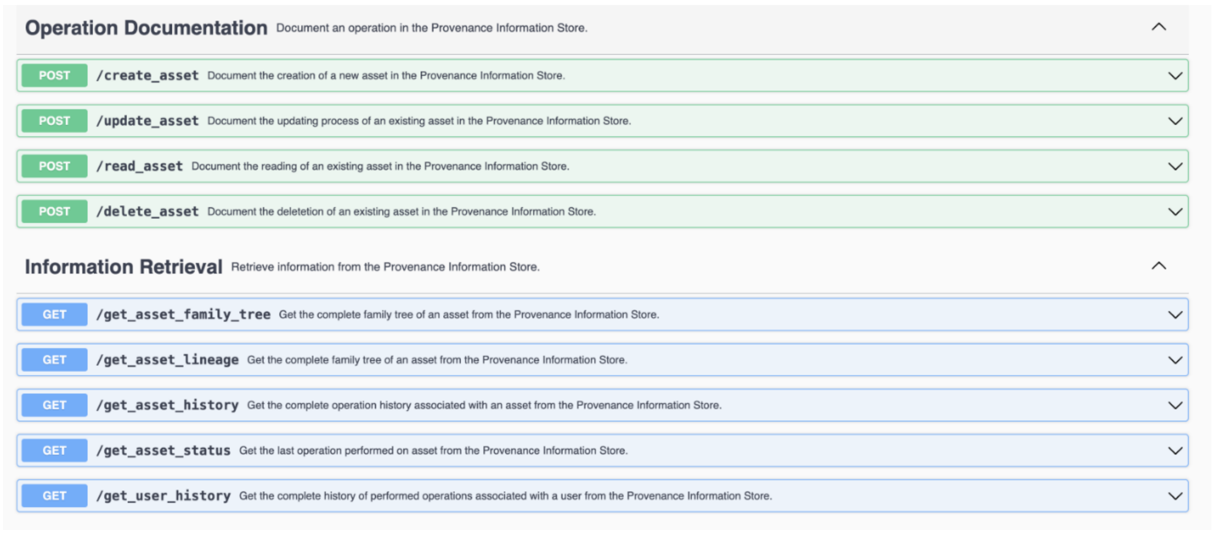
10.2.1.1 Components providing Input to the Lineage Tracker

Input Required	Component providing the Input	Communication Method
Who performed which action on which version of which dataset when?	Data Factory Storage	API

10.2.1.2 Components to which the Lineage Tracker provides input

Output Served	Component ingesting the output	Communication Method
Dataset specific lineage information (graph structure) in structured format (json)	Analytics Engine, Data Quality Assessment	API
UUID and version of dataset in case it was updated	Data Ledger	API

10.2.2 API Calls Descriptions



Operation Documentation Document an operation in the Provenance Information Store.

POST /create_asset Document the creation of a new asset in the Provenance Information Store.

POST /update_asset Document the updating process of an existing asset in the Provenance Information Store.

POST /read_asset Document the reading of an existing asset in the Provenance Information Store.

POST /delete_asset Document the deletetion of an existing asset in the Provenance Information Store.

Information Retrieval Retrieve information from the Provenance Information Store.

GET /get_asset_family_tree Get the complete family tree of an asset from the Provenance Information Store.

GET /get_asset_lineage Get the complete family tree of an asset from the Provenance Information Store.

GET /get_asset_history Get the complete operation history associated with an asset from the Provenance Information Store.

GET /get_asset_status Get the last operation performed on asset from the Provenance Information Store.

GET /get_user_history Get the complete history of performed operations associated with a user from the Provenance Information Store.

10.2.3 Sequence Diagrams

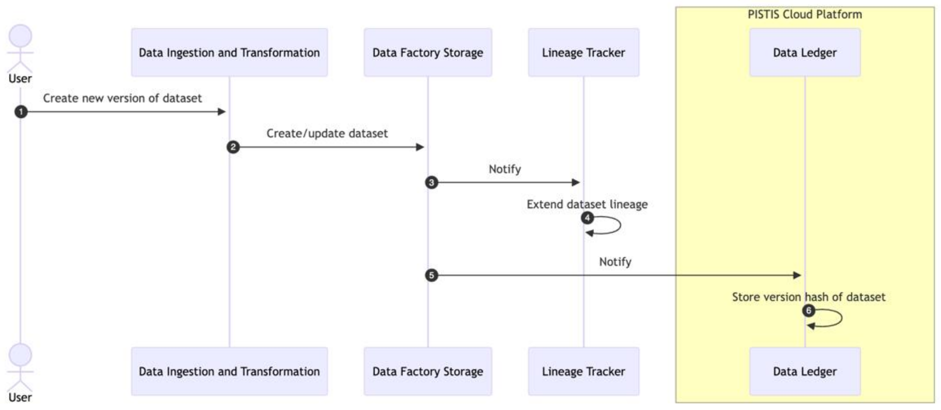


Figure 39: Lineage Tracker Sequence Diagram #1

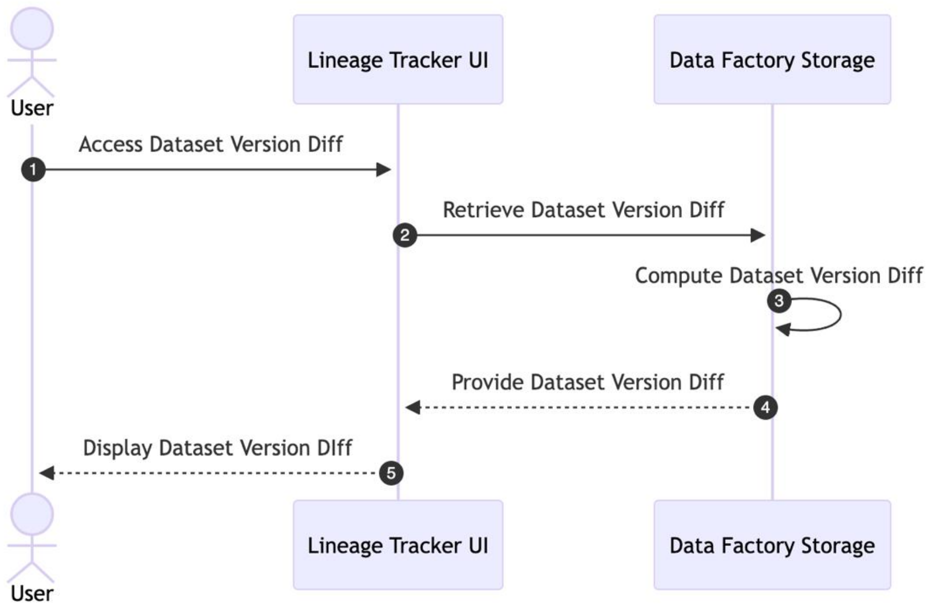

User
Lineage Tracker UI
Data Factory Storage
1 Access Dataset Version Diff
2 Retrieve Dataset Version Diff
3 Compute Dataset Version Diff
4 Provide Dataset Version Diff
5 Display Dataset Version DIff
User
Lineage Tracker UI
Data Factory Storage

Figure 40: Lineage Tracker Sequence Diagram #2

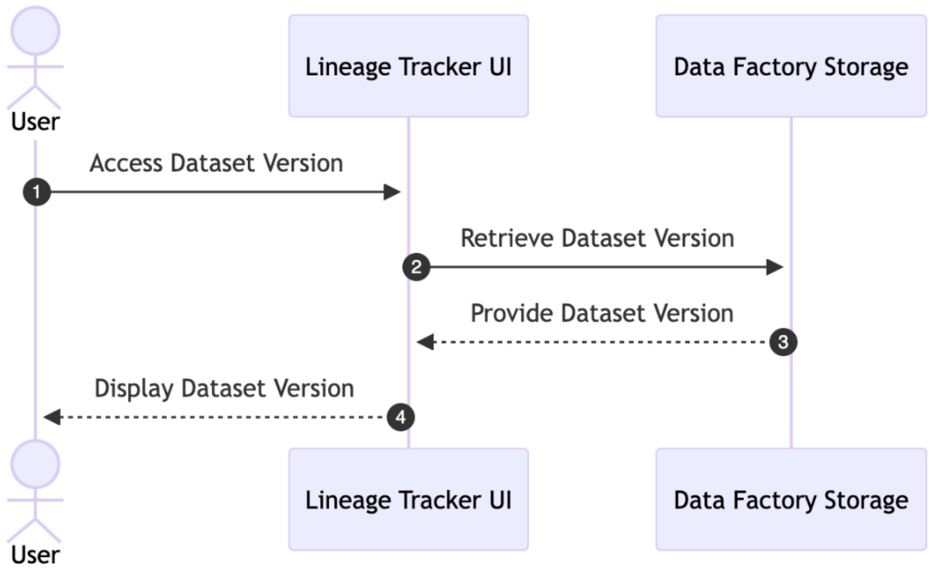


Figure 41: Lineage Tracker Sequence Diagram #3

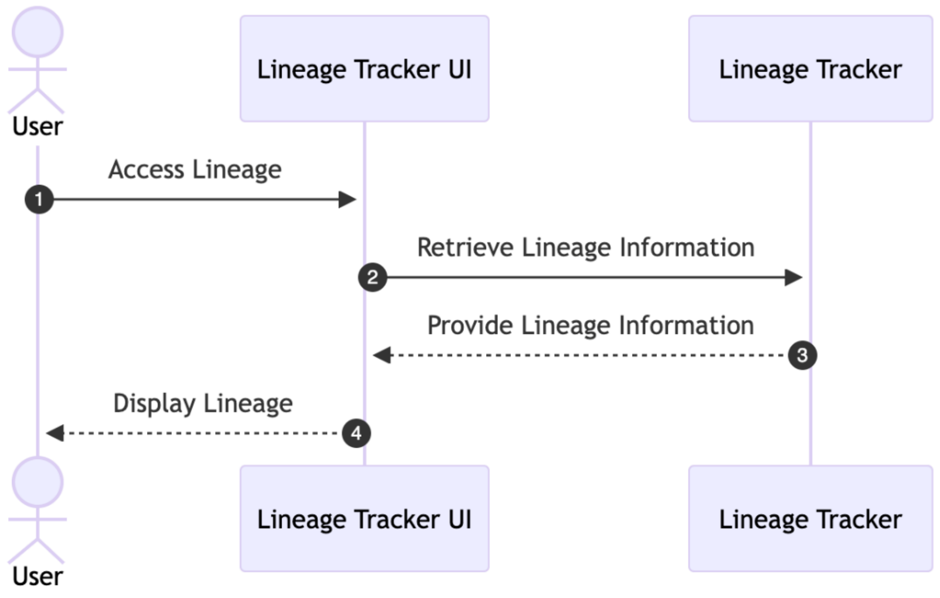


Figure 42: Lineage Tracker Sequence Diagram #4

10.3 GDPR CHECKER

Figure 43

The GDPR Checker is a rule-based engine designed to enforce appropriate rules for ensuring compliance with the EU General Data Protection Regulation (GDPR). It acts as a **conformance safeguard for users**, preventing any leakage of personally identifiable information (PII). The GDPR rules enforced by the GDPR Checker are explicitly derived from the legal partners of the PISTIS consortium, and this collaboration is essential for maintaining compliance.

GDPR Checker is a rule enforcer and nothing more. It is configurable and capable of dynamically incorporating rules. However, since the official GDPR rules have not been provided yet, the GDPR Checker will operate with simplified rules created internally for the Alpha version. These rules do not represent the authentic GDPR regulations.

The GDPR Checker evaluates a user’s dataset to determine its compliance status by applying the defined rules from the legal partners. If a dataset is found non-compliant, the GDPR Checker returns recommendations to the user, highlighting the specific rules that have been violated and suggesting potential actions, such as utilising the Anonymizer component before further processing.

The GDPR checking process is completely optional; it is at the user's discretion to decide whether to assess their dataset for GDPR violations. The legal department must also supply privacy assurance levels based on data type. For example, healthcare-related data will adhere to stricter privacy policies compared to more mainstream data, such as sports. Consequently, the GDPR Checker will apply the relevant subset of rules depending on the data classification.

Figure 38 presents the high-level architecture of the GDPR Checker including the internal components of the tool.

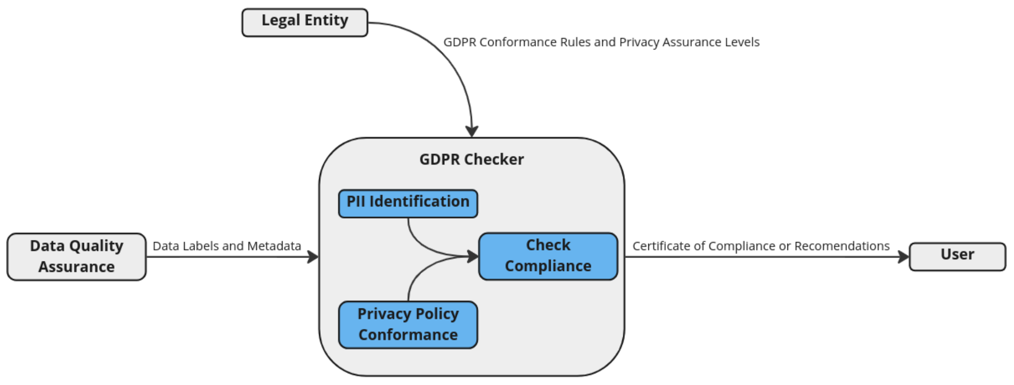

Legal Entity
GDPR Conformance Rules and Privacy Assurance Levels
GDPR Checker
PII Identification
Check Compliance
Privacy Policy Conformance
Data Quality Assurance
Data Labels and Metadata
Certificate of Compliance or Recomendations
User

Figure 43: GDPR Checker high-level Architecture

10.3.1 Relation to Other Components

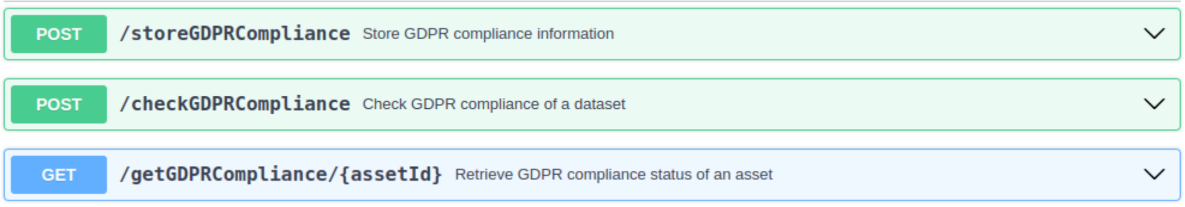
10.3.1.1 Components providing Input to GDPR checker component

Input Required	Component providing the Input	Communication Method
Data labels and metadata	Data Quality Assessment	API
Metadata	Data Catalogue	API
Data	Data Storage	API
GDPR Conformance Rules and Privacy Assurance Levels	PISTS Legal entity	API

10.3.1.2 Components to which GDPR checker component provides input

Output Served	Component ingesting the output	Communication Method
GDPR checker report (compliance status or list of static pre-defined possible mitigation policies/ recommendations)	Data Factory Catalogue	API
GDPR Compiance Report	Public Ledger	API

10.3.2 API Calls Descriptions


POST /storeGDPRCompliance Store GDPR compliance information

POST /checkGDPRCompliance Check GDPR compliance of a dataset

GET /getGDPRCompliance/{assetId} Retrieve GDPR compliance status of an asset

10.3.3 Sequence Diagrams

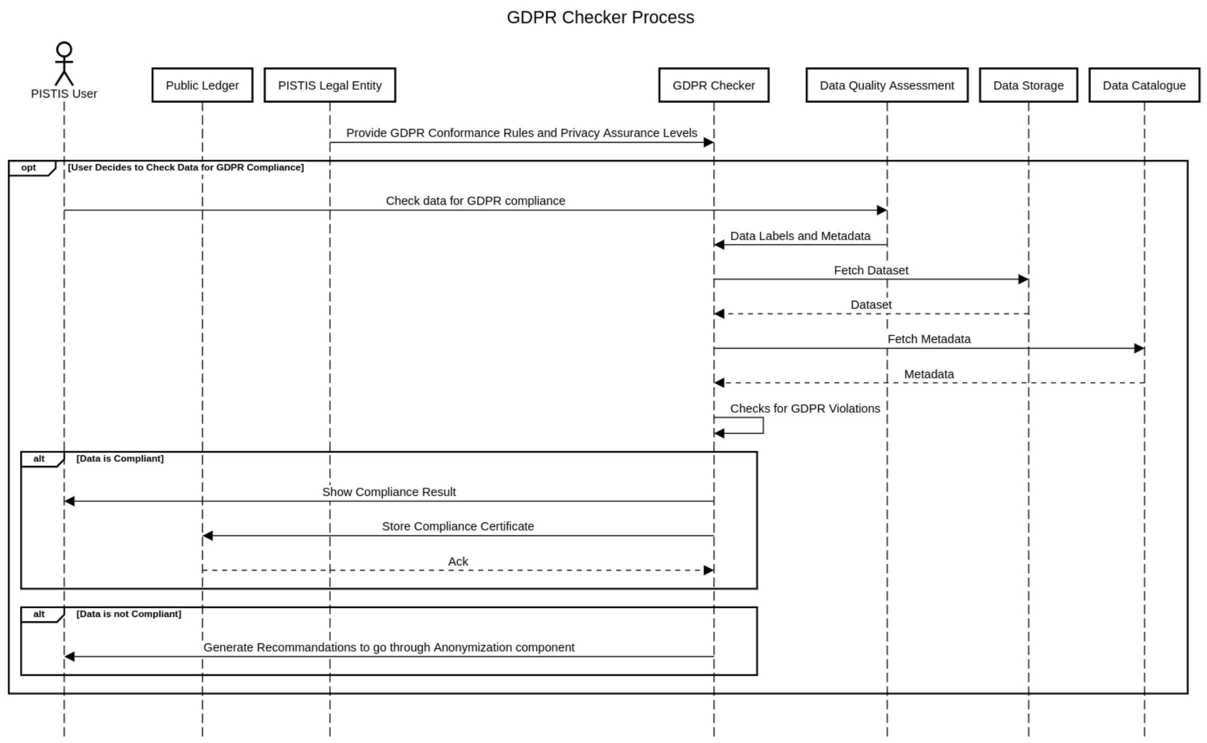


Figure 44: GDPR checker Sequence Diagram

10.4 Searchable Encryption

The Searchable Encryption (SE) component is a fundamental aspect of the PISTIS platform and enables data users to search over the encrypted data (or indexes to data). In other words, it allows users to perform searches on data that have been encrypted, ensuring that sensitive information is protected from unauthorised access. Indeed, SE not only protects the data privacy of data owners but also enables data users to search over the encrypted data. However, it assumes that the user sending the search query owns the decryption key and that the sender must know the identity of the user querying the data to encrypt using the corresponding encryption key. This raises the question of how to achieve this, given that the encrypted data is shared among several receivers. To address this, we can draw inspiration from attribute-based encryption (ABE), wherein only participants possessing the requisite permissions, and the corresponding ABE (matching) attribute private keys can decrypt and view the encrypted data (indexes). In concrete terms, the secret decryption key for the encrypted indexes is intricately linked to a specific set of attributes. This connection implies that if there exists a subset of attributes containing at least t elements, which corresponds to the set associated with the secret keys, then utilizing the secret keys becomes possible for decryption purposes.

