



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

D4.1 PISTIS Reference Architecture and API Documentation

Editor(s)	Sotiris Koussouris, Marios Phinikettos
Lead Beneficiary	Suite5
Status	Final
Version	1.00
Due Date	31/12/2023
Delivery Date	31/12/2023
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	D4.1 - PISTIS Reference Architecture and API Documentation
Contributor(s)	Suite5, ATOS, FHG, EUT, SPH, UBITECH, ATHENA, ASSENTIAN, ICCS
Reviewer(s)	Wiktor Markiewicz (IDC) Gianluigi Viscusi (IMPERIAL)
Abstract	This deliverable presents the first version of the PISTIS architecture, defining the different components that constitute the overall PISTIS environment, and provides the description of the first version of the APIs, which will be essential for designing and integrating the different components.

Executive Summary

Deliverable D4.1 presents the first version of the PISTIS architecture. It is dedicated to identifying the required components (which are grouped into bundles) and interactions between the components, defining the APIs that will be facilitating the control and data exchange for bringing around the anticipated functionalities of the overall system.

As such, the document at hand first presents the overview of the architecture and goes on with describing at a high level the underlying components of the different bundles and the interactions between them.

The overall PISTIS platform is envisaged as a federated ecosystem of deployments (PISTIS Data Factories) that help Data Providers to expose their data assets and Data Consumers to acquire these, resulting in building a Data Space implementation that is governed by a central entity (PISTIS Cloud Platform) that is acting as the facilitator for the monetisation and the transaction facilitation services.

The overall platform is composed of different bundles which aim to cover the essential functionalities and features to provide to the involved stakeholders a secure and trusted environment that can support fair data transactions, based on objective methods for monetising their data assets. These bundles are the following:

- **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.
- **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 interest for 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 provides 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 architecture presented in the document is an important reference for developing the overall PISTIS platform. It will be further refined and updated to document the coming releases of the platform, as new releases of the architecture are expected under WP4.

Table of Contents

- 1 Introduction 14
 - 1.1 Document structure 14
- 2 PISTIS High Level Architecture 15
 - 2.1 PISTIS Architecture 15
 - 2.2 Bundles Composing PISTIS 18
 - 2.3 Architecture’s On-Line Resources 19
- 3 Data Management and Assessment Bundle 20
 - 3.1 Data Check-In 20
 - 3.1.1 Relation to Other Components 21
 - 3.1.2 API Calls Descriptions 21
 - 3.1.3 Sequence Diagrams 21
 - 3.2 Data Transformation 22
 - 3.2.1 Relation to Other Components 22
 - 3.2.2 API Calls Descriptions 23
 - 3.2.3 Sequence Diagrams 23
 - 3.3 Job configurator 24
 - 3.3.1 Relation to Other Components 24
 - 3.3.2 API Calls Descriptions 24
 - 3.3.3 Sequence Diagrams 25
 - 3.4 Analytics Engine 25
 - 3.4.1 Relation to Other Components 25
 - 3.4.2 API Calls Descriptions 26
 - 3.4.3 Sequence Diagrams 27
 - 3.5 Data Enrichment..... 27
 - 3.5.1 Relation to Other Components 28
 - 3.5.2 API Calls Descriptions 28
 - 3.5.3 Sequence Diagrams 29
 - 3.6 Data Quality Assessment..... 30
 - 3.6.1 Relation to Other Components 31
 - 3.6.2 API Calls Descriptions 31
 - 3.6.3 Sequence Diagrams 32
 - 3.7 Data Insights Generator 33
 - 3.7.1 Relation to Other Components 33

3.7.2	API Calls Descriptions	33
3.7.3	Sequence Diagrams	34
4	Data & Metadata Storage Bundle	35
4.1	Data Catalogues.....	35
4.1.1	Relation to Other Components	35
4.1.2	API Calls Descriptions	36
4.1.3	Sequence Diagrams	38
4.2	Factory Data Storage	39
4.2.1	Relation to Other Components	40
4.2.2	API Calls Descriptions	41
4.2.3	Sequence Diagrams	42
5	Data Discovery Bundle	43
5.1	Distributed Query Engine	43
5.1.1	Relation to Other Components	43
5.1.2	API Calls Descriptions	44
5.1.3	Sequence Diagrams	45
5.2	Matchmaking Services.....	46
5.2.1	Relation to Other Components	47
5.2.2	API Calls Descriptions	47
5.2.3	Sequence Diagrams	48
6	Data Exchange Bundle	49
6.1	PISTIS Data Factory Connector	49
6.1.1	Relation to Other Components	49
6.1.2	API Calls Descriptions	50
6.1.3	Sequence Diagrams	51
6.2	Smart Contract Checker	51
6.2.1	Relation to Other Components	52
6.2.2	API Calls Descriptions	53
6.2.3	Sequence Diagrams	53
7	Data Monetisation Bundle	54
7.1	Access Policy Editor	54
7.1.1	Relation to Other Components	55
7.1.2	API Calls Descriptions	55
7.1.3	Sequence Diagrams	56

- 7.2 Asset Description Bundler 58
 - 7.2.1 Relation to Other Components 59
 - 7.2.2 API Calls Descriptions 60
 - 7.2.3 Sequence Diagrams 61
- 7.3 FAIR Data Valuation Service 61
 - 7.3.1 Relation to Other Components 62
 - 7.3.2 API Calls Descriptions 62
 - 7.3.3 Sequence Diagrams 63
- 7.4 PISTIS Digital DLT-FIAT Wallet 63
 - 7.4.1 Relation to Other Components 63
 - 7.4.2 API Calls Descriptions 65
 - 7.4.3 Sequence Diagrams 66
- 7.5 Data Investment Planner 68
 - 7.5.1 Relation to Other Components 68
 - 7.5.2 API Calls Descriptions 68
 - 7.5.3 Sequence Diagrams 69
- 7.6 PISTIS Market Insights 69
 - 7.6.1 Relation to Other Components 69
 - 7.6.2 API Calls Descriptions 70
 - 7.6.3 Sequence Diagrams 70
- 7.7 NFT Generator 70
 - 7.7.1 Relation to Other Components 71
 - 7.7.2 API Calls Descriptions 71
 - 7.7.3 Sequence Diagrams 72
- 7.8 Monetisation Plan Designer 73
 - 7.8.1 Relation to Other Components 73
 - 7.8.2 API Calls Descriptions 74
 - 7.8.3 Sequence Diagrams 74
- 7.9 Data Usage and Intentions Analytics 75
 - 7.9.1 Relation to Other Components 75
 - 7.9.2 API Calls Descriptions 76
 - 7.9.3 Sequence Diagrams 76
- 8 Security, Trust & Privacy Preservation Bundle 77
 - 8.1 Anonymizer 77

- 8.1.1 Relation to Other Components 78
- 8.1.2 API Calls Descriptions 79
- 8.1.3 Sequence Diagrams 79
- 8.2 Lineage Tracker 80
 - 8.2.1 Relation to Other Components 80
 - 8.2.2 API Calls Descriptions 81
 - 8.2.3 Sequence Diagrams 81
- 8.3 GDPR checker 83
 - 8.3.1 Relation to Other Components 84
 - 8.3.2 API Calls Descriptions 84
 - 8.3.3 Sequence Diagrams 85
- 8.4 Searchable Encryption..... 85
 - 8.4.1 Relation to Other Components 86
 - 8.4.2 API Calls Descriptions 87
 - 8.4.3 Sequence Diagrams 87
- 8.5 Encryption/Decryption Engine 87
 - 8.5.1 Relation to Other Components 88
 - 8.5.2 API Calls Descriptions 88
 - 8.5.3 Sequence Diagrams 88
- 9 Transactions Services Bundle..... 90
 - 9.1 On/Off Platform Contract Inspector 90
 - 9.1.1 Relation to Other Components 91
 - 9.1.2 API Calls Descriptions 91
 - 9.1.3 Sequence Diagrams 92
 - 9.2 Transaction Auditor 93
 - 9.2.1 Relation to Other Components 93
 - 9.2.2 API Calls Descriptions 93
 - 9.2.3 Sequence Diagrams 94
 - 9.3 Smart Contract Template Composer 94
 - 9.3.1 Relation to Other Components 94
 - 9.3.2 API Calls Descriptions 95
 - 9.3.3 Sequence Diagrams 95
 - 9.4 Smart Contract Execution Engine..... 95
 - 9.4.1 Relation to Other Components 96

9.4.2	API Calls Descriptions	97
9.4.3	Sequence Diagrams	97
9.5	Data Factory User Wallet	98
9.5.1	Relation to Other Components	99
9.5.2	API Calls Descriptions	99
9.5.3	Sequence Diagrams	99
10	Ledgers Bundle.....	100
10.1	PISTIS Data Ledger.....	100
10.1.1	Relation to Other Components	100
10.1.2	API Calls Descriptions	101
10.1.3	Sequence Diagrams	101
10.2	PISTIS Monetary Ledger	101
10.2.1	Relation to Other Components	102
10.2.2	API Calls Descriptions	102
10.2.3	Sequence Diagrams	103
11	AI & Interoperability Repositories Bundle	104
11.1	PISTIS Models Repository.....	104
11.1.1	Relation to Other Components	105
11.1.2	API Calls Descriptions	105
11.1.3	Sequence Diagrams	106
11.2	Data Factory ML Models Repository.....	107
11.2.1	Relation to Other Components	107
11.2.2	API Calls Descriptions	107
11.2.3	Sequence Diagrams	108
11.3	AI Model Editor	109
11.3.1	Relation to Other Components	109
11.3.2	API Calls Descriptions	109
11.3.3	Sequence Diagrams	110
12	Identity and Access management Bundle	111
12.1	Identity Manager.....	111
12.1.1	Relation to Other Components	111
12.1.2	API Calls Descriptions	112
12.1.3	Sequence Diagrams	113
13	System Services Bundle	116

- 13.1 Data Factories Registrant 116
 - 13.1.1 Relation to Other Components 116
 - 13.1.2 API Calls Descriptions 117
 - 13.1.3 Sequence Diagrams 117
- 13.2 System and Activities Monitor 118
 - 13.2.1 Relation to Other Component..... 118
 - 13.2.2 API Calls Descriptions 118
 - 13.2.3 Sequence Diagrams 118
- 14 Conclusions 119

List of Figures

Figure 1: Macroscopic view of PISTIS Architecture	16
Figure 2: PISTIS Architecture	17
Figure 3: Data Check-In Sequence Diagram	22
Figure 4: Data Transformation Sequence Diagram	23
Figure 5: Job Configurator Sequence Diagram.....	25
Figure 6: Analytics Engine Facilitator Sequence Diagram	27
Figure 7: Data Enrichment Sequence Diagram	30
Figure 8: Metadata Quality Assessment Sequence Diagram	32
Figure 9: Data Quality Assessment Sequence Diagram	32
Figure 10: Data Insight Generator Sequence Diagram	34
Figure 11: Data Catalogues Sequence Diagram	39
Figure 12: Factory Data Storage Sequence Diagram.....	42
Figure 13: Querying Data Explorer Sequence Diagram.....	45
Figure 14: Insert/Update Dataset in LSH Storage Sequence Diagram.....	45
Figure 15: Delete Dataset from LSH Storage Sequence Diagram	46
Figure 16: Matchmaking Services Sequence Diagram	48
Figure 17: PISTIS Data Factory Connector Sequence Diagram	51
Figure 18: Smart Contract Checker High-level Architecture	52
Figure 19: Smart Contract Checker Sequence Diagram	53
Figure 20: Access Policy Editor Sequence Diagram #1 – User Verification.....	56
Figure 21: Access Policy Editor Sequence Diagram #2 – Client Creation.....	57
Figure 22: Access Policy Editor Sequence Diagram #3 – Policy Setting and Evaluation	58
Figure 23: Asset Description Bundler Sequence Diagram.....	61
Figure 24: FAIR Data Valuation Service Sequence Diagram.....	63
Figure 25: Digital DLT-FIAT Wallet Sequence Diagram - #1	66
Figure 26: Digital DLT-FIAT Wallet Sequence Diagram - #2	67
Figure 27: Digital DLT-FIAT Wallet Sequence Diagram - #3	67
Figure 28: Data Investment Planner Sequence Diagram	69
Figure 29: Market Insights Sequence Diagram	70
Figure 30: NFT Generator- Sequence Diagram	72
Figure 31: Monetisation Plan Designer - Sequence Diagram.....	74
Figure 32: Data Usage and Intentions Analytics- Sequence Diagram.....	76
Figure 33: Anonymiser Sequence Diagram	79
Figure 34: Lineage Tracker Sequence Diagram #1	81
Figure 35: Lineage Tracker Sequence Diagram #2	82
Figure 36: Lineage Tracker Sequence Diagram #3	82
Figure 37: Lineage Tracker Sequence Diagram #4	83
Figure 38: GDPR Checker high-level Architecture.....	84
Figure 39: GDPR checker Sequence Diagram.....	85
Figure 40: Searchable Encryption high-level Architecture.....	86
Figure 41: Searchable Encryption Sequence Diagram	87
Figure 42 Encryption/Decryption high-level Architecture	88
Figure 43: Encrypt/Decrypt Sequence Diagram.....	89

- Figure 44: On chain platform inspection..... 90
- Figure 45: On platform contract inspection 92
- Figure 46: Off platform contract inspection 92
- Figure 47: Transactions Auditor Sequence Diagram 94
- Figure 48: Smart Contract Template Composer - Sequence Diagram 95
- Figure 49: Smart Contract Execution Engine High-Level Architecture 96
- Figure 50: Smart Contract Execution Engine Sequence Diagram (Organisation Boundaries). 97
- Figure 51: Smart Contract Execution Engine Sequence Diagram (PISTIS Cloud Platform) 98
- Figure 52: Data Factory User Wallet High-Level Architecture 98
- Figure 53: Data Factory User Wallet Sequence Diagram 99
- Figure 54: PISTIS Data Ledger Sequence Diagram 101
- Figure 55: PISTIS Monetary Ledger Sequence Diagram 103
- Figure 56: PISTIS Models Repository - Sequence Diagram 106
- Figure 57: Factory ML Models Repo Sequence Diagram 108
- Figure 58: AI Model Editor Sequence Diagram 110
- Figure 59: Identity Manager Authentication Sequence Diagram 113
- Figure 60: Identity Manager Authorisation Sequence Diagram 113
- Figure 61: Identity Manager Resource management Sequence Diagram 114
- Figure 62: Identity Manager Data Factory registration Sequence Diagram 115
- Figure 63: Identity Manager Authorisation check for user, audience, and scope Sequence Diagram 115
- Figure 64: Identity Manager Access Policies retrieval for a Data Asset Sequence Diagram . 115
- Figure 65: Data Factories Registrant Sequence Diagram 117
- Figure 66: Resources and Activities Monitor- Sequence Diagram..... 118

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

PISTIS provides a comprehensive framework and a reference platform made up of connected deployments to unlock the full potential of the data economy. It aims to transform how data is handled, shared, and monetized. By incorporating top-notch methods and solutions from various fields such as data management, analytics, finance, crypto, and security, PISTIS creates a coordinated architecture. This architecture is designed to meet the needs of key stakeholders and achieve the broader goal of European Data Spaces. The ultimate aim is to facilitate efficient and reliable sharing and monetization of data assets.

The current deliverable presents the initial work performed towards defining the system design of the technological output of the project, that of the PISTIS platform, that consists of the *PISTIS Data Factory* (a deployment that is installed at the premises of each user), and of the *PISTIS Cloud Platform* (a centralised deployment controlling the overall environment). D4.1 relies on the previously collected technical requirements and user stories presented in the deliverable D1.2 as well as the preliminary work performed in work packages WP2 and WP3, which work towards defining in detail and delivering the different technical components of PISTIS.

As PISTIS is working on the technical development side following the agile development methodology¹, this work will be done in several iterations. At the end of each iteration a new version of the platform together with the updated architecture and relevant documentation will be published in the next WP4 deliverables. The development process will continue until the end of the project aiming at the release of the v1.00 of the platform that will drive post-project exploitation.

The publishing of this deliverable is an important milestone for the project giving a start the intensive implementational work.

1.1 DOCUMENT STRUCTURE

The document is structured as follows:

Section 1 is the introduction and this document structure description.

Section 2 provides a macroscopic view on the architecture and is followed by a detailed view of the architecture, revealing the different bundles and their internal components.

Sections 3 - 13 describe the different bundles and present details on the different components belonging to each bundle, as well as the APIs and the Sequence Diagrams of each component.

Finally, section 14 concludes the document.

¹ https://en.wikipedia.org/wiki/Agile_software_development - Accessed 29/1 2023

2 PISTIS HIGH LEVEL ARCHITECTURE

The he PISTIS architecture builds upon the initial high-level architectural concepts introduced early in the project. These concepts were inspired by the high-level architecture and component descriptions outlined in the project's Design of Architecture. The initial figures involve various bundles that were created to organize different sets of components. These bundles became focal points for discussions among technical partners and demonstrators when designing various usage scenarios, as seen in Work Package 1 (WP1). The bundles also played a key role in defining the necessary functionalities at both the platform level and the bundle/component level.

The user stories outlined in WP1 were the foundation for creating the PISTIS MVP (see PISTIS deliverable D1.2). In WP4, we worked on defining bundles that meet specific needs and consistently described how different parts of the system will work together to deliver the platform's anticipated services.

This section provides a high-level view on the overall PISTIS architecture, where the different bundles and components are revealed, and which are analysed in the next sections.

2.1 PISTIS ARCHITECTURE

As described in previous deliverables, 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 are operating a “PISTIS Data Factory” deployment at their premises (coloured green in the following figure) 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.