In the context of PISTIS, the SE component will adopt a dynamic searchable encryption (DSE) scheme over encrypted indexes stored in the Blockchain and the Factory Data Storage. In concrete terms, the secret decryption key for the encrypted indexes is intricately linked to a specific set of attributes. This connection implies that if there exists a subset of attributes containing at least t elements, which corresponds to the set associated with the secret keys, then utilizing the secret keys becomes possible for decryption purposes. On top of that, the mapping between the keywords and the encrypted indexes will be stored in the Blockchain. Figure 45 presents the high-level architecture of the Searchable Encryption including the internal components of the tool. Such a component represents a significant advancement in secure data handling. It exemplifies how modern encryption techniques can be harmonised with emerging technologies like blockchain to create a secure, transparent, and efficient data management system. This component is pivotal in enabling the PISTIS platform to handle sensitive data with the utmost security while ensuring that the data remains accessible and useful for authorised users.

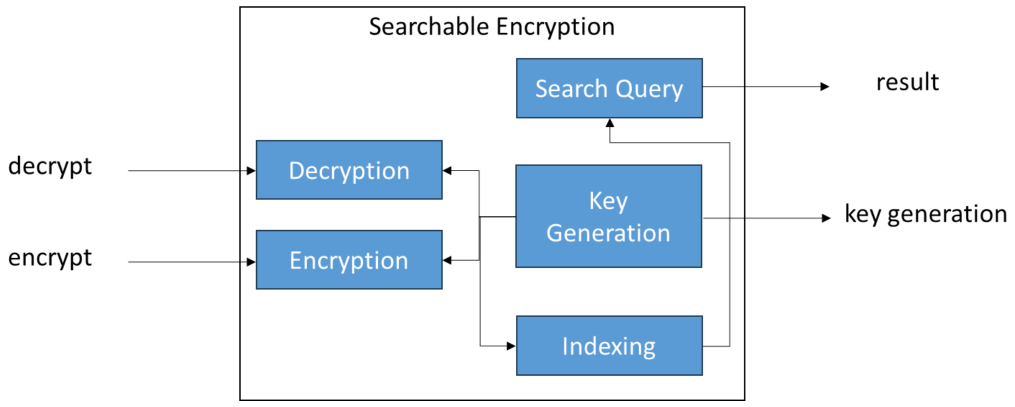


Figure 45: Searchable Encryption high-level Architecture

10.4.1 Relation to Other Components

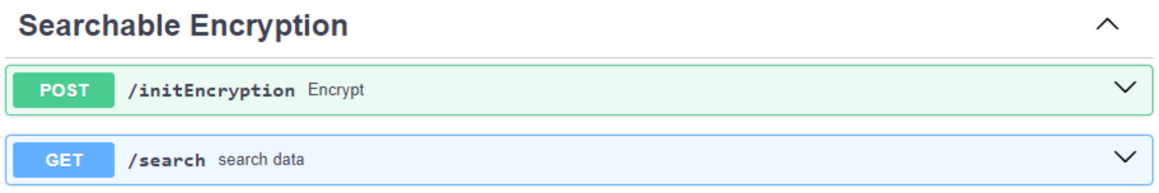
10.4.1.1 Components providing Input to Searchable Encryption component

Input Required	Component providing the Input	Communication Method
Data to be encrypted and indexed for supporting searchable encryption	Job Configurator	API
Encrypted Keyword (Index)	Factory Data Storage	API

10.4.1.2 Components to which Searchable Encryption component provides input

Output Served	Component ingesting the output	Communication Method
Search results based on encrypted indexes	Distributed Query Engine	API
Encrypted Keyword and Transaction ID to store	Factory Data Storage	API

10.4.2 API Calls Descriptions



Searchable Encryption

POST /initEncryption Encrypt

GET /search search data

10.4.3 Sequence Diagrams

Figure 46 depicts the sequence of actions for the Searchable Encryption component.

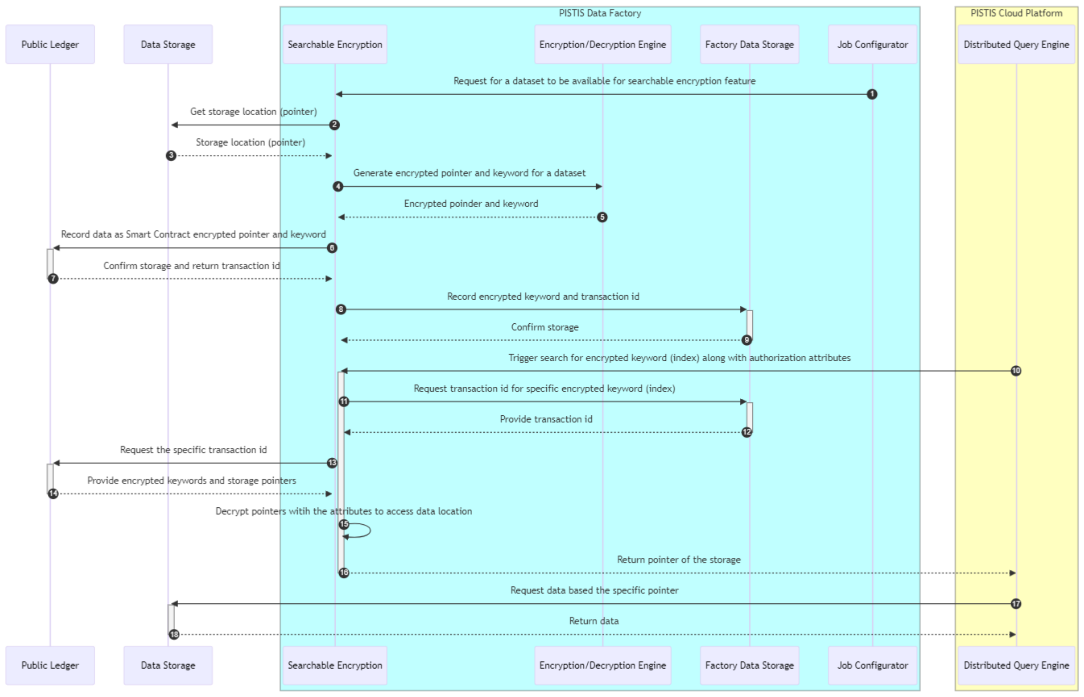

PISTIS Data Factory
PISTIS Cloud Platform
Public Ledger
Data Storage
Searchable Encryption
Encryption/Decryption Engine
Factory Data Storage
Job Configurator
Distributed Query Engine
Request for a dataset to be available for searchable encryption feature
Get storage location (pointer)
Storage location (pointer)
Generate encrypted pointer and keyword for a dataset
Encrypted poinder and keyword
Record data as Smart Contract encrypted pointer and keyword
Confirm storage and return transaction id
Record encrypted keyword and transaction id
Confirm storage
Trigger search for encrypted keyword (index) along with authorization attributes
Request transaction id for specific encrypted keyword (index)
Provide transaction id
Request the specific transaction id
Provide encrypted keywords and storage pointers
Decrypt pointers witih the attributes to access data location
Return pointer of the storage
Request data based the specific pointer
Return data

Figure 46: Searchable Encryption Sequence Diagram

10.5 Encryption/Decryption Engine

The Encryption/Decryption engine within the PISTIS platform is a comprehensive and versatile tool designed to handle a wide array of encryption and decryption needs. This component not only supports standard encryption methodologies, such as symmetric and asymmetric encryption, but also extends its capabilities to ensure robust data protection and privacy across various aspects of the platform.

Moreover, in the context of the Self-Sovereign Identity (SSI) concept the key for encryption/decryption is associated with the SSI of the user. Figure 47 represents the high-level architecture of the Encryption/Decryption component including the internal components of the tool.

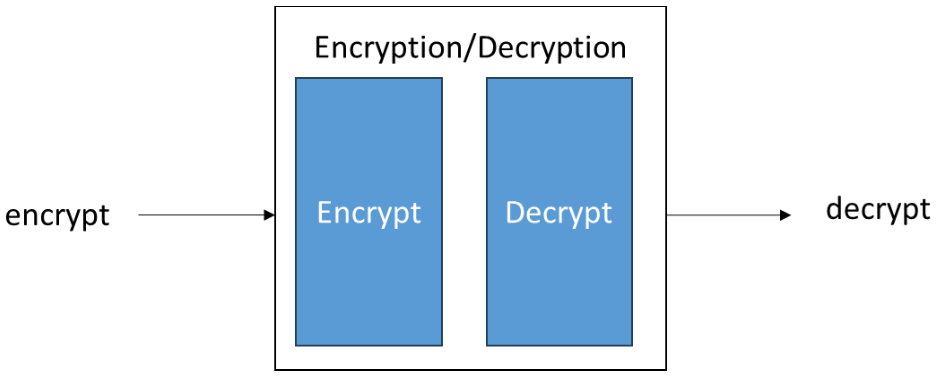


Figure 47: Encryption/Decryption high-level Architecture

10.5.1 Relation to Other Components

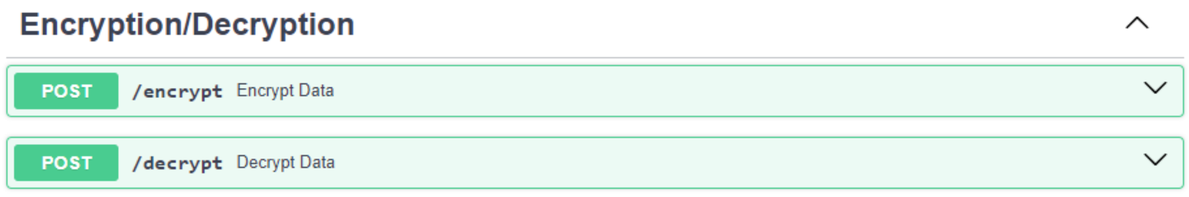
10.5.1.1 Components providing Input to Encryption/Decryption component

Input Required	Component providing the Input	Communication Method
Credentials for Encryption/Decryption	Data Factory User Wallet	Internal function (offered by Data Factory User Wallet)
Dataset in a file format	Job Configurator	API

10.5.1.2 Components to which Encryption/Decryption component provides input

Output Served	Component ingesting the output	Communication Method
Credentials for Key Decryption Release	Data Factory User Wallet	Internal function (offered by Data Factory User Wallet)
Encrypted/Decrypted Dataset in a file format	Job Configurator	API

10.5.2 API Calls Descriptions


Encryption/Decryption

POST /encrypt Encrypt Data

POST /decrypt Decrypt Data

10.5.3 Sequence Diagrams

In this sequence diagram, the process begins with the Job Configurator initiating a request to the Data Storage to retrieve a dataset associated with a specific Asset ID. The Data Storage responds by returning the dataset in a file format. The Job Configurator then forwards this dataset, along with the user's JWT that was obtained from the session to the Encryption/Decryption Engine for processing, either encrypting or decrypting the data as needed. Once the Encryption/Decryption Engine completes the process, it returns a newly encrypted or decrypted file back to the Job Configurator. In the second release, the Encryption/Decryption will communicate with the Identity Wallet, to get the user's proof ID. The Job Configurator proceeds to upload this new file to the Data Storage. Finally, the Job Configurator requests the Data Storage to delete the old unencrypted or previous version of the dataset and generate a new Asset ID, completing the workflow.

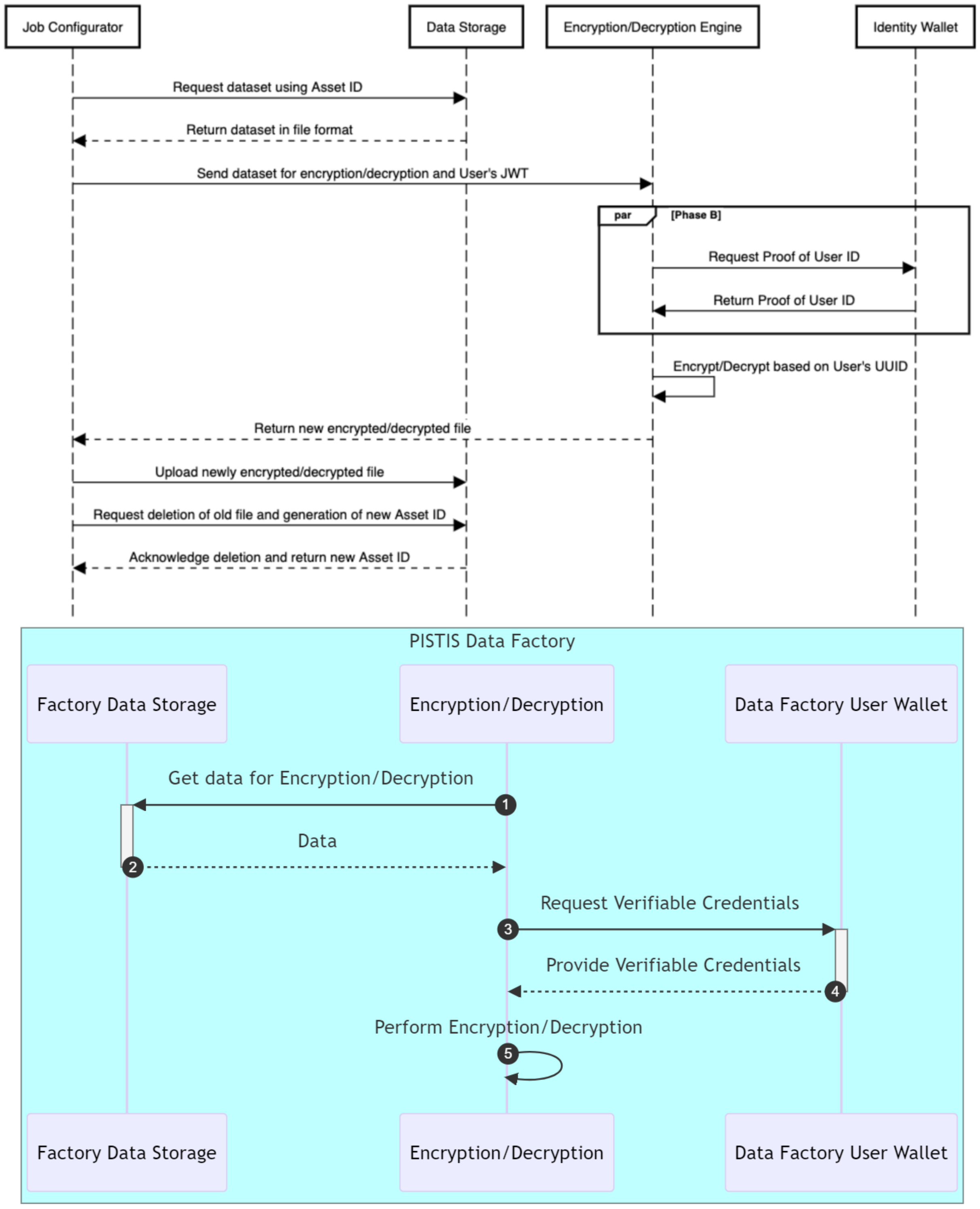

Job Configurator
Data Storage
Encryption/Decryption Engine
Identity Wallet
Request dataset using Asset ID
Return dataset in file format
Send dataset for encryption/decryption and User's JWT
par
[Phase B]
Request Proof of User ID
Return Proof of User ID
Encrypt/Decrypt based on User's UUID
Return new encrypted/decrypted file
Upload newly encrypted/decrypted file
Request deletion of old file and generation of new Asset ID
Acknowledge deletion and return new Asset ID
PISTIS Data Factory
Factory Data Storage
Encryption/Decryption
Data Factory User Wallet
Get data for Encryption/Decryption
1
Data
2
Request Verifiable Credentials
3
Provide Verifiable Credentials
4
Perform Encryption/Decryption
5
Factory Data Storage
Encryption/Decryption
Data Factory User Wallet

Figure 48: Encrypt/Decrypt Sequence Diagram

11 TRANSACTIONS SERVICES BUNDLE

The Transaction Services bundle provides all the necessary functionalities to author, execute, and validate the different transactions with the use of smart contracts.

This bundle consists of the following components:

- On/Off Platform Contract Inspector
- Transaction Auditor
- Smart Contract Template Composer
- Smart Contract Execution Engine
- Data Factory User Wallet

These are presented in the following sub-sections.

11.1 ON/OFF PLATFORM CONTRACT INSPECTOR

The "on" part of the on/off-platform contract inspector operates as an integral branch of the lineage tracking system. This inspector leverages the information stored in the blockchain regarding the usage and ownership of the dataset to fulfil its core functions. It facilitates the retrieval of usage tracking information, active contracts, license terms. Additionally, it plays a crucial role in determining whether there is a violation of the contract's license, helping to ensure the adherence to contract agreements and maintain data integrity within the platform.

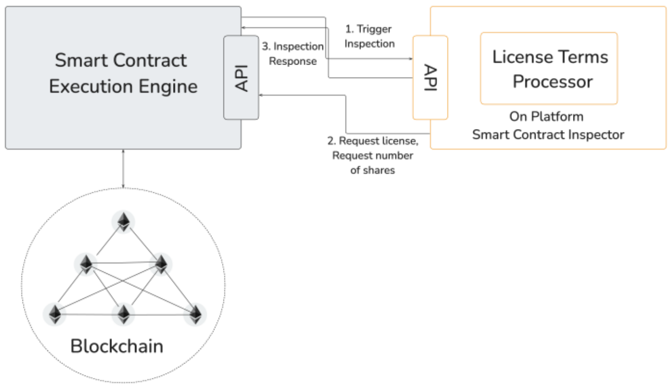

Smart Contract Execution Engine
API
3. Inspection Response
1. Trigger Inspection
API
License Terms Processor
On Platform Smart Contract Inspector
2. Request license, Request number of shares
Blockchain

Figure 49: On chain platform inspection

The off-chain contract inspector serves as a component designed to identify duplicate entries within existing datasets through a process known as fingerprinting. Unlike its on-platform counterpart, the off-chain inspector operates externally to the primary data platform during data check in. Its primary objective is to scan and compare dataset entries, analysing their unique fingerprints or cryptographic hashes to detect any identical or closely matching records. By doing so, it helps maintain data quality and integrity by preventing the unintended duplication of data, ensuring that datasets remain accurate and free from redundant information, which is crucial for the proper PISTIS platform operation.

11.1.1 Relation to Other Components

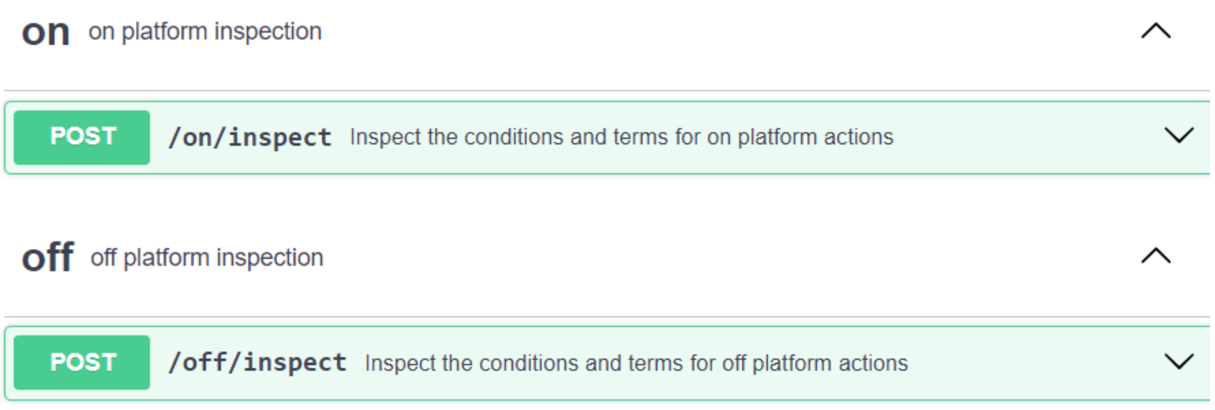
11.1.1.1 Components providing Input to On/Off Platform Contract Inspector

Input Required	Component providing the Input	Communication Method
License	Smart Contract Execution Engine	API
Data format, data location	Data Check-In	API
Transaction Details	Smart Contract Execution Engine	API

11.1.1.2 Components to which On/Off Platform Contract Inspector provides input

Output Served	Component ingesting the output	Communication Method
True/false for compliance with terms, and with term is violated	Smart Contract Execution Engine	API
Similarity value with existing dataset, the existing dataset	PISTIS Data Catalogue	API

11.1.2 API Calls Descriptions


on on platform inspection

POST `/on/inspect` Inspect the conditions and terms for on platform actions

off off platform inspection

POST `/off/inspect` Inspect the conditions and terms for off platform actions

11.1.3 Sequence Diagrams

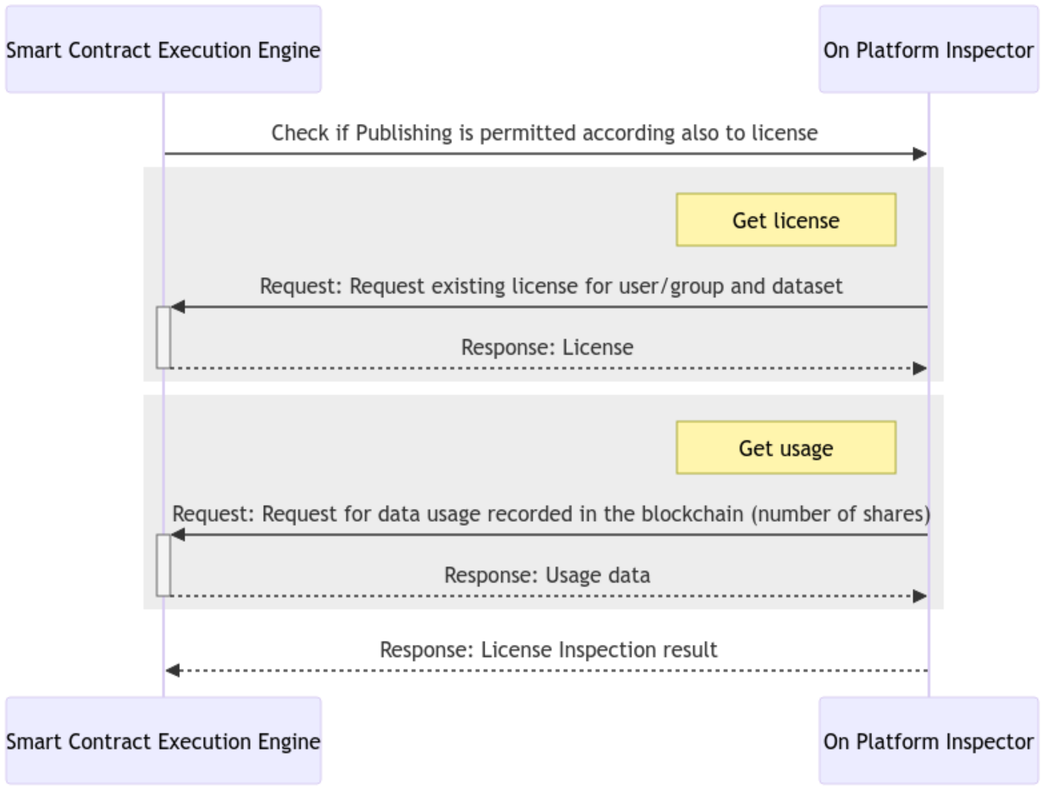

Smart Contract Execution Engine
On Platform Inspector
Check if Publishing is permitted according also to license
Get license
Request: Request existing license for user/group and dataset
Response: License
Get usage
Request: Request for data usage recorded in the blockchain (number of shares)
Response: Usage data
Response: License Inspection result
Smart Contract Execution Engine
On Platform Inspector

Figure 50: On platform contract inspection

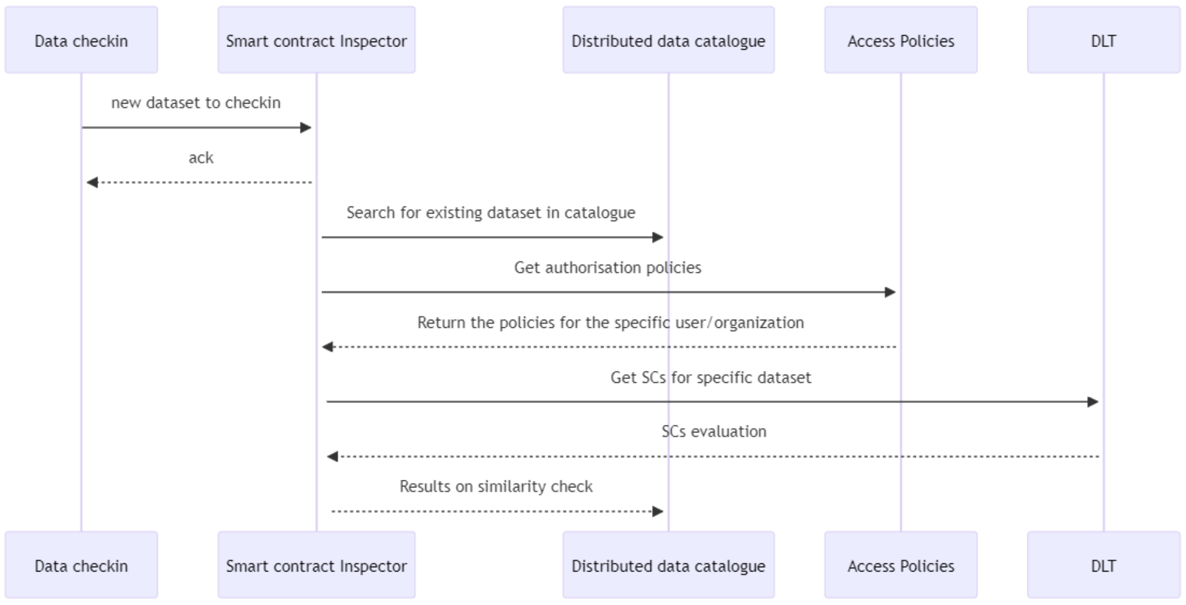

Data checkin
Smart contract Inspector
Distributed data catalogue
Access Policies
DLT
new dataset to checkin
ack
Search for existing dataset in catalogue
Get authorisation policies
Return the policies for the specific user/organization
Get SCs for specific dataset
SCs evaluation
Results on similarity check
Data checkin
Smart contract Inspector
Distributed data catalogue
Access Policies
DLT

Figure 51: Off platform contract inspection

11.2 Transaction Auditor

The Transaction Auditor is responsible for monitoring the validity of the transactions in PISTIS. It enables users to select a transaction from the PISTIS ledgers and accompanies it with metadata requested and retrieved from the PISTIS Data Catalogue. In addition, the Transaction Auditor retrieves monetary details from the Monetary Ledger.

By analysing the aforementioned information about the selected transaction, the PISTIS Auditor evaluates the transaction and stores the evaluation result through a smart contract in the PISTIS ledgers that can be then used as a validation certificate for the transaction.

11.2.1 Relation to Other Components

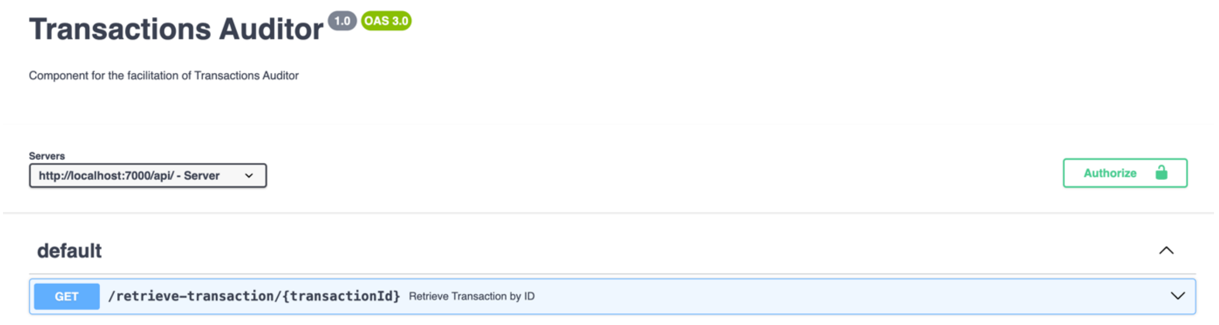
11.2.1.1 Components providing Input to Transaction Auditor

Input Required	Component providing the Input	Communication Method
Transaction ID and dataset details	PISTIS Data Catalogue	API
Dataset metadata	Data Ledger	API
Monetary details of transaction	Monetary Ledger	API

11.2.1.2 Components to which Transaction Auditor provides input

Output Served	Component ingesting the output	Communication Method
Request for transaction ID and dataset details	PISTIS Data Catalogue	API
Request for dataset metadata	Data Ledger	API
Request for monetary details of transaction	Monetary Ledger	API

11.2.2 API Calls Descriptions



Component for the facilitation of Transactions Auditor

Servers
http://localhost:7000/api/ - Server

GET /retrieve-transaction/{transactionId} Retrieve Transaction by ID

11.2.3 Sequence Diagrams

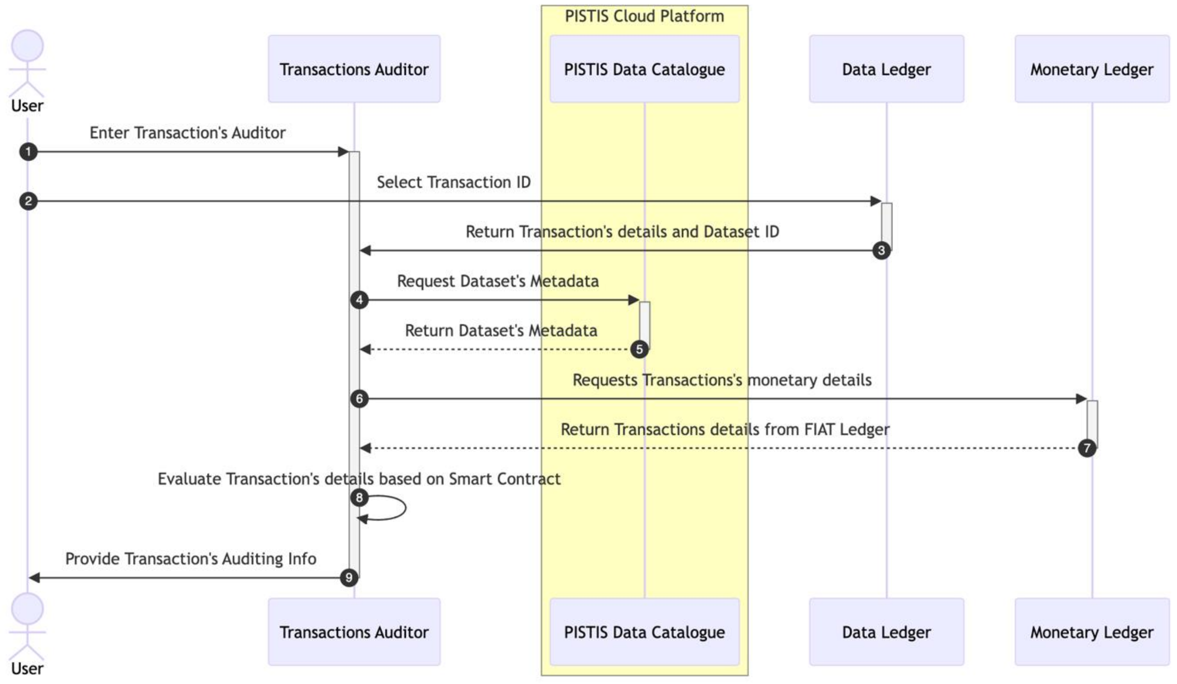

PISTIS Cloud Platform
User
Transactions Auditor
PISTIS Data Catalogue
Data Ledger
Monetary Ledger
1 Enter Transaction's Auditor
2 Select Transaction ID
3 Return Transaction's details and Dataset ID
4 Request Dataset's Metadata
5 Return Dataset's Metadata
6 Requests Transactions's monetary details
7 Return Transactions details from FIAT Ledger
8 Evaluate Transaction's details based on Smart Contract
9 Provide Transaction's Auditing Info

Figure 52: Transactions Auditor Sequence Diagram

11.3 Smart Contract Template Composer

The Smart Contract Template Composer overall purpose is to allow the Asset Description Bundler to select the appropriate templates and transform the readily available smart contract templates into the corresponding code, so it can be then sent to the blockchain, triggering the execution of the associated transactions.

As such, this composer's responsibility lies in delivering a ready-to-execute contract (by selecting and filling in pre-designed contract templates) that can capture the details of the agreements pertaining to the exchange of datasets. This contract will be presented to the Data Provider prior to finalizing the publication of a dataset in the PISTIS Data Catalogue.

Additionally, the Smart Contract Template Composer empowers data providers to personalise these templates, drafting smart contracts aligned to their specific requirements.