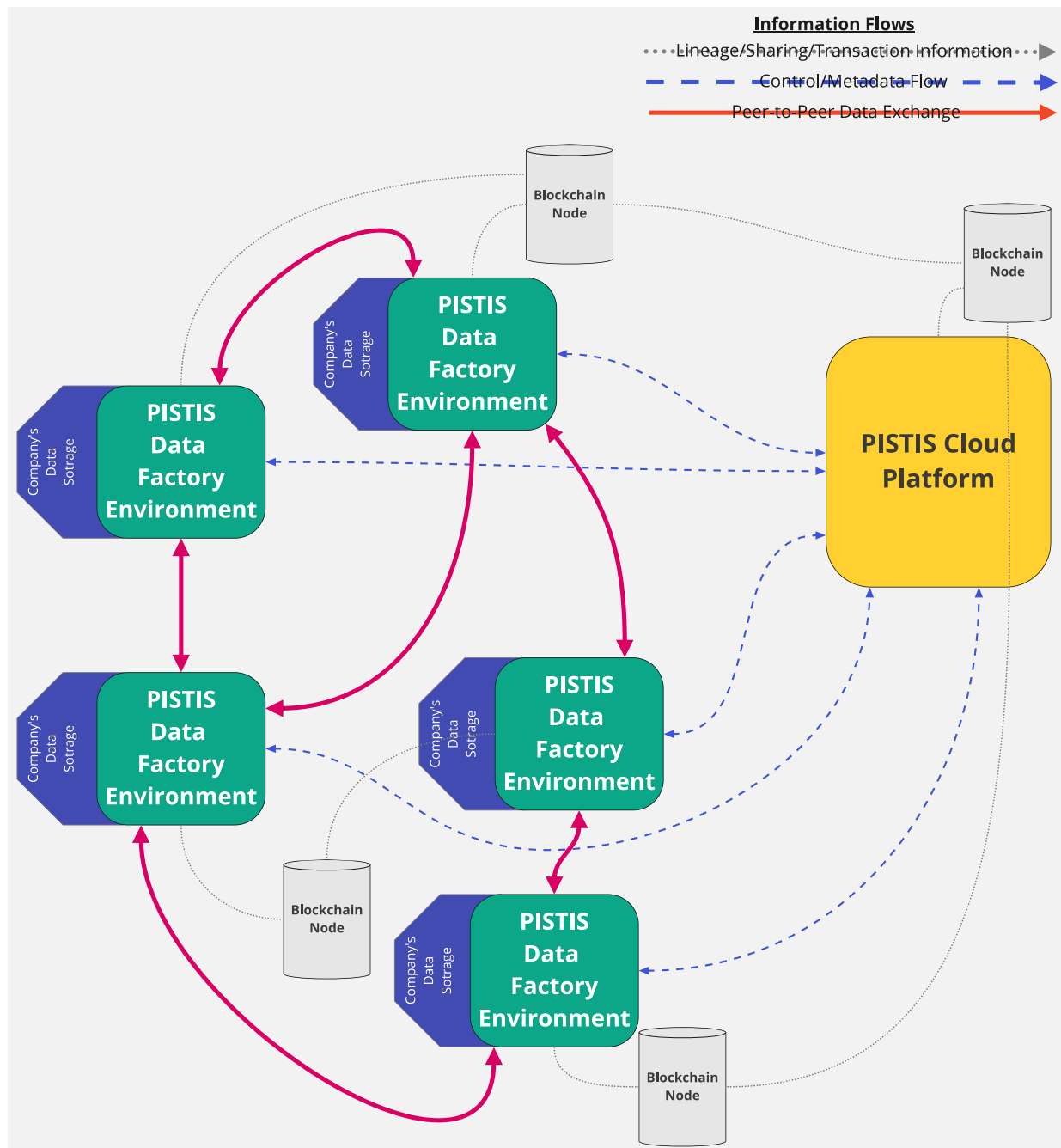


Figure 1: Macroscopic view of PISTIS Architecture

It needs to be 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. The next figure gives a closer look at the PISTIS architecture. It shows the various bundles and the components within them. Additionally, it illustrates the main data, metadata, and control flows across the entire platform.

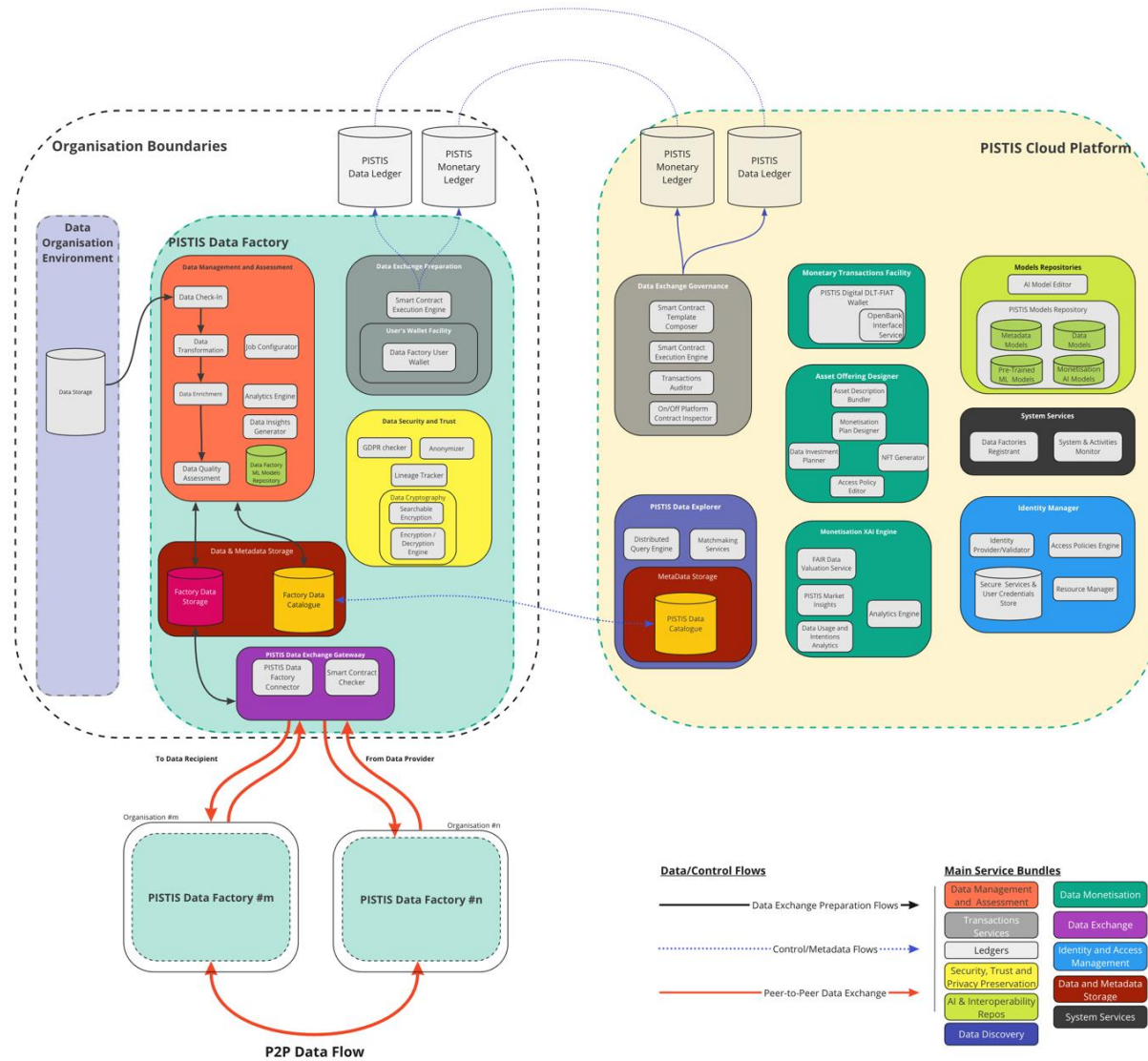


Figure 2: PISTIS Architecture

2.2 BUNDLES COMPOSING PISTIS

As shown in the architecture figure above, 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 are described in the following section, 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.

2.3 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.

3 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.

3.1 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 in order 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 in order to get access to the required data source.
- *GetDataFromSubscription*: the possibility to subscribe as a client to a Kafka topic² 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.

3.1.1 RELATION TO OTHER COMPONENTS

3.1.1.1 Components providing Input to Data Check-In

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

3.1.1.2 Components to which Data Check-In provides input

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

3.1.2 API CALLS DESCRIPTIONS

Data Check-In - OpenAPI 3.1 0.1 OAS 3.1

<https://raw.githubusercontent.com/asynccapi/spec/v2.6.0/examples/streetlights-kafka.yml>

[Terms of service](#)
[Contact the developer](#)
[Find out more about Swagger](#)

Servers

Authorize

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

3.1.3 SEQUENCE DIAGRAMS

² <https://kafka.apache.org/intro> - Accessed 29/12/2023

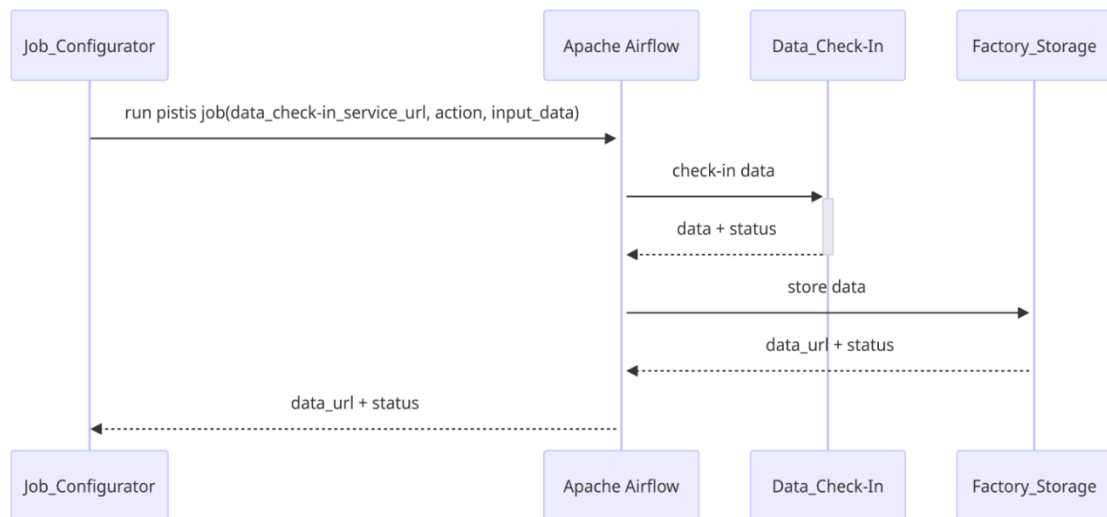


Figure 3: Data Check-In Sequence Diagram

3.2 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 in order 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.