11.3.1 Relation to Other Components

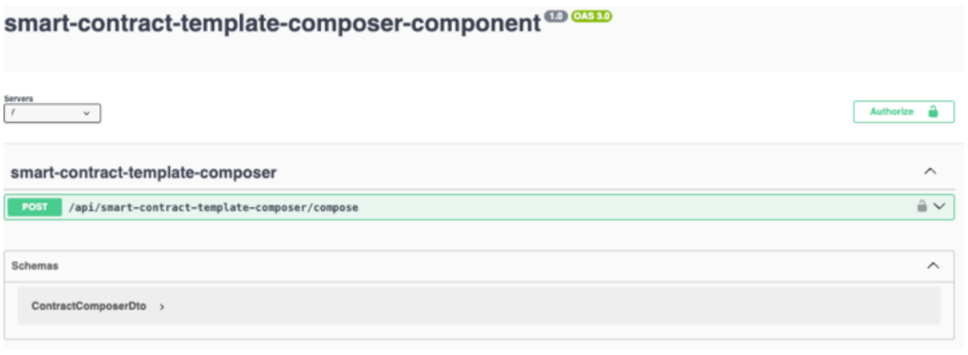
11.3.1.1 Components providing Input to the Smart Contract Template Composer

Input Required	Component providing the Input	Communication Method
Request for new contract	Asset Description Bundler	API

11.3.1.2 Components to which the Smart Contract Template Composer provides input

Output Served	Component ingesting the output	Communication Method
New contract template to be deployed	Asset Description bundler	API

11.3.2 API Calls Descriptions


smart-contract-template-composer-component 1.0 OAS 3.0
Servers
/
Authorize
smart-contract-template-composer
POST /api/smart-contract-template-composer/compose
Schemas
ContractComposerDto

11.3.3 Sequence Diagrams

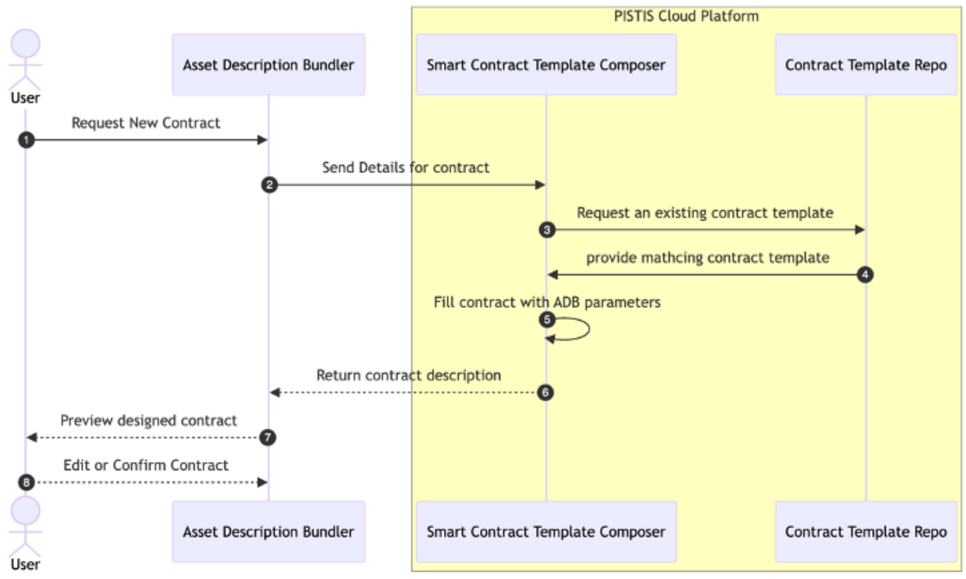

PISTIS Cloud Platform
User
Asset Description Bundler
Smart Contract Template Composer
Contract Template Repo
1 Request New Contract
2 Send Details for contract
3 Request an existing contract template
4 provide mathcing contract template
5 Fill contract with ADB parameters
6 Return contract description
7 Preview designed contract
8 Edit or Confirm Contract

Figure 53: Smart Contract Template Composer - Sequence Diagram

11.4 Smart Contract Execution Engine

The Smart Contract Execution Engine component, as its name suggests, is responsible for the actual execution of the smart contracts. Smart Contract Execution Engine executes all the stored and valid data trading chain code and makes the data artefacts available to the demand side. It has two modes of operation; one as part of the Data Exchange Preparation bundle where its functionality supports the PISTIS Data Factory environment, and one as part of the

Data Exchange Governance bundle where its functionality supports the PISTIS Cloud Platform. This component is responsible for the initiation of authorisation and observability operations (e.g., triggering On/Off platform or the Smart Contract Checker) prior to any contract execution to check the validity and the status of trading /sharing actions in a secure and proper manner. Figure 54 presents the high-level architecture of the Smart Contract Execution Engine including the internal components of the tool.

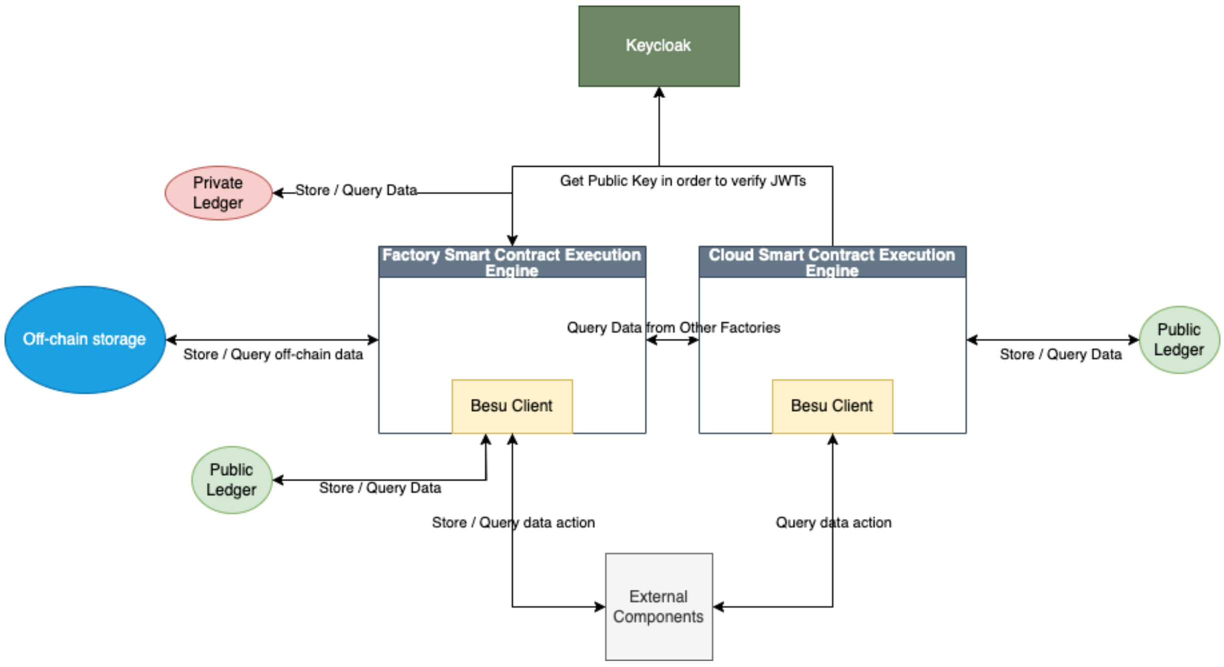

Keycloak
Private Ledger
Store / Query Data
Get Public Key in order to verify JWTs
Factory Smart Contract Execution Engine
Cloud Smart Contract Execution Engine
Off-chain storage
Store / Query off-chain data
Query Data from Other Factories
Store / Query Data
Public Ledger
Besu Client
Besu Client
Public Ledger
Store / Query Data
Store / Query data action
Query data action
External Components

Figure 54: Smart Contract Execution Engine High-Level Architecture

11.4.1 Relation to Other Components

11.4.1.1 Components providing Input to Smart Contract Execution Engine component

Input Required	Component providing the Input	Communication Method
Organisation Boundaries		
Check authenticity (get VC attributes)	Data Factory User Wallet	Internal function (offered by Data Factory User Wallet)
Data from the Smart Contracts of the Private Ledger	PISTIS Data Ledger	Internal function (offered by PISTIS Data Ledger)
Validation Confirmation or Violation Errors	Smart Contract Checker	API
Public Keys	KeyCLoak	API
GDPR Reports	GDPR Checker	API
Get Users Address from different factory	Cloud Platform SCEE	API
PISTIS Cloud Platform		
Check authenticity (get VC attributes)	Digital DLT-Fiat Wallet	Internal function

Data from the Smart Contracts of the Public Ledger	PISTIS Data Ledger	Internal function
User's Address	Factory's SCEE	API
Store Finalise Purchase Data in the Public Ledger	Factory's SCEE	API
Public Keys	KeyCLoak	API

11.4.1.2 Components to which Smart Contract Execution Engine component provides input

Output Served	Component ingesting the output	Communication Method
Organisation Boundaries		
Check authenticity (get VC attributes) and user creation	Data Factory User Wallet	Internal function (offered by Data Factory User Wallet)
Deployed smart contract to be executed and its functional logic	PISTIS Data Ledger	Internal function (offered by PISTIS Data Ledger)
Check violations, functional logic, and validity of the Smart Contract	Smart Contract Checker	API
Inspection result prior to execution	On/Off Platform Inspector	API
PISTIS Cloud Platform		
User's address Request	Factory's SCEE	API
Deployed smart contract to be executed	PISTIS Data Ledger	Internal function

11.4.2 API Calls Descriptions

Infrastructure	Endpoint	Type	Description	Input / Output
Cloud	/api/scee_cloud/getUserAddressFromFactory/{userId}/{factoryId}	GET	Get user's IOTA Address from a factory	**Input:** • UserID • FactoryID **Output:** • User Address
Data Factory	/api/scee/getTransactionFromPublic?index=5	GET	Get Purchase Provenance Data from public ledger	**Input:** • Index **Output:** • Purchase Public Information

Data Factory	/api/scee/getTransactionCountFromPublic	GET	Get Purchase Provenance Data number of records from public ledger	**Input:** - **Output:** • Number of records
Data Factory	/api/scee/updateUser	POST	Update User's IOTA Wallet	**Input:** • User's UUID • New IOTA Adrdress **Output:** • Transaction Info
Data Factory	/api/scee/setUser	POST	Set User's IOTA Wallet	**Input:** • User's UUID • IOTA Adrdress **Output:** • Transaction Info
Data Factory	/api/scee/storeAnonymizerMetadata	POST	Store Anonymizer Metadata	**Input:** • AssetID • anonymizerMetadata **Output:** • Transaction Info
Data Factory	/api/scee/storeMetadata	POST	Store Dataset Metadata	**Input:** • title • assetID • pistisURL • Metadata • hashOfDataset **Output:** • Transaction Info
Data Factory	/api/scee/storeMetadata	POST	Store Dataset Metadata	**Input:** • title • assetID • pistisURL • Metadata • hashOfDataset **Output:** • Transaction Info
Data Factory	/api/scee/storeSale - One Off	POST	Store One-Off type Sale	**Input:** • Type • Price • Asset ID • Seller Download Limit: • Download Times • Download Frequency

				• Download Until License: • User Group ID • Can Resell • Can Share • Number of Shares • Number of Resells • Can Edit • License Types **Output:** • Transaction Info
Data Factory	/api/scee/storeSale - Subscription	POST	Store Subscription Type Sale	**Input:** • Type • Price • Asset ID • Seller • Subscription Frequency Download Limit: • Download Times • Download Frequency • Download Until License: • User Group ID • Can Resell • Can Share • Number of Shares • Number of Resells • Can Edit • License Types **Output:** • Transaction Info
Data Factory	/api/scee/downloadData	POST	Store Connectors Data	**Input:** • Asset ID • TransactionID • Download Started At **Output:** • Transaction Info
Data Factory	/api/scee/storePurchase	POST	Store a Purchase	**Input:** • Asset ID • AssetFactory • Seller ID • Buyer ID • Price **Output:** • Transaction Info

Data Factory	/api/scee/storeSharing	POST	Store a Sharing action	**Input:** • Asset ID • AssetFactory • Seller ID • Buyer ID **Output:** • Transaction Info
Data Factory	/api/scee/getUser/{UserID}	GET	Get User's IOTA Address	**Input:** • User ID **Output:** • User's IOTA Address
Data Factory	/api/scee/getAnonymizerMetadata/{assetID}	GET	Get Anonymizer's Metadata	**Input:** • Asset ID **Output:** • Anonymizer's Metadata
Data Factory	/api/scee/getMetadata/{assetID}	GET	Get Dataset's Metadata	**Input:** • Asset ID **Output:** • Dataset's Metadata
Data Factory	/api/scee/getLicense/{assetID}	GET	Get Asset's Licence	**Input:** • Asset ID **Output:** • Licence
Data Factory	/api/scee/getDataUsage/{assetID}	GET	Get Asset's Data Usage Data (like downloads, shares etc)	**Input:** • Asset ID **Output:** • DataUsage
Data Factory	/api/scee/getTotalResources/{assetID}	GET	Get Asset's Total Resources. Get the number of Sales, Shares etc	**Input:** • Asset ID **Output:** • TotalResources
Data Factory	/api/scee/getTransactionInfo/{assetID}	GET	Get Asset's Transaction Information, such as price, transactionID etc	**Input:** • Asset ID **Output:** • Asset Transactions Information
Data Factory	/api/scee/getAssetTransactionSummary/{assetID}	GET	Get Asset's Summary in a specific period	**Input:** • Asset ID • from • to **Output:**

				• Asset Summary
Data Factory	/api/scee/getAsset/{assetID}	GET	Get Asset's Information	**Input:** • Asset ID **Output:** • Asset
Data Factory	/api/scee/getPurchase/{transactionID}	GET	Get Purchase's Information	**Input:** • Transaction ID **Output:** • Transaction Information

11.4.3 Sequence Diagrams

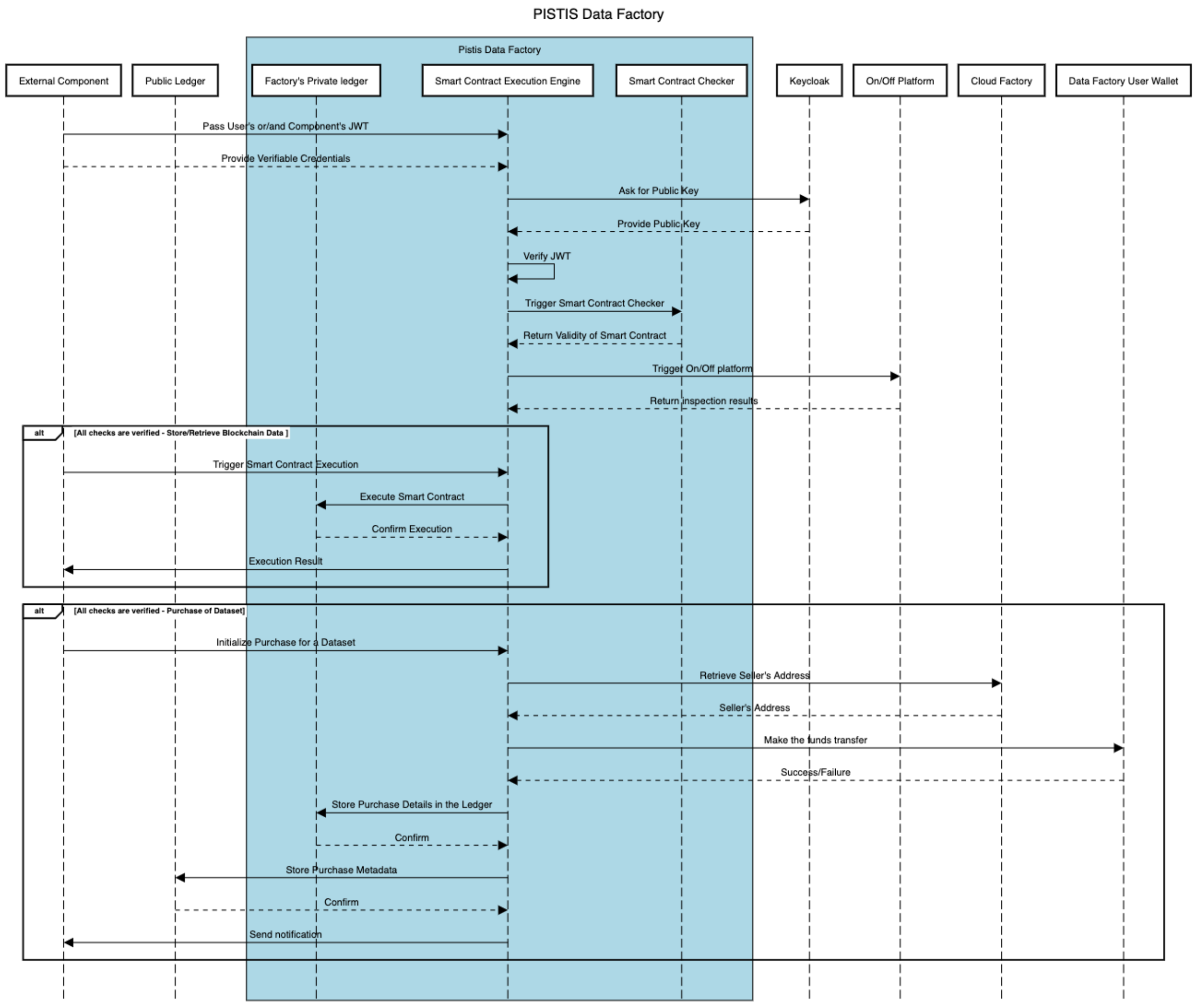

PISTIS Data Factory
Pistis Data Factory
External Component
Public Ledger
Factory's Private ledger
Smart Contract Execution Engine
Smart Contract Checker
Keycloak
On/Off Platform
Cloud Factory
Data Factory User Wallet
Pass User's or/and Component's JWT
Provide Verifiable Credentials
Ask for Public Key
Provide Public Key
Verify JWT
Trigger Smart Contract Checker
Return Validity of Smart Contract
Trigger On/Off platform
Return inspection results
alt
[All checks are verified - Store/Retrieve Blockchain Data]
Trigger Smart Contract Execution
Execute Smart Contract
Confirm Execution
Execution Result
alt
[All checks are verified - Purchase of Dataset]
Initialize Purchase for a Dataset
Retrieve Seller's Address
Seller's Address
Make the funds transfer
Success/Failure
Store Purchase Details in the Ledger
Confirm
Store Purchase Metadata
Confirm
Send notification