3.2.1 RELATION TO OTHER COMPONENTS

3.2.1.1 Components providing Input to Data Transformation

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

3.2.1.2 Components to which Data Transformation provides input

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

3.2.2 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

Data transformations Data transformation services

[Find out more](#) ^

- POST /**transformation** Perform data transformations on a given dataset.
- GET /**transformationCatalog** Get available transformations

3.2.3 SEQUENCE DIAGRAMS

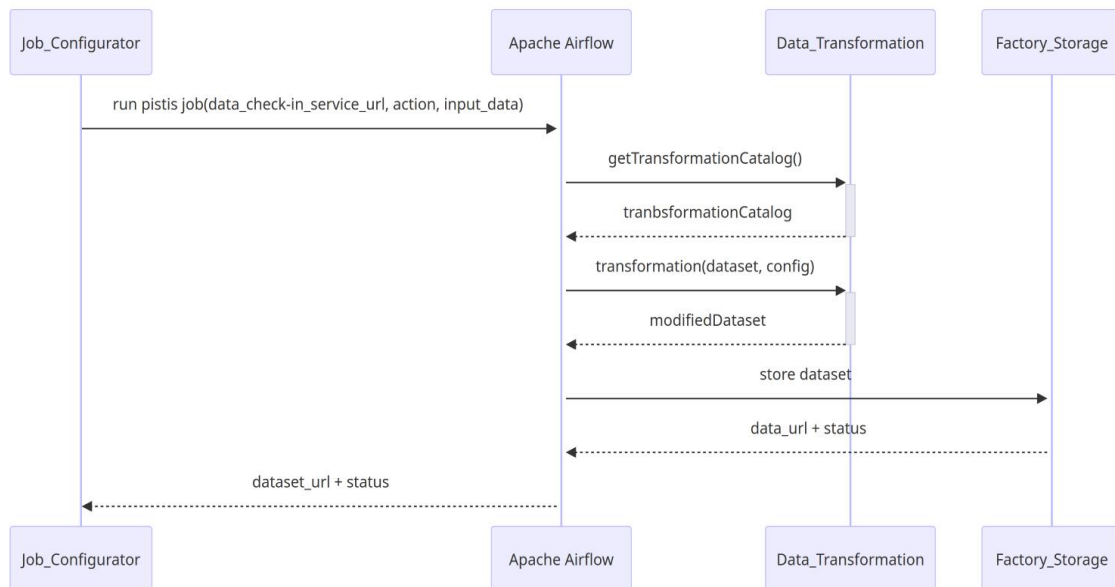


Figure 4: Data Transformation Sequence Diagram

3.3 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.

3.3.1 RELATION TO OTHER COMPONENTS

3.3.1.1 Components providing Input to Job Configurator

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

3.3.1.2 Components to which Job Configurator provides input

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

3.3.2 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

Authorize 

workflow Pitis workflow specification based on Pitis Job approach and supported by means of Airflow DAGs

[Find out more](#) ^

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

GET /workflow/runs List workflow runs 

3.3.3 SEQUENCE DIAGRAMS

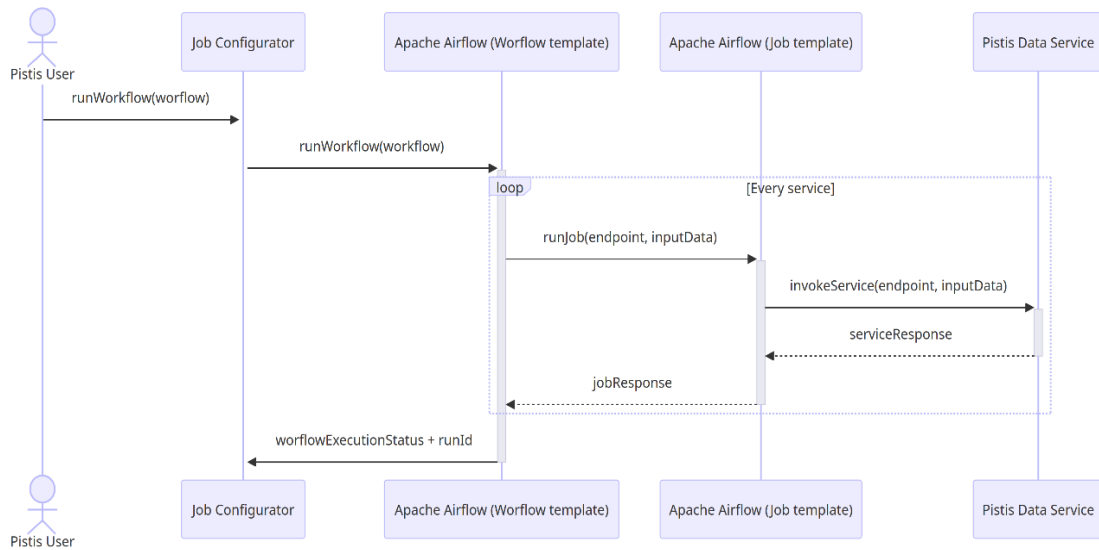


Figure 5: Job Configurator Sequence Diagram

3.4 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.

3.4.1 RELATION TO OTHER COMPONENTS

3.4.1.1 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

3.4.1.2 to which Analytics Engine provides input

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

3.4.2 API CALLS DESCRIPTIONS

Analytics Engine Facilitator 0.1 OAS 2.0

[Base URL: seas.datavaults.eu/]
<https://raw.githubusercontent.com/asynccapi/spec/v2.0.0/examples/streetlights-kafka.yml>

Authorize

On premise

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

Data Assets

GET /playground/cloud/storage/retieveDataSetCollection 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

3.4.3 SEQUENCE DIAGRAMS

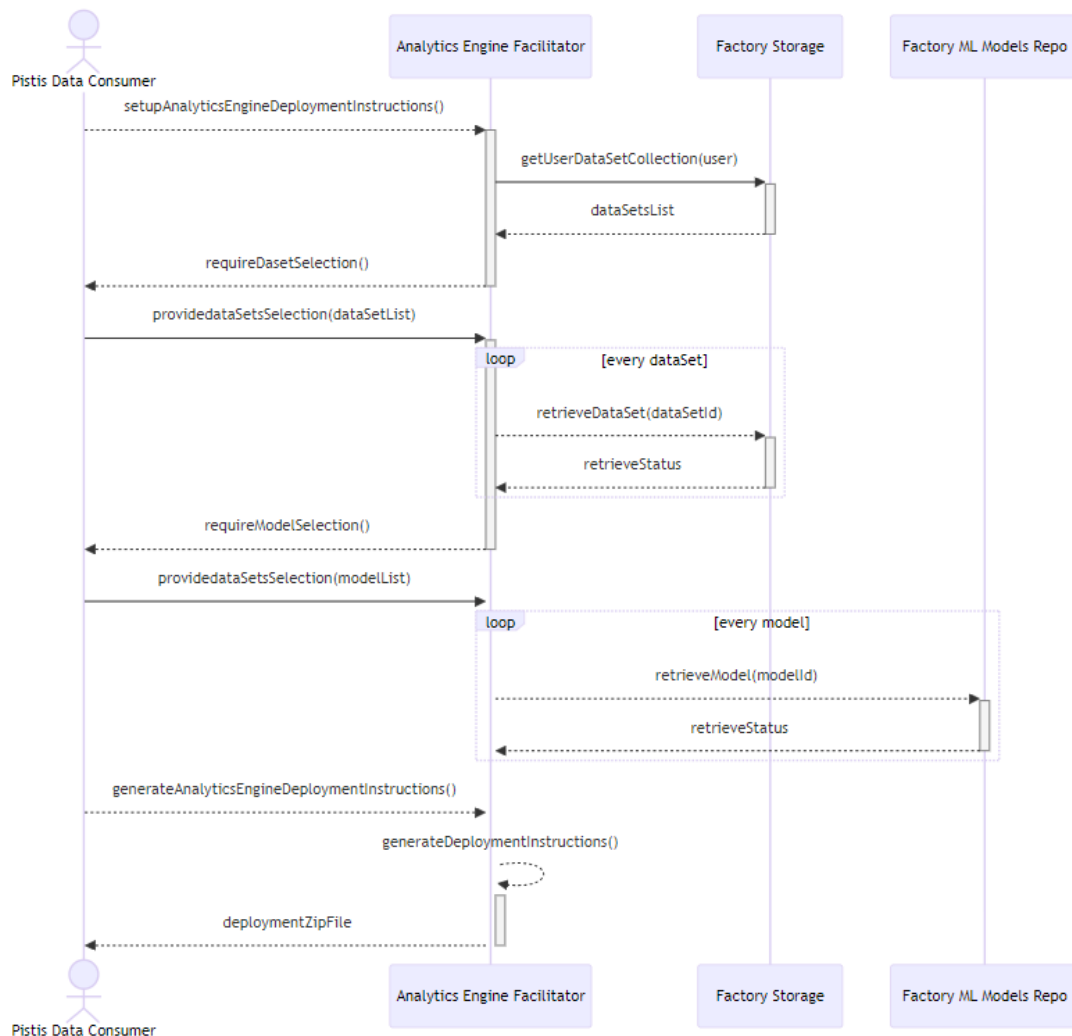


Figure 6: Analytics Engine Facilitator Sequence Diagram

3.5 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.