Figure 55: Smart Contract Execution Engine Sequence Diagram (Organisation Boundaries)

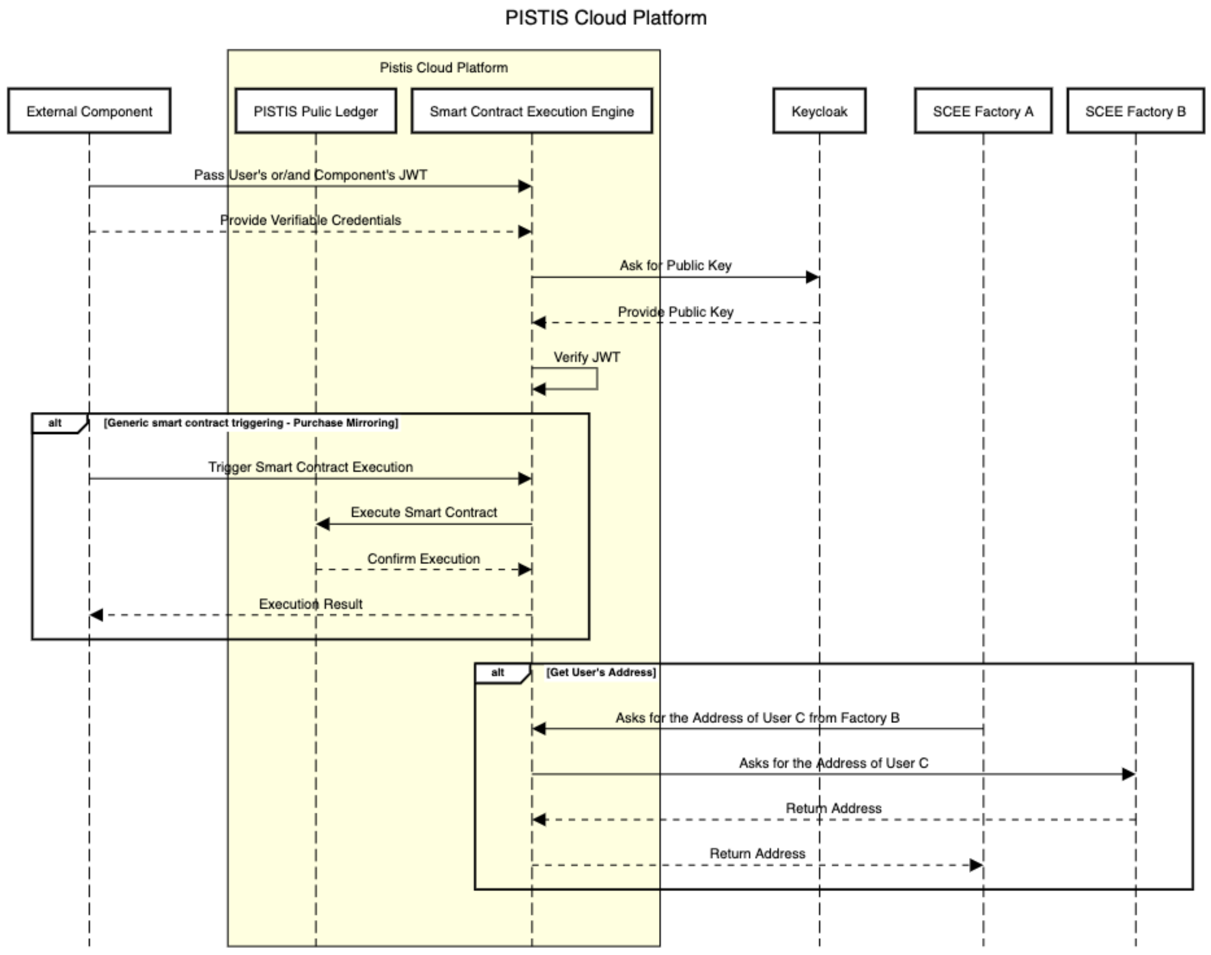


Figure 56: Smart Contract Execution Engine Sequence Diagram (PISTIS Cloud Platform)

11.5 Data Factory User Wallet

The Data Factory User Wallet is the identity wallet implemented on the Quorum Hyperledger Besu, operating within the Ethereum network. It securely stores verifiable credentials for self-sovereign identity (SSI) and handles identity-related operations such as user authentication and authorization within the PISTIS platform. This wallet, alongside the Digital DLT-FIAT Wallet (which is implemented on the IOTA network), forms the comprehensive Wallet solution for each user in PISTIS.

The existence of two distinct wallets offers enhanced flexibility and agility. The Data Factory User Wallet focuses on identity-related operations, while the Digital DLT-FIAT Wallet manages token transactions and interacts with the ledger where PISTIS tokens are stored. Additionally, the Data Factory User Wallet authenticates transactions before they are executed.

For the Alpha version, the Identity Key used by the Data Factory User Wallet is a software-based key, but in the Beta version, it will be upgraded to a hardware-based key for enhanced security. In the current Alpha version, users access their wallet solely through the Smart Contract Execution Engine of their Factory.

Figure 52 presents the high-level architecture of the Identity Wallet including the internal components of the tool.

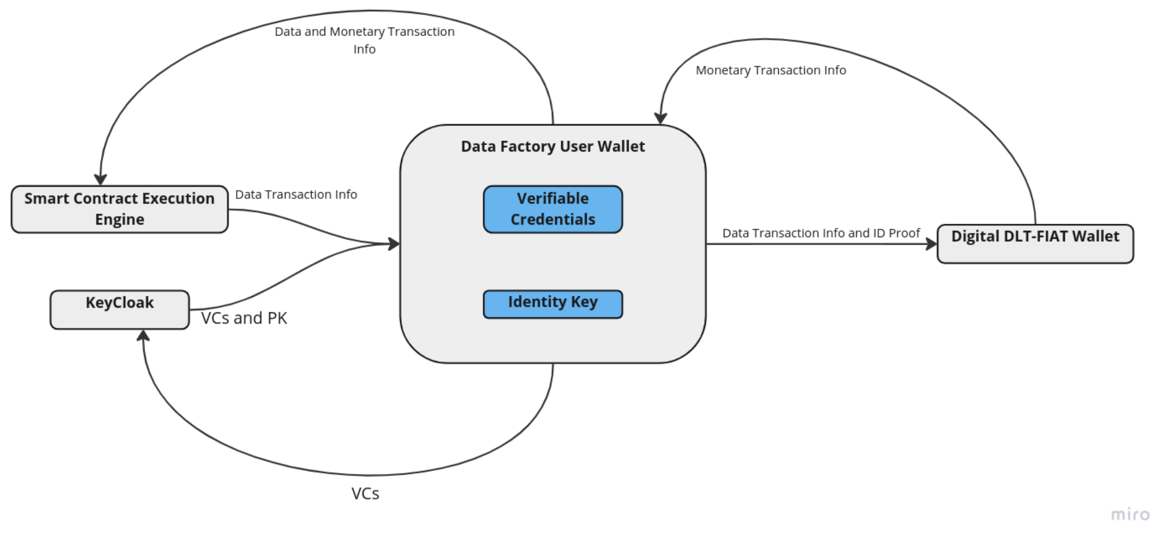


Figure 57: Data Factory User Wallet High-Level Architecture

11.5.1 Relation to Other Components

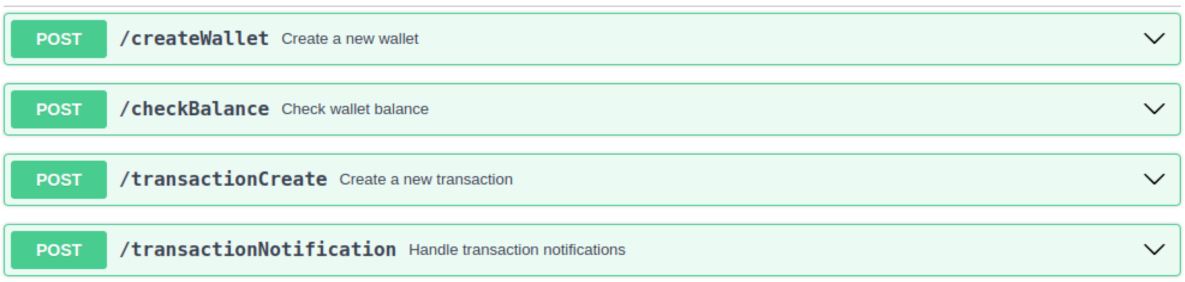
11.5.1.1 Components providing Input to Data Factory User Wallet component

Input Required	Component providing the Input	Communication Method
Verifiable Credentials	KeyCLoak	API
Monetary transactions information	Digital DLT-FIAT Wallet	API
Data transactions information	Smart Contract Execution Engine (Factory)	API
Public Keys	KeyCLoak	API

11.5.1.2 Components to which Data Factory User Wallet component provides input

Output Served	Component ingesting the output	Communication Method
Verifiable Credentials	KeyCloak	API
Data transactions information	Digital DLT-FIAT Wallet	API
Data and Monetary transactions information	Smart Contract Execution Engine (Factory)	API

11.5.2 API Calls Descriptions



POST `/createWallet` Create a new wallet

POST `/checkBalance` Check wallet balance

POST `/transactionCreate` Create a new transaction

POST `/transactionNotification` Handle transaction notifications

11.5.3 Sequence Diagrams

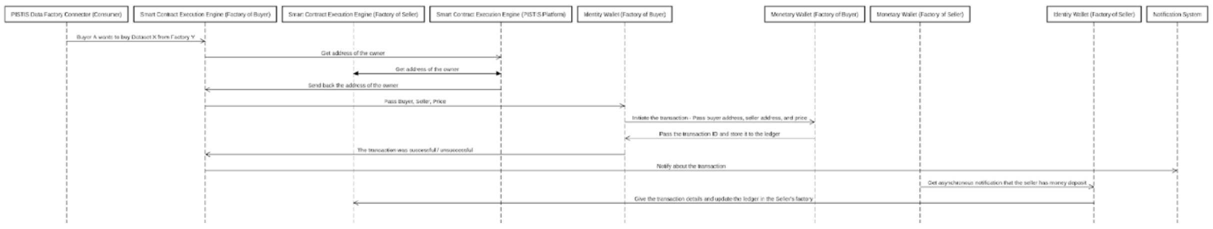

PISTIS Data Factory Connector (Consumer)
Smart Contract Execution Engine (Factory of Buyer)
Smart Contract Execution Engine (Factory of Seller)
Smart Contract Execution Engine (PISTIS Platform)
Identity Wallet (Factory of Buyer)
Monetary Wallet (Factory of Buyer)
Monetary Wallet (Factory of Seller)
Identity Wallet (Factory of Seller)
Notification System
Buyer A wants to buy Dataset X from Factory Y
Get address of the owner
Get address of the owner
Send back the address of the owner
Pass Buyer, Seller, Price
Initiate the transaction - Pass buyer address, seller address, and price
Pass the transaction ID and store it to the ledger
The transaction was successful / unsuccessful
Notify about the transaction
Get asynchronous notification that the seller has money deposit
Give the transaction details and update the ledger in the Seller's factory

Figure 58: Data Factory User Wallet Sequence Diagram

12 LEDGERS BUNDLE

The Ledgers bundle provides the necessary functionality for storing the data and the monetary transactions.

This bundle consists of the following components:

- PISTIS Data Ledger
- PISTIS Monetary Ledger

These are presented in the following sub-sections.

12.1 PISTIS DATA LEDGER

A ledger in PISTIS is a decentralized record-keeping system developed using Quorum Hyperledger Besu on the Ethereum Blockchain. Smart Contracts within the ledger are implemented in Solidity. The ledger communicates exclusively with a component called the Smart Contract Execution Engine (SCEE), which is a Node.js server. External interactions with the ledger are mediated through the SCEE. The ledger is divided into two types: private and public. Each node or organization in the network has its own private ledger, where sensitive information is securely stored. The public ledger, which is singular, mirrors certain transaction details, such as information regarding a Dataset Purchase and Dataset metadata from the private ledgers (IOTAN and BESU) but does not contain any sensitive data, such as participating entities or identity information. This setup ensures secure and controlled access to information, aligning with the decentralized architecture of PISTIS.

12.1.1 RELATION TO OTHER COMPONENTS

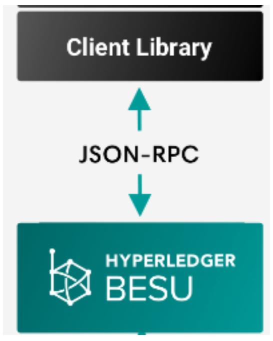
12.1.1.1 Components providing Input to Data Ledger component

Input Required	Component providing the Input	Communication Method
Smart Contract execution (e.g., storing data)	Smart Contract Execution Engine	RPC-HTTP

12.1.1.2 Components to which Data Ledger component provides input

Output Served	Component ingesting the output	Communication Method
Smart Contract execution (e.g., query data)	Smart Contract Execution Engine	RPC-HTTP

12.1.2 API Calls Descriptions



12.1.3 Sequence Diagrams

Figure 59 depicts the sequence of actions for the PISTIS Data Ledger component.

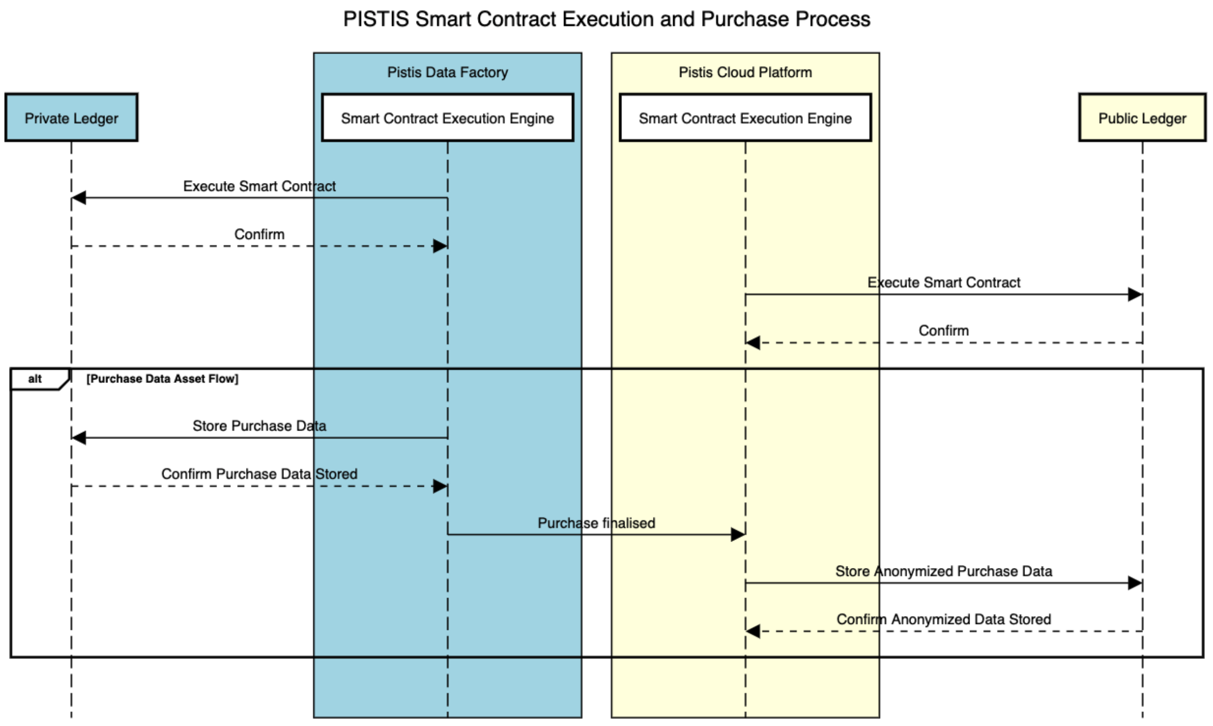

PISTIS Smart Contract Execution and Purchase Process
Private Ledger
Pistis Data Factory
Smart Contract Execution Engine
Pistis Cloud Platform
Smart Contract Execution Engine
Public Ledger
Execute Smart Contract
Confirm
Execute Smart Contract
Confirm
alt
[Purchase Data Asset Flow]
Store Purchase Data
Confirm Purchase Data Stored
Purchase finalised
Store Anonymized Purchase Data
Confirm Anonymized Data Stored

Figure 59: PISTIS Data Ledger Sequence Diagram

12.2 PISTIS Monetary Ledger

IOTA protocol bypasses the natural bottlenecks of conventional blockchains, commonly used by most DLTs and allows for a more scalable and efficient network. It eliminates the need for centralised validation and offers a decentralised and secure environment for transactions and data exchange. Unlike traditional cryptocurrencies, IOTA requires no fees for transaction processing. The amount deducted from the sender's wallet is the same as the amount added to the recipient's wallet, making transactions seamless and cost-effective.

This component will be foundational infrastructure to PISTIS, implementing the essential functionalities for the Digital DLT-FIAT Wallet regarding digital asset management. It is a vital component facilitating monetary transactions for data trading, using cryptocurrency within a private network.

The selection of IOTA DLT network in the monetary management of the PISTIS platform, represents an adoption of an open, feeless, and scalable technology, designed to support value transfers. Its primary objective is to establish trust between users, without the need for a central monetary management authority, while enhancing the operational efficiency of the PISTIS platform.

12.2.1 Relation to Other Components

Components providing Input to Monetary Ledger

Input Required	Component providing the Input	Communication Method
Required parameters for trading and digital asset management (as set by the IOTA protocol).	Digital DLT-FIAT Wallet Backend	API

12.2.1.1 Components to which Monetary Ledger provides input

Output Served	Component ingesting the output	Communication Method
Currency transaction and digital asset management action results.	Digital DLT-FIAT Wallet Backend	API

12.2.2 API Calls Descriptions

API Calls Descriptions are described in the IOTA protocol and provided by the IOTA Foundation.

12.2.3 SEQUENCE DIAGRAMS

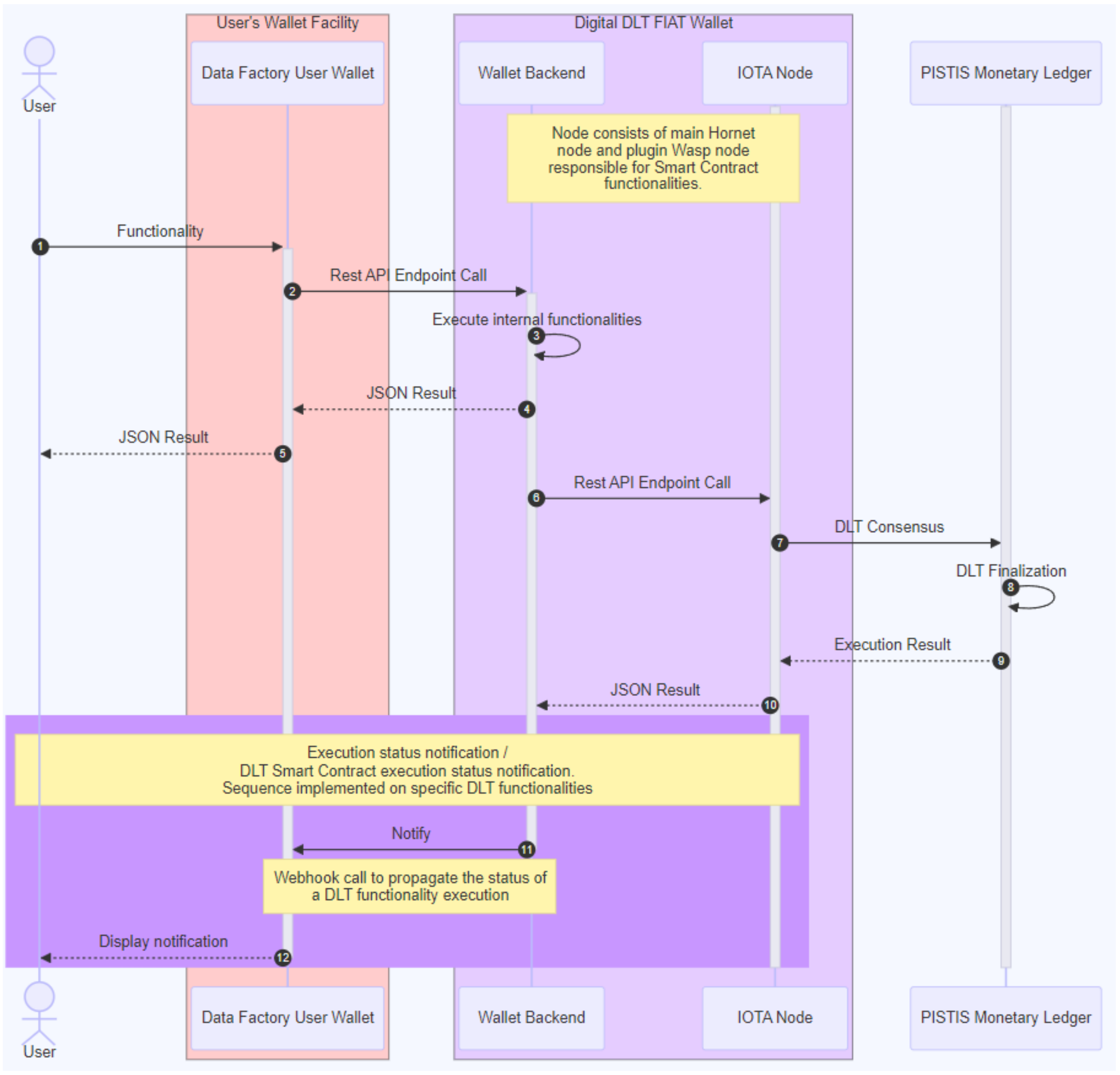

User's Wallet Facility
Digital DLT FIAT Wallet
User
Data Factory User Wallet
Wallet Backend
IOTA Node
PISTIS Monetary Ledger
Node consists of main Hornet node and plugin Wasp node responsible for Smart Contract functionalities.
1 Functionality
2 Rest API Endpoint Call
3 Execute internal functionalities
4 JSON Result
5 JSON Result
6 Rest API Endpoint Call
7 DLT Consensus
8 DLT Finalization
9 Execution Result
10 JSON Result
Execution status notification / DLT Smart Contract execution status notification. Sequence implemented on specific DLT functionalities
11 Notify
Webhook call to propagate the status of a DLT functionality execution
12 Display notification

Figure 60: PISTIS Monetary Ledger Sequence Diagram

13 AI & Interoperability Repositories Bundle

The AI & Interoperability Repos bundle provides the different repositories for storing and propagating different models (data models, AI models and metadata models) that need to be consumed by the different components.

This bundle consists of the following components:

- PISTIS Models Repository
- Data Factory ML Models Repository
- AI Model Editor

These are presented in the following sub-sections.

13.1 PISTIS Models Repository

The PISTIS Models Repository is responsible for the storage, management, and government of PISTIS models, supporting better understanding and control of their information and architecture. These models include:

- **Data models** that define entities, attributes, and relationships within a specific domain. Data Providers must use the data models when describing their actual data and a browser for these models will be available.
- **Metadata models** that define the metadata that shall be provided to accompany each dataset traded over PISTIS. This repository is s based on the RDF standard and established sub-standards, such as DCAT and Gaia-X self-descriptions. The Data Catalogues read the models during the start-up process to configure themselves accordingly. Additionally, the Metadata Model Management ensures the automatic generation of a machine-and-human-readable documentation.
- **Pre-trained ML models** that are consumed by the components of the "Data Management and Assessment" bundle to improve the data management and transformation operations
- **Monetisation AI models**, which are used by the analytics engine residing in the Cloud platform to accommodate the needs of the PISTIS Market Insights component

The PISTIS Models Repository also supports version control for the PISTIS models, allowing users to track changes over time, which is crucial for managing updates and ensuring consistency across the whole PISTIS ecosystem. It also maintains metadata associated with each data model, providing information about their version, size, and creation/last update date. Users can search and retrieve specific data models based on various criteria, facilitating easy access to relevant information.

In addition, the PISTIS Model Administrator is the role coming from the organisationt hat is managing the overall PISTIS infrastructure that maintains the models and can upload new artefacts and fill-in new metadata information on the existing ones.

13.1.1 Relation to Other Components

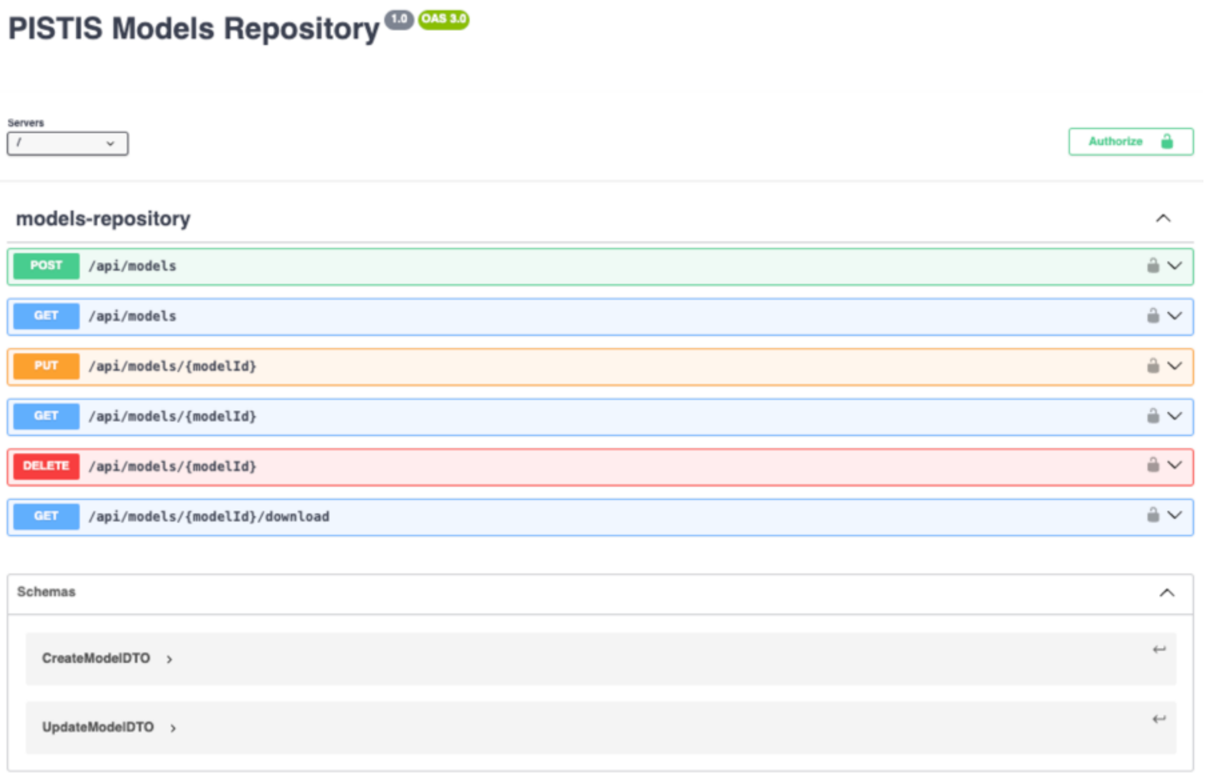
13.1.1.1 Components providing Input to the PISTIS Models Repository

Input Required	Component providing the Input	Communication Method
AI Model	AI Model Designer	API

13.1.1.2 Components to which the PISTIS Models Repository provides input

Output Served	Component ingesting the output	Communication Method
Metadata Models	Factory Data Catalogue	API
Metadata Models	PISTIS Data Catalogue	API
Data Models	Job Configurator	API
Metadata Models	Metadata Quality Assessment	API
Data Models	Data Check-in	API
Data Models	Data Transformation	API
Data Models	Data Enrichment	API
Data Models	Data Quality Assessment	API
Pre-Trained ML Models	Analytics Engine (Data Factory)	API
Pre-Trained ML Models	Data Factory ML Models Repository	API
Monetisation AI Models	Analytics Engine (Cloud Platform)	API

13.1.2 API Calls Descriptions



s

13.1.3 Sequence Diagrams

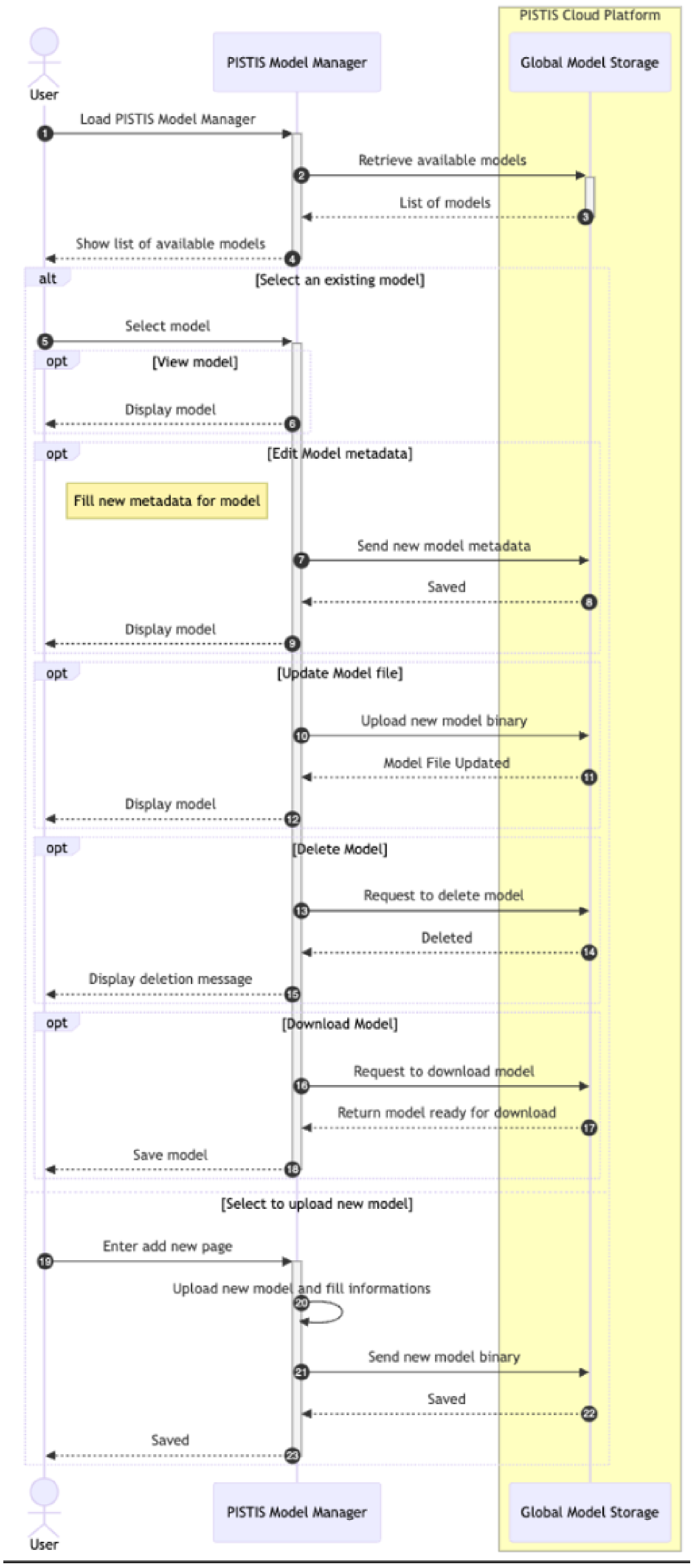


Figure 61: PISTIS Models Repository - Sequence Diagram

13.2 Data Factory ML Models Repository

The Data Factory ML Models repository will provide support for CRUD and serving operations over a concrete pre-trained model.

This repository is essentially a similar deployment of the PISTIS Models Repository but concerns only ML models that can be uploaded by Data Factory users to run analyses over their own data, while they can also fetch already Pre-trained ML models from the PISTIS Models Repository.

13.2.1 Relation to Other Components

13.2.1.1 Components providing Input to Pre-trained AI Models Repo

Input Required	Component providing the Input	Communication Method
Ready-Made Pre-Trained Models	PISTIS Models Repository (Data Factory environment))	API

13.2.1.2 Components to which Pre-trained AI Models Repo provides input

Output Served	Component ingesting the output	Communication Method
Model	Analytics Engine (Data Factory environment)	API

13.2.2 API Calls Descriptions

API Calls (to be decided how this will be formatted).

13.2.3 SEQUENCE DIAGRAMS

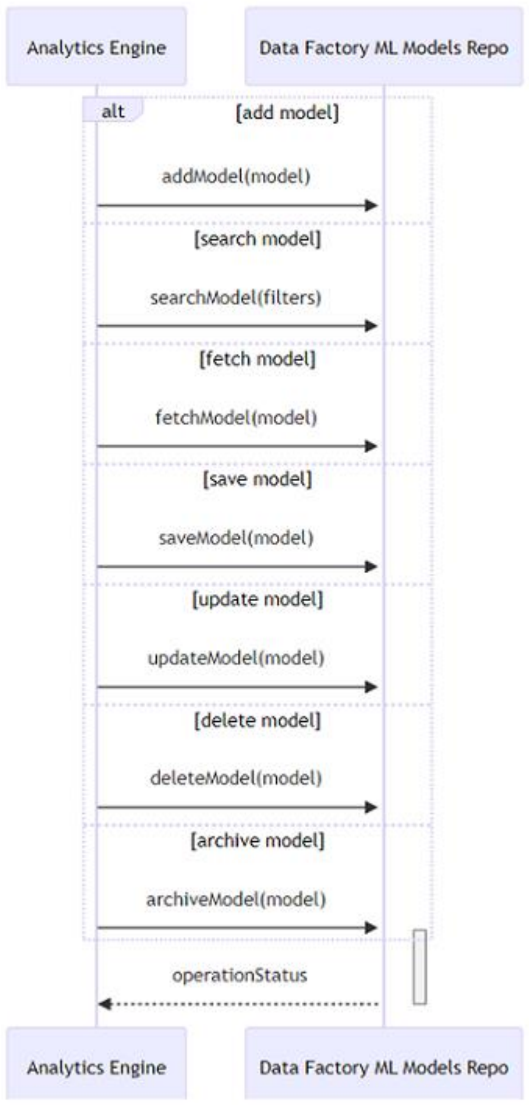


Figure 62: Factory ML Models Repo Sequence Diagram

13.3 AI Model Editor

The AI Model Editor will provide support for creating, editing, and sharing computational AI models.

AI Model Editor will allow the end user to cover a basic workflow that includes at least the following tasks:

- Create a project enabling collaboration (or not) with others to work with data.
- Add a notebook to the project.
- Add code and run the notebook.
- Review the model pipelines and save the desired pipeline as a model.
- Deploy and test a concrete model.