3.5.1 RELATION TO OTHER COMPONENTS

3.5.1.1 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

3.5.1.2 Components to which Data enrichment provides input

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

3.5.2 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	<code>/api/timeseries_analysis</code> Retrieve time series analysis results	∨
GET	<code>/api/text_analysis</code> Retrieve text analysis results using NLP	∨

Metadata Metadata enrichment



GET	<code>/api /knowledge_graph_completion</code>	Use knowledge graph completion methods to find missing information	∨
------------	---	--	---

Data Data enrichment



GET	<code>/api/data_aggregation</code>	Perform groupby operation on columns of a dataset	∨
------------	------------------------------------	---	---

GET	<code>/api /data_schema_matching</code>	Identify corresponding attributes in datasets and suggest matches	∨
------------	---	---	---

GET	<code>/api /data_schema_enrichment</code>	Use NLP topic modeling to predict column names	∨
------------	---	--	---

3.5.3 SEQUENCE DIAGRAMS

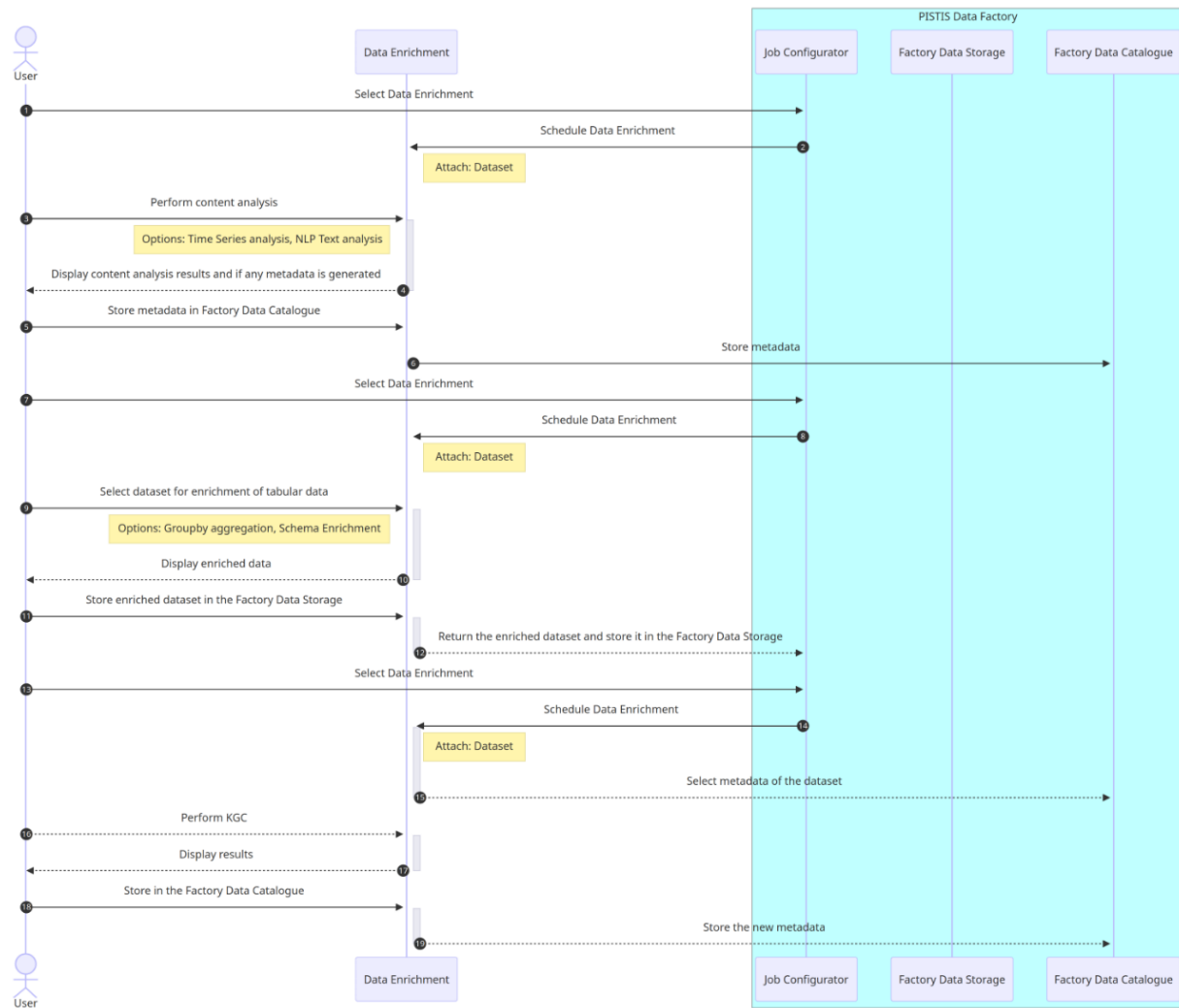


Figure 7: Data Enrichment Sequence Diagram

3.6 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 library³.

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.

3.6.1 RELATION TO OTHER COMPONENTS

3.6.1.1 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

3.6.1.2 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

3.6.2 API CALLS DESCRIPTIONS

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

GET `/dqa/assessment` 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/assessment` Start Quality Assessment ∨

³ <https://greatexpectations.io/> - Accessed 29/12/2023

3.6.3 SEQUENCE DIAGRAMS

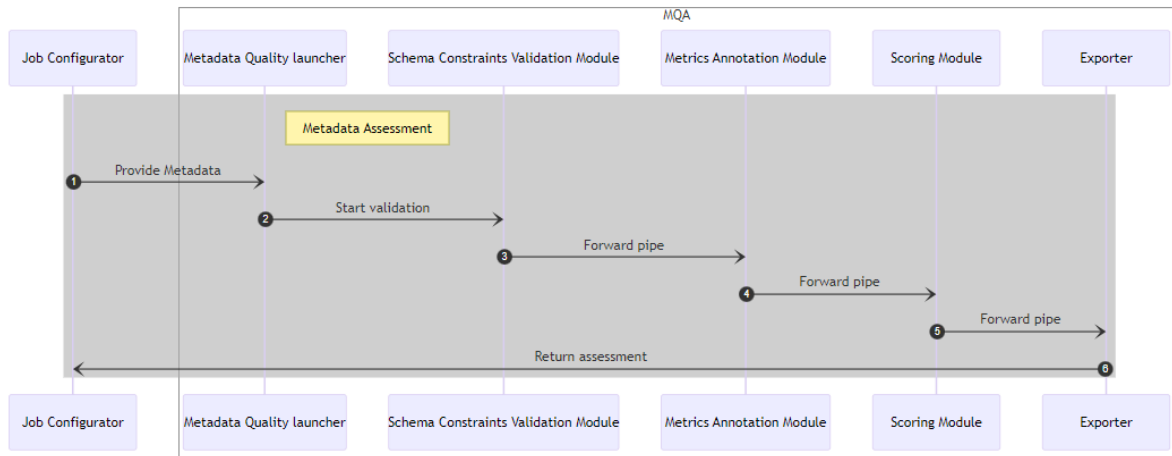


Figure 8: Metadata Quality Assessment Sequence Diagram

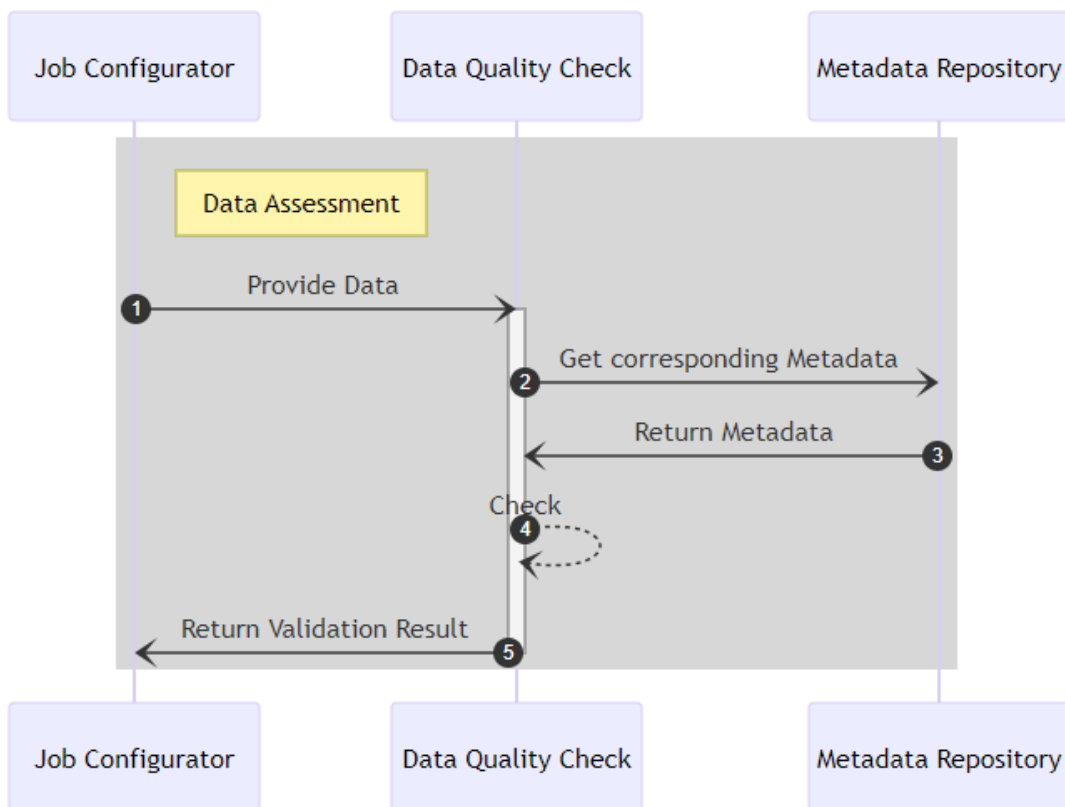


Figure 9: Data Quality Assessment Sequence Diagram

3.7 DATA INSIGHTS GENERATOR

The Data Insights generator is a component that provides information about the structure and data types of a given dataset in order 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.