13.3.1 Relation to Other Components

13.3.1.1 Components providing Input to AI Model Editor

Input Required	Component providing the Input	Communication Method
Ready-made ML Models	Pre-Trained ML Models Repository	API
Ready-made ML/Monetisation AI Models	PISTIS Models Repository (Pre-trained ML and Monetisation AI Models Repository)	API

13.3.1.2 Components to which AI Model Editor provides input

Output Served	Component ingesting the output	Communication Method
AI Model to be executed	Analytics Engine	API
AI Model to be saved	PISTIS Models Repository	API
AI Model to be saved	Data Factory ML Models Repository	API

13.3.2 API Calls Descriptions

The AI Model Editor will be using the API endpoints provided by the PISTIS Models Repository and the Data Factory ML Models Repository

13.3.3 Sequence Diagrams

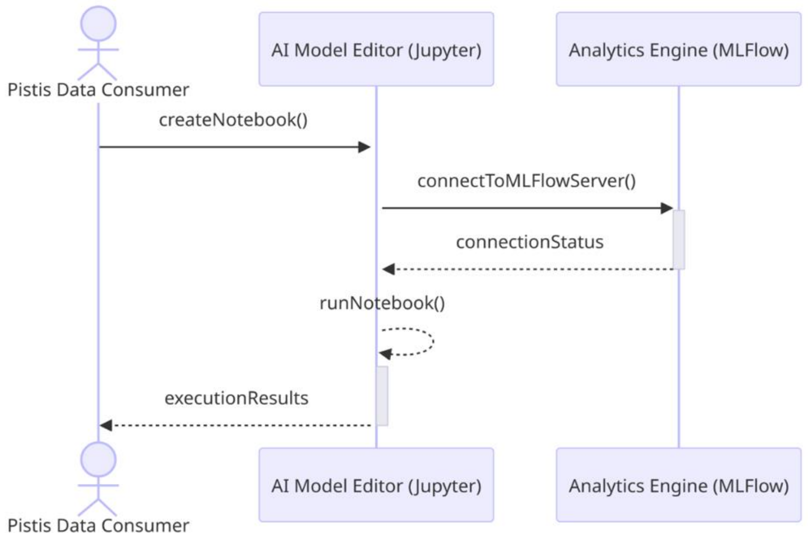


Figure 63: AI Model Editor Sequence Diagram

14 Identity and Access Management Bundle

The Identity & Access Management bundle governs identity provisioning and validation, as well as access management, both at the data and component levels. This bundle consists of the following components:

- Identity Manager (including several sub-components)

This is presented in the following sub-sections.

14.1 Identity Manager

The Identity Manager component within the PISTIS platform will ensure security and controlled interactions within its ecosystem. The component will integrate and provide various methods and technologies to authenticate, authorize, and manage identities and resource access effectively.

The component will provide authentication and authorisation based on OpenIDC[4] protocol, contributing to securing a user's identity validation and single sign-on capabilities. Various access control policies will be used for authorisation, such as Attribute-Based Access Control (ABAC) and Role-Based Access Control (RBAC), depending on the requirements of each module and the desired functionality and applicable scenario. Additionally, the component will provide to the PISTIS platform alignment with European Union standards for electronic identification and transaction, by integrating with eIDAS[5] verifiable credentials.

The identifiable functional-wise internal components of the PISTIS Identity Manager are: (a) Access Policy Engine, (b) Identity Provider/Validator, (c) Resource Manager and (d) Secure Services & Users Public Keys Store.

14.1.1 Relation to Other Components

14.1.1.1 Components providing Input to the Identity Manager

Input Required	Component providing the Input	Communication Method
OpenIDC Client Credentials	All components and modules that need Authentication	HTTP Request
OpenIDC Client Credentials and Authorisation parameters	All components and modules that need Authorisation	HTTP Request
Resource representation	Data Check-In	HTTP Request
Access Policy representation	Access Policy Editor	HTTP Request
Data Factory Representation	Data Factory Registrant	HTTP Request

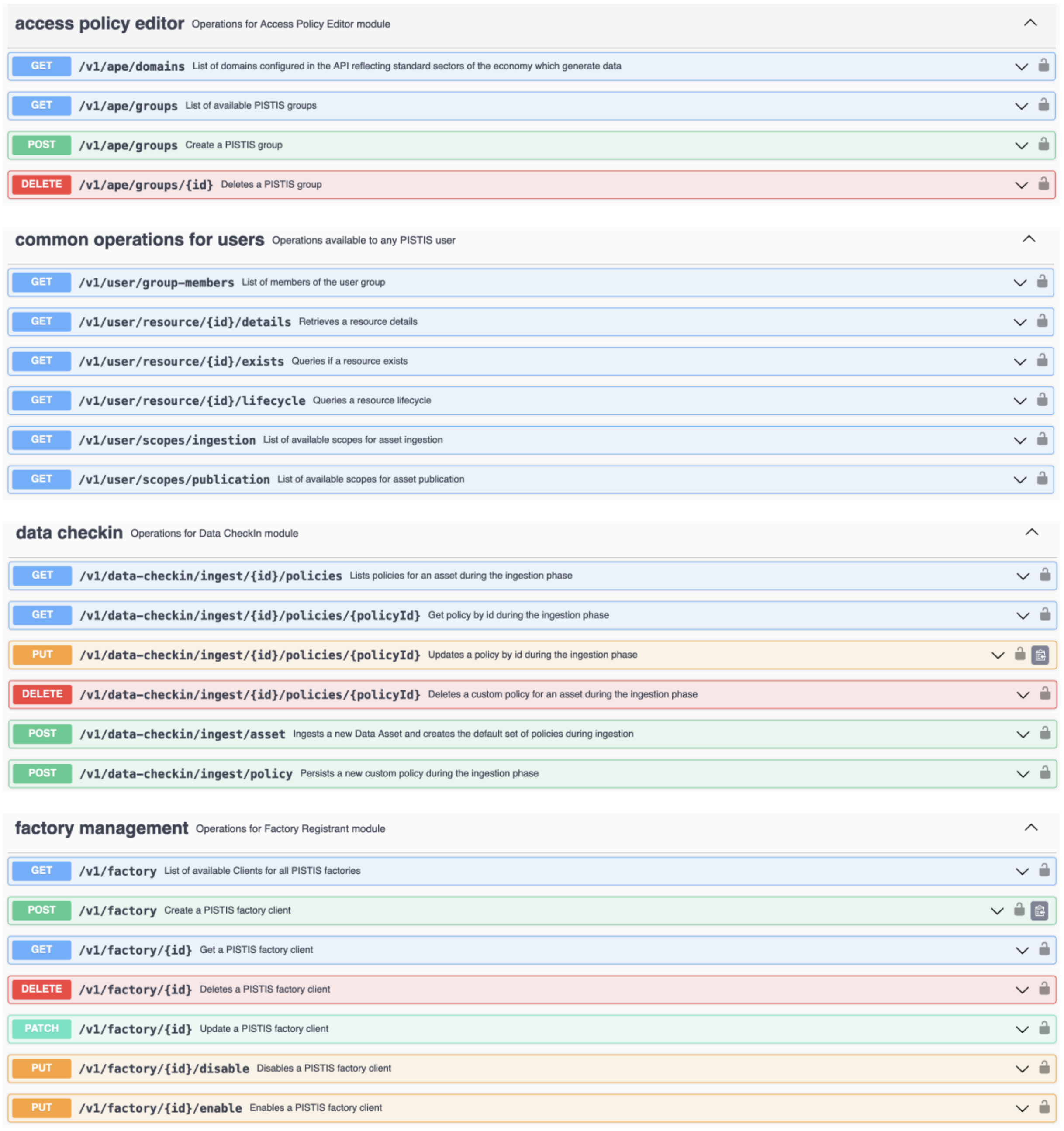
14.1.1.2 Components to which Identity Manager provides input

Output Served	Component ingesting the output	Communication Method
OpenIDC Authentication Token	All components and modules that need Authentication	HTTP Request

[4] https://github.com/OpenIDC
[5] https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52021PC0281&from=EN

JSON Web Token (JWT)	All components and modules that need Authorisation	HTTP Request
List of Access Policies representation	Access Policy Editor	HTTP Request
List of Access Policies representation	Asset Description Bundler	HTTP Request
True/False	On/Off Platform Smart Contract Inspector	HTTP Request

14.1.2 API Calls Descriptions



access policy editor Operations for Access Policy Editor module

GET	`/v1/ape/domains`	List of domains configured in the API reflecting standard sectors of the economy which generate data
GET	`/v1/ape/groups`	List of available PISTIS groups
POST	`/v1/ape/groups`	Create a PISTIS group
DELETE	`/v1/ape/groups/{id}`	Deletes a PISTIS group

common operations for users Operations available to any PISTIS user

GET	`/v1/user/group-members`	List of members of the user group
GET	`/v1/user/resource/{id}/details`	Retrieves a resource details
GET	`/v1/user/resource/{id}/exists`	Queries if a resource exists
GET	`/v1/user/resource/{id}/lifecycle`	Queries a resource lifecycle
GET	`/v1/user/scopes/ingestion`	List of available scopes for asset ingestion
GET	`/v1/user/scopes/publication`	List of available scopes for asset publication

data checkin Operations for Data CheckIn module

GET	`/v1/data-checkin/ingest/{id}/policies`	Lists policies for an asset during the ingestion phase
GET	`/v1/data-checkin/ingest/{id}/policies/{policyId}`	Get policy by id during the ingestion phase
PUT	`/v1/data-checkin/ingest/{id}/policies/{policyId}`	Updates a policy by id during the ingestion phase
DELETE	`/v1/data-checkin/ingest/{id}/policies/{policyId}`	Deletes a custom policy for an asset during the ingestion phase
POST	`/v1/data-checkin/ingest/asset`	Ingests a new Data Asset and creates the default set of policies during ingestion
POST	`/v1/data-checkin/ingest/policy`	Persists a new custom policy during the ingestion phase

factory management Operations for Factory Registrant module

GET	`/v1/factory`	List of available Clients for all PISTIS factories
POST	`/v1/factory`	Create a PISTIS factory client
GET	`/v1/factory/{id}`	Get a PISTIS factory client
DELETE	`/v1/factory/{id}`	Deletes a PISTIS factory client
PATCH	`/v1/factory/{id}`	Update a PISTIS factory client
PUT	`/v1/factory/{id}/disable`	Disables a PISTIS factory client
PUT	`/v1/factory/{id}/enable`	Enables a PISTIS factory client

14.1.3 Sequence Diagrams

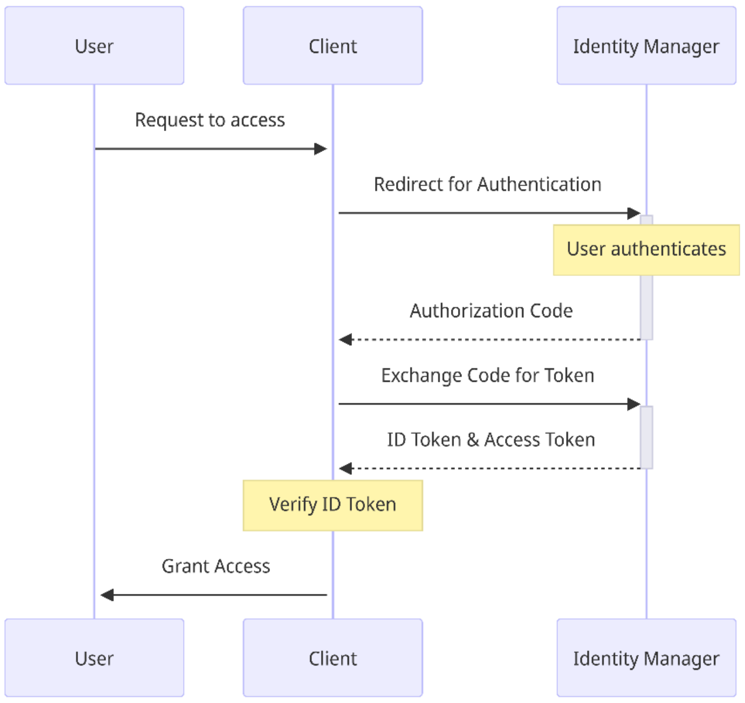


Figure 64: Identity Manager Authentication Sequence Diagram

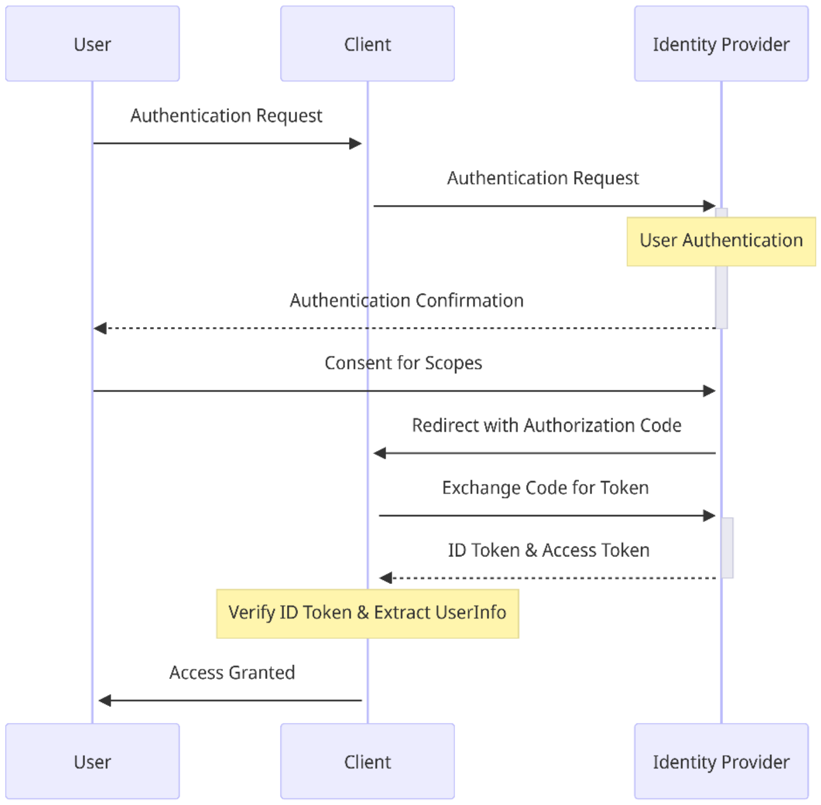


Figure 65: Identity Manager Authorisation Sequence Diagram

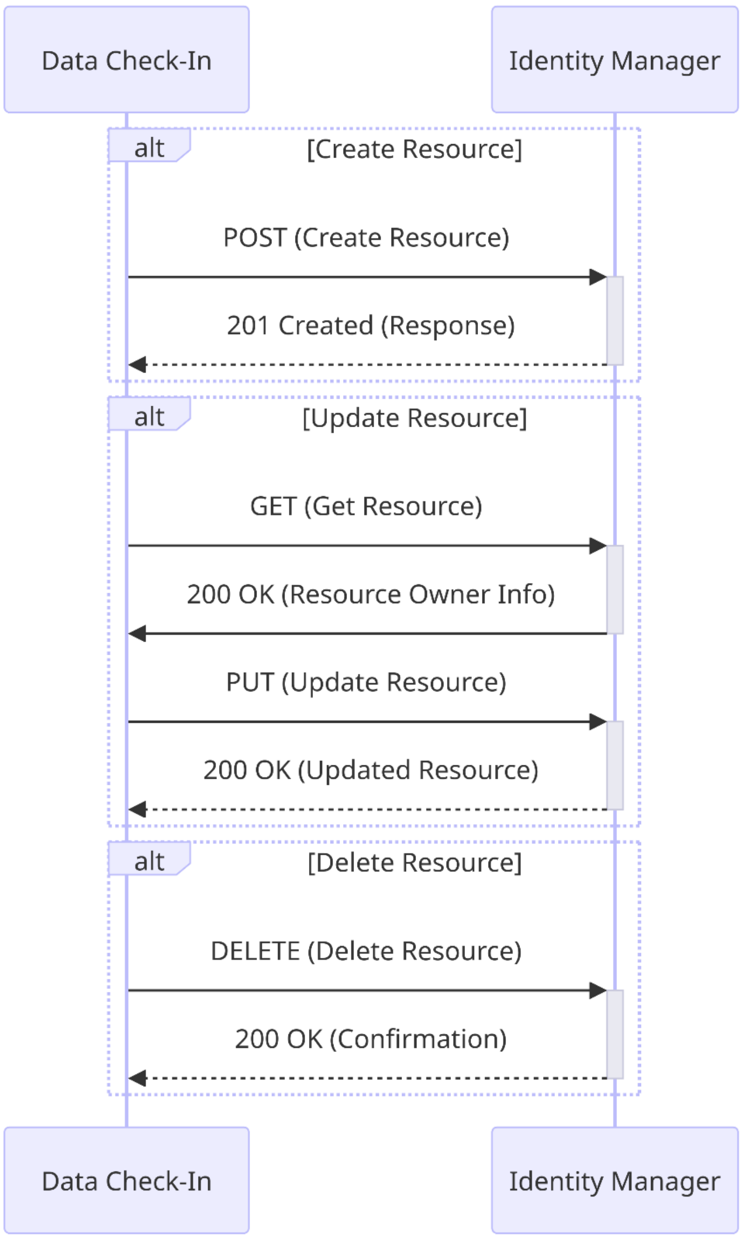


Figure 66: Identity Manager Resource Management Sequence Diagram

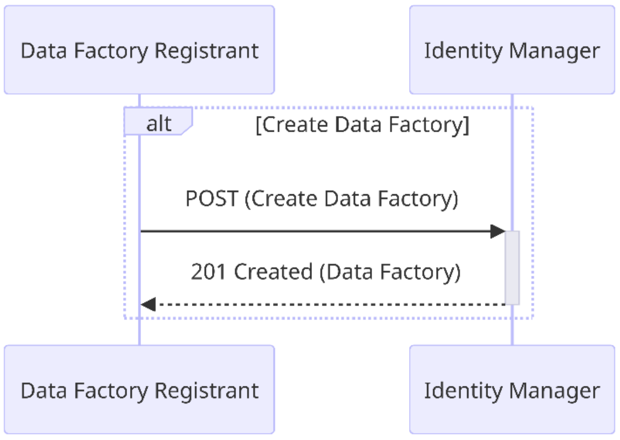


Figure 67: Identity Manager Data Factory Registration Sequence Diagram

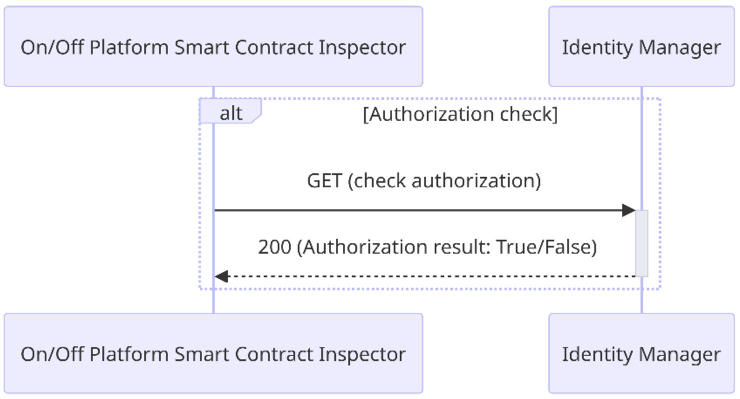

On/Off Platform Smart Contract Inspector
Identity Manager
alt
[Authorization check]
GET (check authorization)
200 (Authorization result: True/False)
On/Off Platform Smart Contract Inspector
Identity Manager

Figure 68: Identity Manager Authorisation check for user, audience, and scope Sequence Diagram

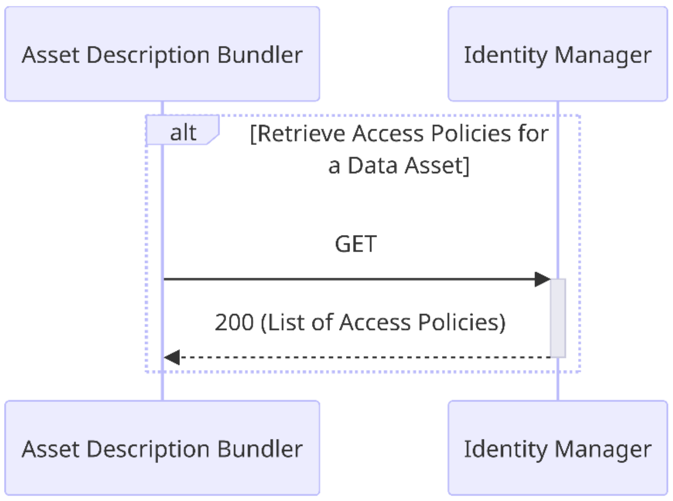

Asset Description Bundler
Identity Manager
alt
[Retrieve Access Policies for a Data Asset]
GET
200 (List of Access Policies)
Asset Description Bundler
Identity Manager

Figure 69: Identity Manager Access Policies retrieval for a Data Asset Sequence Diagram

15 System Services Bundle

The System Services bundle is used by system administrators to configure and monitor the overall PISTIS environment.

This bundle consists of the following components:

- Data Factories Registrant
- System and Activities Monitor

These are presented in the following sub-sections.

15.1 Data Factories Registrant

The Data Factories Registrant is responsible for the registration of new data factories in the PISTIS ecosystem and their connection with the PISTIS Platform through dedicated connectors.

The PISTIS Data Factory enables the user to make a request for the registration of the newly deployed installation. Upon confirmation of non-existence in the registry, the Factory Data Service sends a notification to the PISTIS administrator who will decide either to accept or reject this new factory. In case of acceptance, a notification is sent to the user through the Discovery Service and the Factory Registrant that the new data factory is connected to the PISTIS Platform. As soon as the new instance is registered the Data Factories Registrant executes an hourly synchronisation with the Factory Discovery Service that will check if this Data factory is connected to the network.

15.1.1 Relation to Other Components

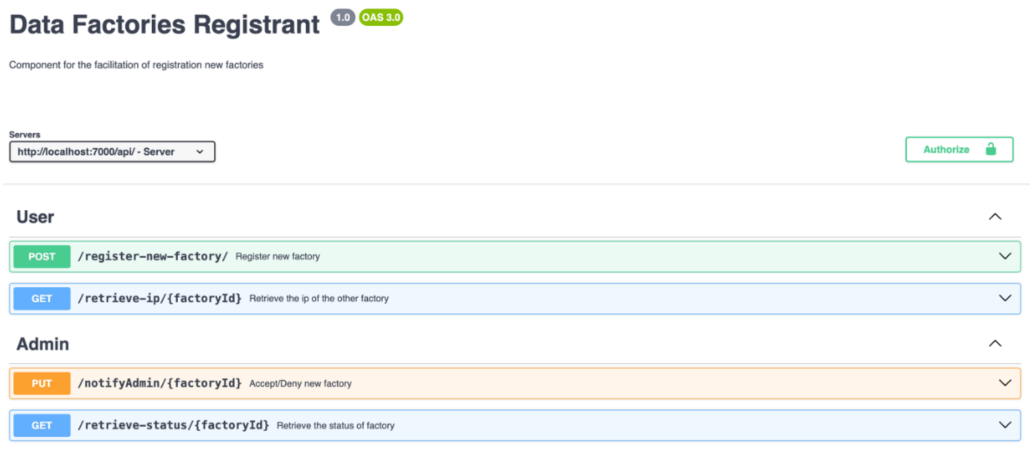
15.1.1.1 Components providing Input to Data Factory Registrant

Input Required	Component providing the Input	Communication Method
Request for connection of data factory to PISTIS network	PISTIS Data Factory	API

15.1.1.2 Components to which Data Factory Registrant provides input

Output Served	Component ingesting the output	Communication Method
Information for registration of data factory in registry	PISTIS Data Factory	API
Information for data factory details	Identity Manager	API

15.1.2 API Calls Descriptions



Data Factories Registrant 1.0 OAS 3.0

Component for the facilitation of registration new factories

Servers
http://localhost:7000/api/ - Server

Authorize

User

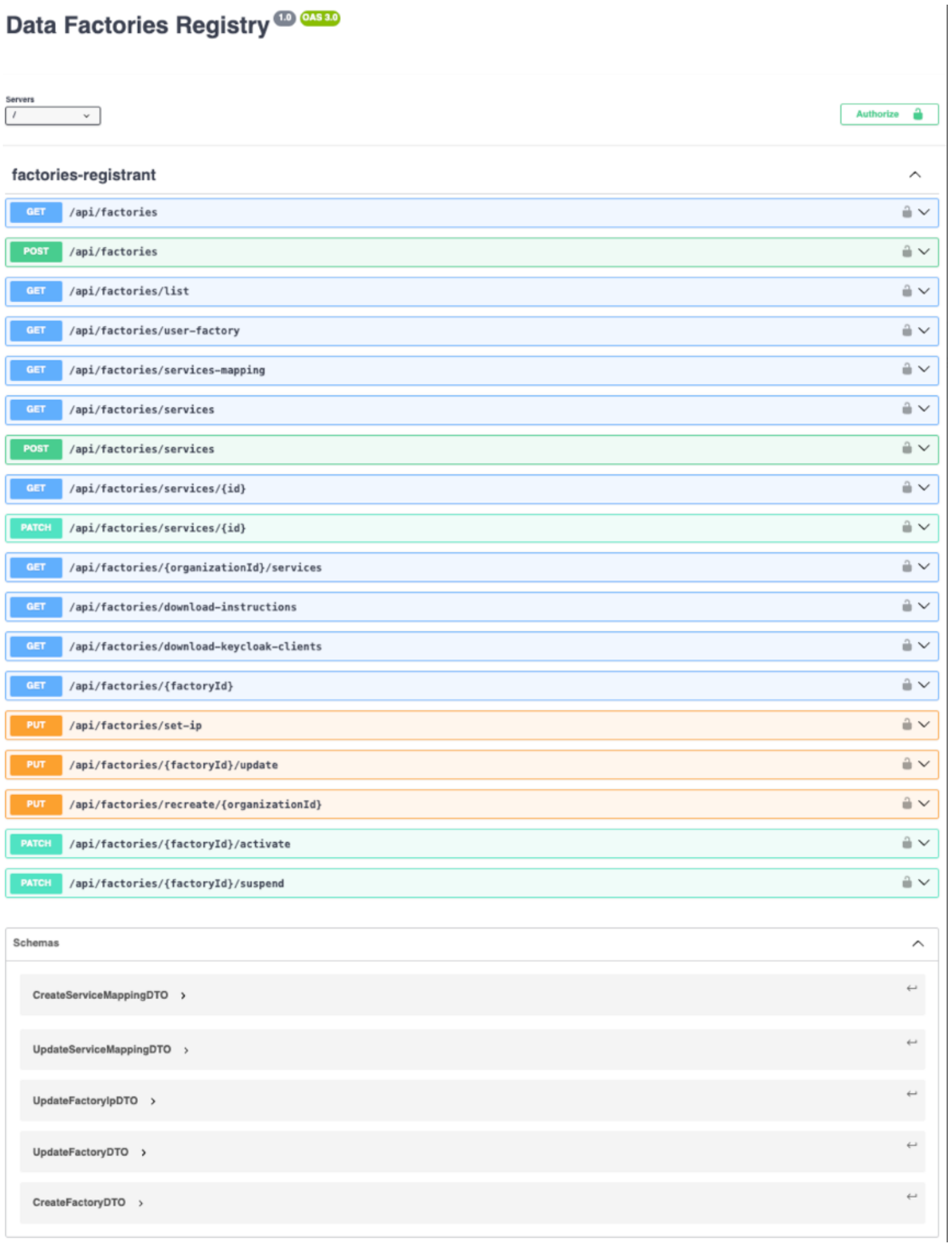
POST /register-new-factory/ Register new factory

GET /retrieve-ip/{factoryId} Retrieve the ip of the other factory

Admin

PUT /notifyAdmin/{factoryId} Accept/Deny new factory

GET /retrieve-status/{factoryId} Retrieve the status of factory



Data Factories Registry 1.0 OAS 3.0

Servers
/

Authorize

factories-registrant

GET /api/factories

POST /api/factories

GET /api/factories/list

GET /api/factories/user-factory

GET /api/factories/services-mapping

GET /api/factories/services

POST /api/factories/services

GET /api/factories/services/{id}

PATCH /api/factories/services/{id}

GET /api/factories/{organizationId}/services

GET /api/factories/download-instructions

GET /api/factories/download-keycloak-clients

GET /api/factories/{factoryId}

PUT /api/factories/set-ip

PUT /api/factories/{factoryId}/update

PUT /api/factories/recreate/{organizationId}

PATCH /api/factories/{factoryId}/activate

PATCH /api/factories/{factoryId}/suspend

Schemas

CreateServiceMappingDTO

UpdateServiceMappingDTO

UpdateFactoryIpDTO

UpdateFactoryDTO

CreateFactoryDTO

15.1.3 Sequence Diagrams

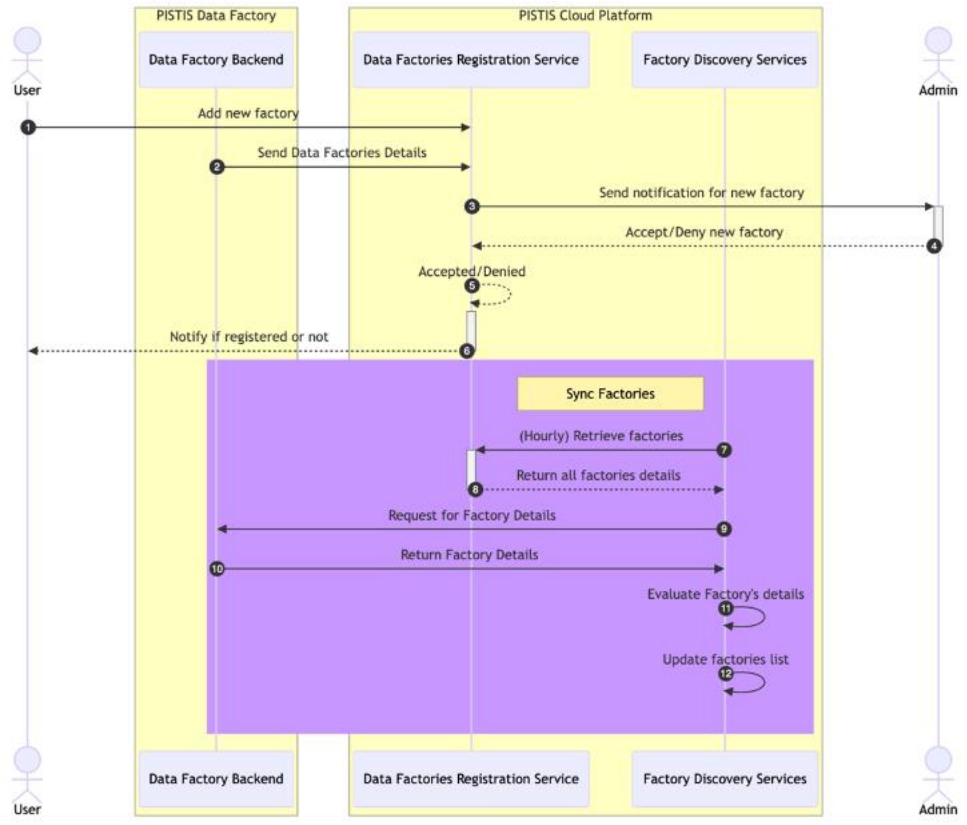

PISTIS Data Factory
PISTIS Cloud Platform
User
Data Factory Backend
Data Factories Registration Service
Factory Discovery Services
Admin
Add new factory
Send Data Factories Details
Send notification for new factory
Accept/Deny new factory
Accepted/Denied
Notify if registered or not
Sync Factories
(Hourly) Retrieve factories
Return all factories details
Request for Factory Details
Return Factory Details
Evaluate Factory's details
Update factories list

Figure 70: Data Factories Registrant Sequence Diagram

15.2 System and Activities Monitor

he System and Activities Monitor of the PISTIS Cloud platform has the main purpose of providing users with a comprehensive dashboard where they can gain insights into the platform's resource allocation and performance. This component communicates with the backbone of the PISTIS Cloud Platform. Additionally, by interacting with the Data Ledger, the System and Activities Monitor receives metrics on data sharing activities that can be visualized.

15.2.1 Relation to Other Component

15.2.1.1 Components providing Input to the System and Activities Monitor

Input Required	Component providing the Input	Communication Method
Data Transactions Activity metrics	Data Ledger	API
Users Information (aggregated)	Identity Manager	API
Resources Info	Cloud Platform Backbone	API

15.2.1.2 Components to which the System and Activities Monitor provides input

Output Served	Component ingesting the output	Communication Method
N/A		

15.2.2 API Calls Descriptions

N/A

15.2.3 Sequence Diagrams

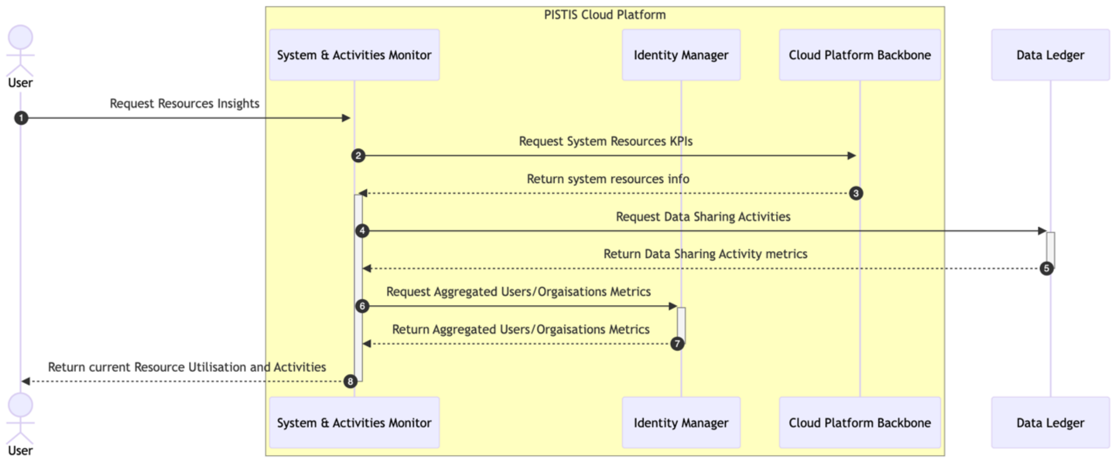

PISTIS Cloud Platform
User
System & Activities Monitor
Identity Manager
Cloud Platform Backbone
Data Ledger
1 Request Resources Insights
2 Request System Resources KPIs
3 Return system resources info
4 Request Data Sharing Activities
5 Return Data Sharing Activity metrics
6 Request Aggregated Users/Orgaisations Metrics
7 Return Aggregated Users/Orgaisations Metrics
8 Return current Resource Utilisation and Activities

Figure 71: Resources and Activities Monitor- Sequence Diagram