3.7.1 RELATION TO OTHER COMPONENTS

3.7.1.1 Components providing Input to Data Insight Generator

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

3.7.1.2 Components to which Data Insight Generator provides input

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

3.7.2 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

▾

insights Data insight generation services

[Find out more](#) ^

POST Get insights report ▾

3.7.3 SEQUENCE DIAGRAMS

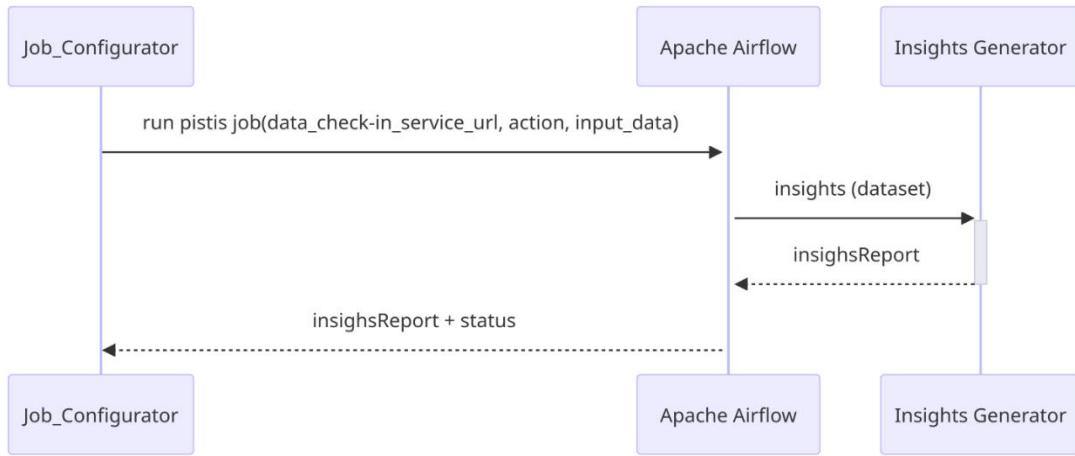


Figure 10: Data Insight Generator Sequence Diagram

4 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.

4.1 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.

4.1.1 RELATION TO OTHER COMPONENTS

4.1.1.1 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

4.1.1.2 Components to which Factory Data Catalogue provides input

Output Served	Component ingesting the output	Communication Method
Synced metadata	PISTIS Data Catalogue	API
Entire Metadata ⁴	PISTIS Data Factory Connector	API

4.1.1.3 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

4.1.1.4 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

4.1.1.5 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

4.1.2 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
 - o Each Factory will most likely only have one catalogue
 - o A Dataset is the metadata about a data asset
 - o A Distribution gives details about the actual access to the Factory Store
- PISTIS Data Catalogue
 - o A Catalogue represent a single data provider
 - o A Dataset is the metadata about a data asset
 - o A Distribution gives details about the actual access (e.g. via a Connector)

⁴ “Entire Metadata” refers to a complete metadata set for data asset, including the provided metadata and added metadata, e.g., through enrichment and data lineage 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	🔒 ↕

4.1.3 SEQUENCE DIAGRAMS

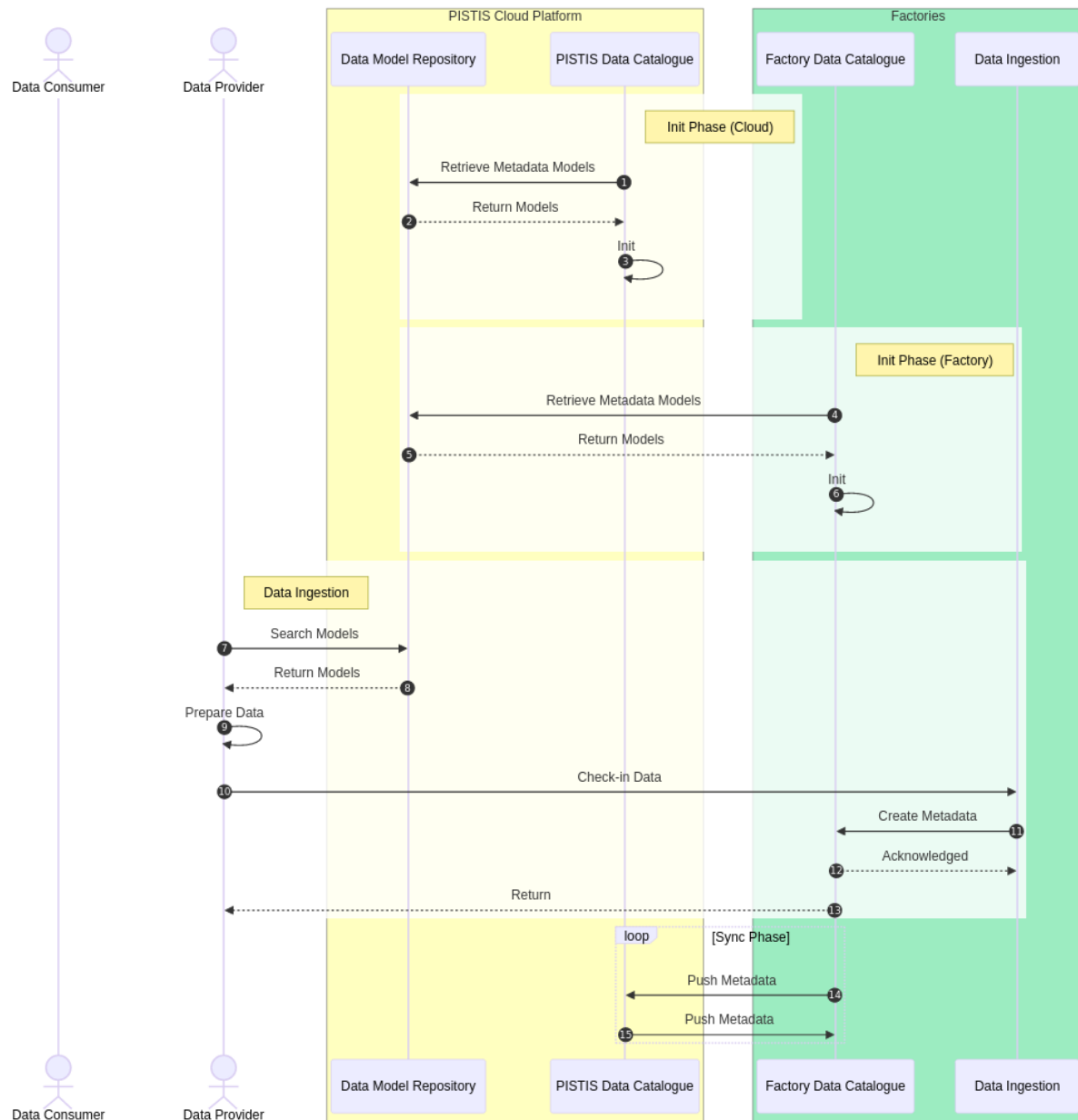


Figure 11: Data Catalogues Sequence Diagram

4.2 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 (for example 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

4.2.1 RELATION TO OTHER COMPONENTS

4.2.1.1 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

4.2.1.2 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

4.2.2 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 referred to as assets and all these assets are stored under a UUID, referred 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	∨

4.2.3 SEQUENCE DIAGRAMS

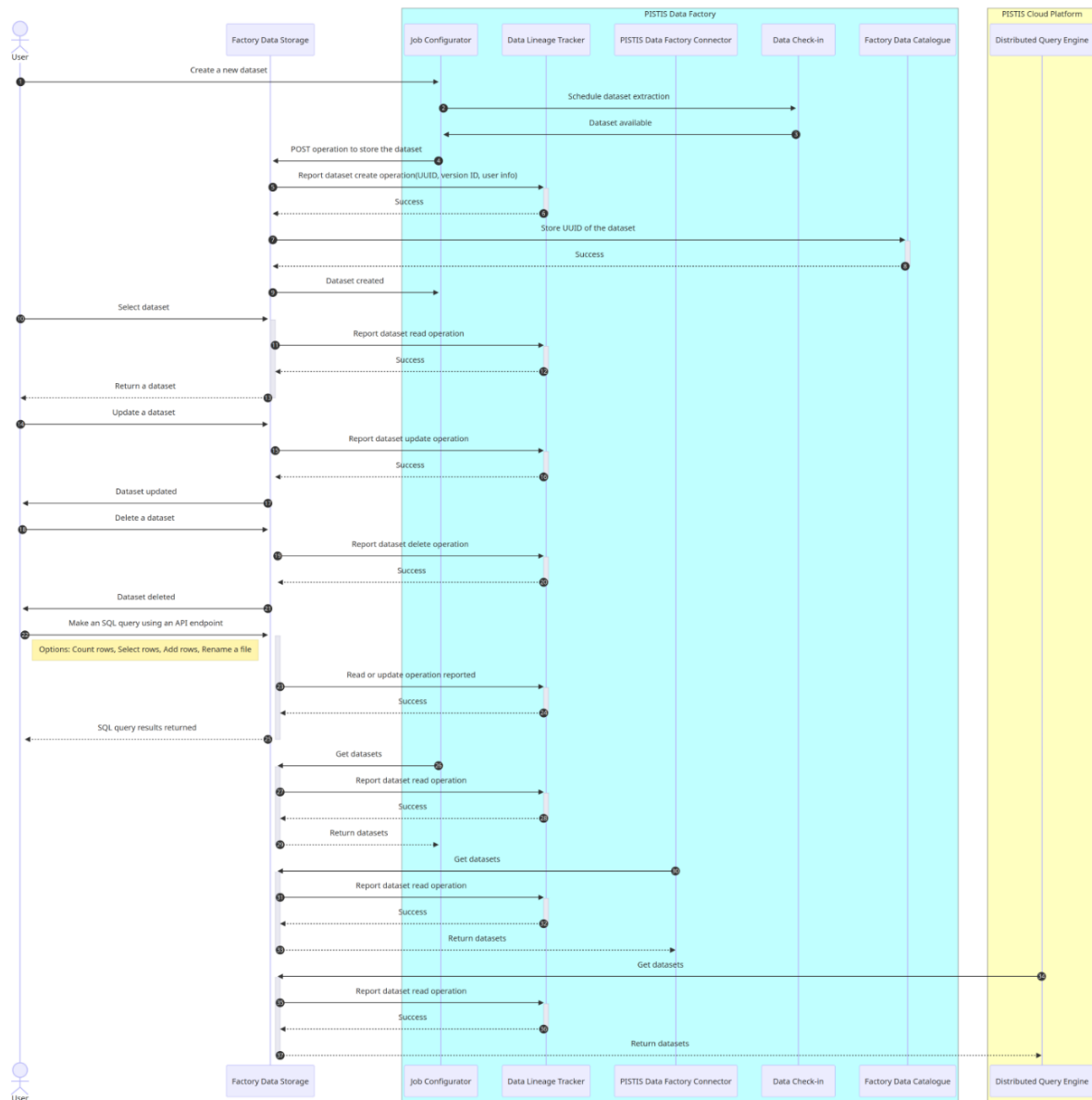


Figure 12: Factory Data Storage Sequence Diagram

5 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.

5.1 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.

5.1.1 RELATION TO OTHER COMPONENTS

5.1.1.1 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

5.1.1.2 Components to which Distributed Query Engine provides input

Output Served	Component ingesting the output	Communication Method
N/A		

5.1.2 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 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 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  ▾

5.1.3 SEQUENCE DIAGRAMS

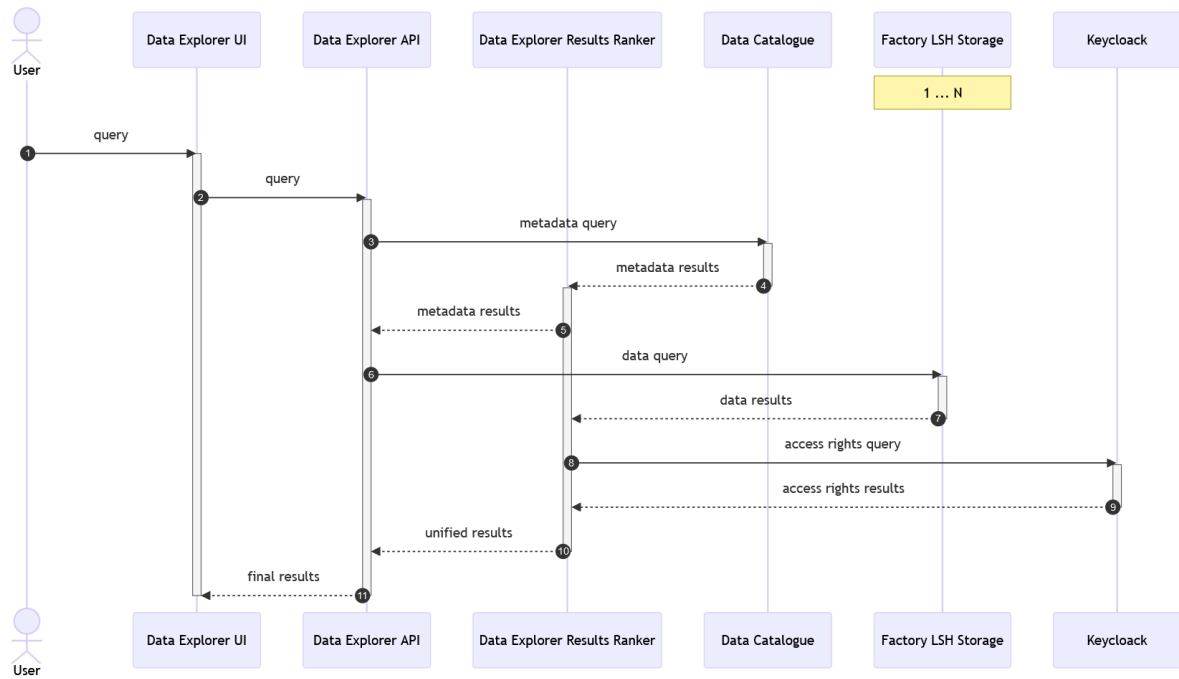


Figure 13: Querying Data Explorer Sequence Diagram

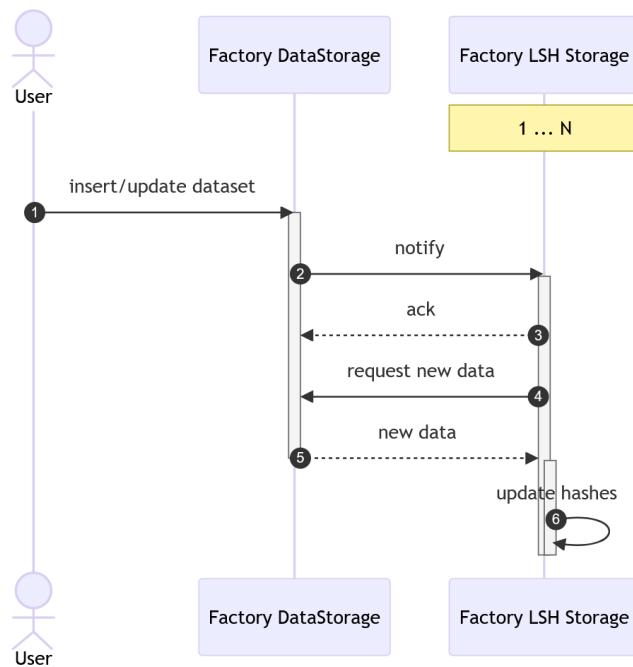


Figure 14: Insert/Update Dataset in LSH Storage Sequence Diagram

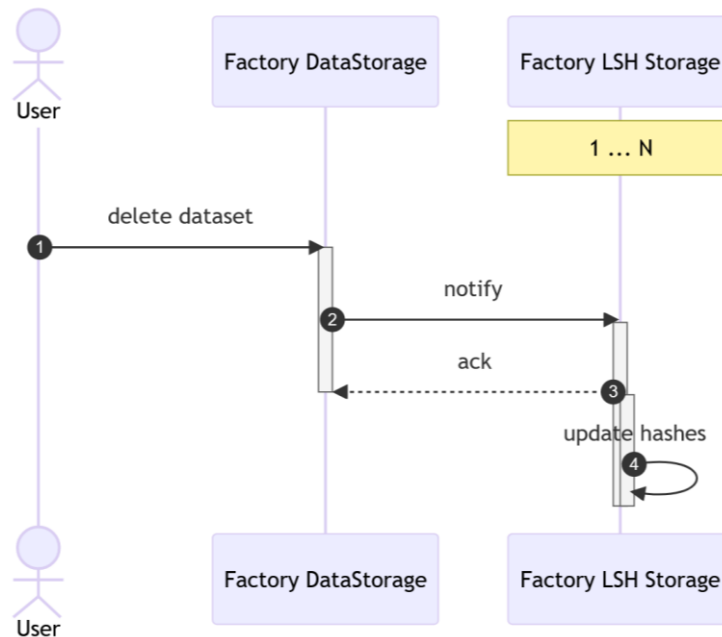


Figure 15: Delete Dataset from LSH Storage Sequence Diagram

5.2 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.

5.2.1 RELATION TO OTHER COMPONENTS

5.2.1.1 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

5.2.1.2 Components to which the Matchmaking Service provides input

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

5.2.2 API CALLS DESCRIPTIONS

Matchmaking Service 1.0.11 OAS 3.0

Servers

Authorize


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 ∨

5.2.3 SEQUENCE DIAGRAMS

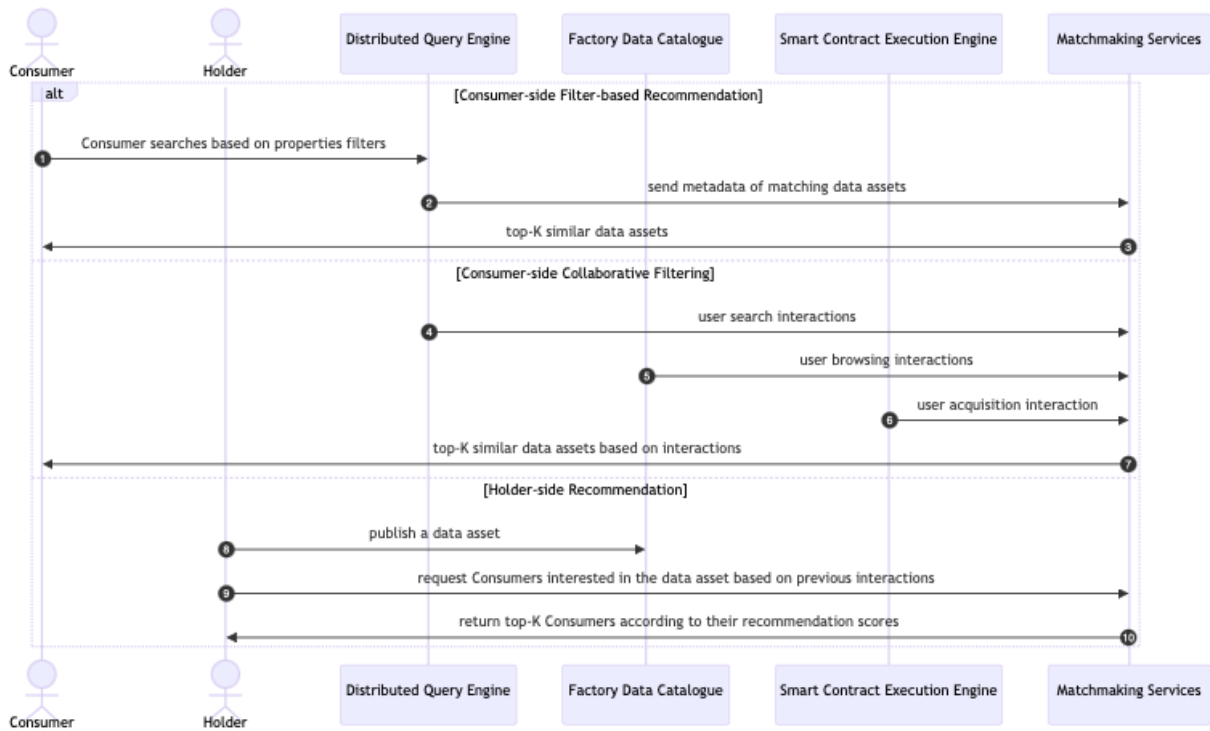


Figure 16: Matchmaking Services Sequence Diagram

6 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.

6.1 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 has to 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.

6.1.1 RELATION TO OTHER COMPONENTS

6.1.1.1 *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

6.1.1.2 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

6.1.2 API CALLS DESCRIPTIONS

PISTIS Data Factory Connector 1.0 OAS 3.0

PISTIS Data Factory Connector description

Servers

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

Authorize

Consumer ^

GET /retrieve-contract/retrieve/{id} Retrieve contract info

PUT /retrieve-contract/update/{id} Update contract info

GET /retrieve-data/{id} Retrieve asset data

Provider ^

GET /asset/metadata/{assetId} Retrieve asset metadata

GET /asset/{assetId} Retrieve asset data

GET /contract/{id} Retrieve contract info

6.1.3 SEQUENCE DIAGRAMS

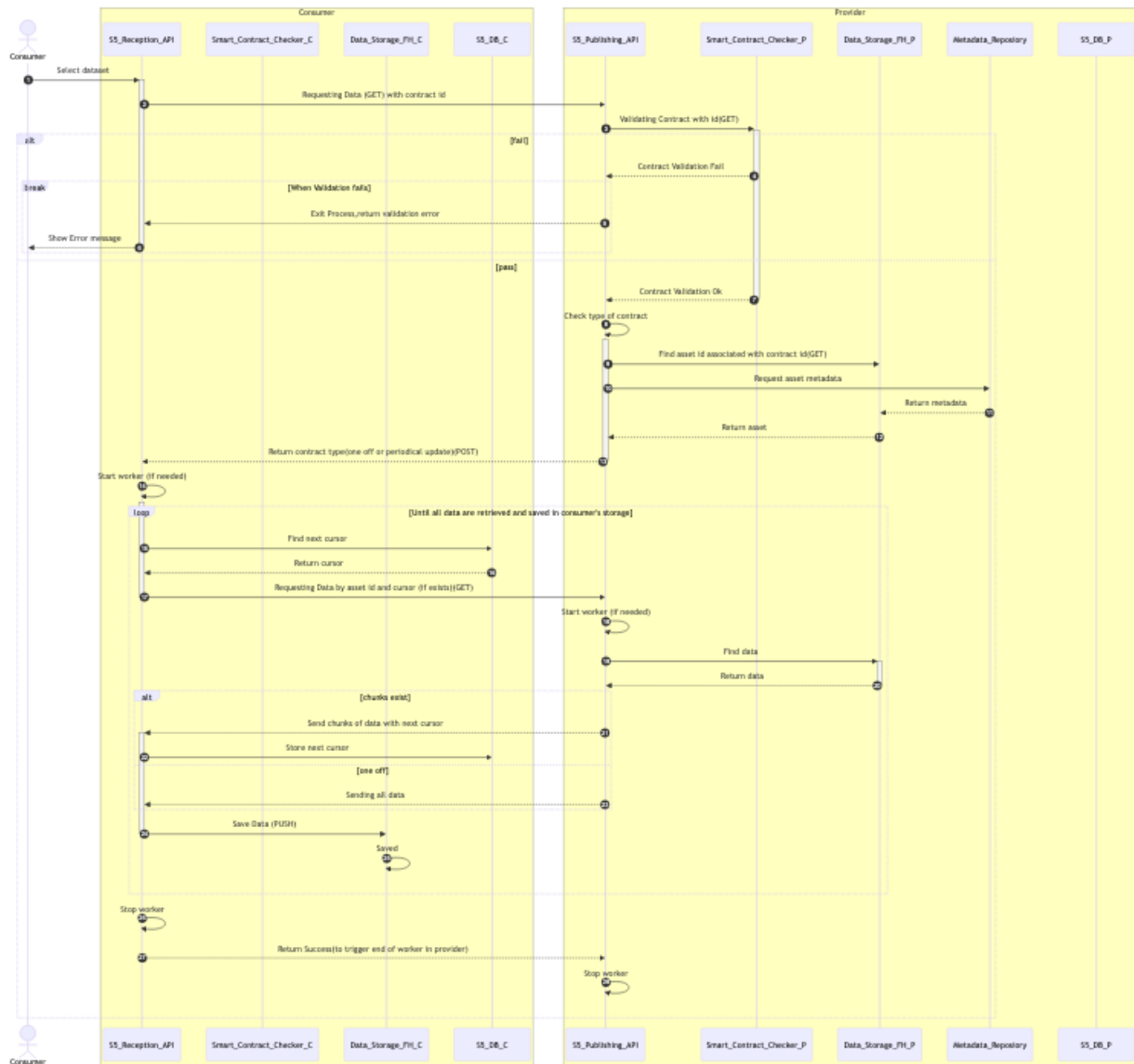


Figure 17: PISTIS Data Factory Connector Sequence Diagram

6.2 SMART CONTRACT CHECKER

The Smart Contract Checker within the PISTIS platform is a sophisticated component designed to ensure the integrity and validity of transactions on the network. The Smart Contract Checker’s checking policy is based on two pillars that collectively uphold the security, authenticity, and compliance of operations on the platform:

- *Checking the authenticity of transactions* (they originate from a valid, authenticated, and registered member of the PISTIS Platform). In other words, there is a need to check

if the smart contract operations are valid and performed by an authorised user by checking all the signatures. In this case two checks will be performed. The first one is if the user has valid verifiable credentials (stored in the wallet) and the second one is if the user is valid and authenticated user of PISTIS platform.

- *Make sure there is no violation in the execution of future data transactions*, in the context of functional logic. For instance, this can be done by checking the common privacy protection profile and the exposed sensitive columns or the existing balance of money prior to data trading.

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.

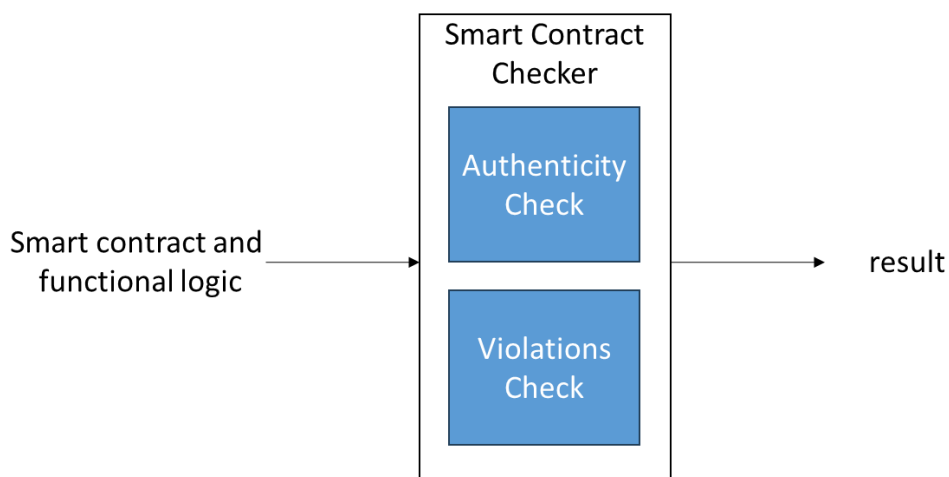


Figure 18: Smart Contract Checker High-level Architecture

6.2.1 RELATION TO OTHER COMPONENTS

6.2.1.1 Components providing Input to Smart Contract Checker component

Input Required	Component providing the Input	Communication Method
User credentials for Smart Contract Verification	Data Factory User Wallet	Internal function
Smart Contract to be checked and its functional logic	Smart Contract Execution Engine	API

6.2.1.2 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	API

6.2.2 API CALLS DESCRIPTIONS

Smart Contract Checker

GET /checkFunctionViolations Check Function violations

6.2.3 SEQUENCE DIAGRAMS

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

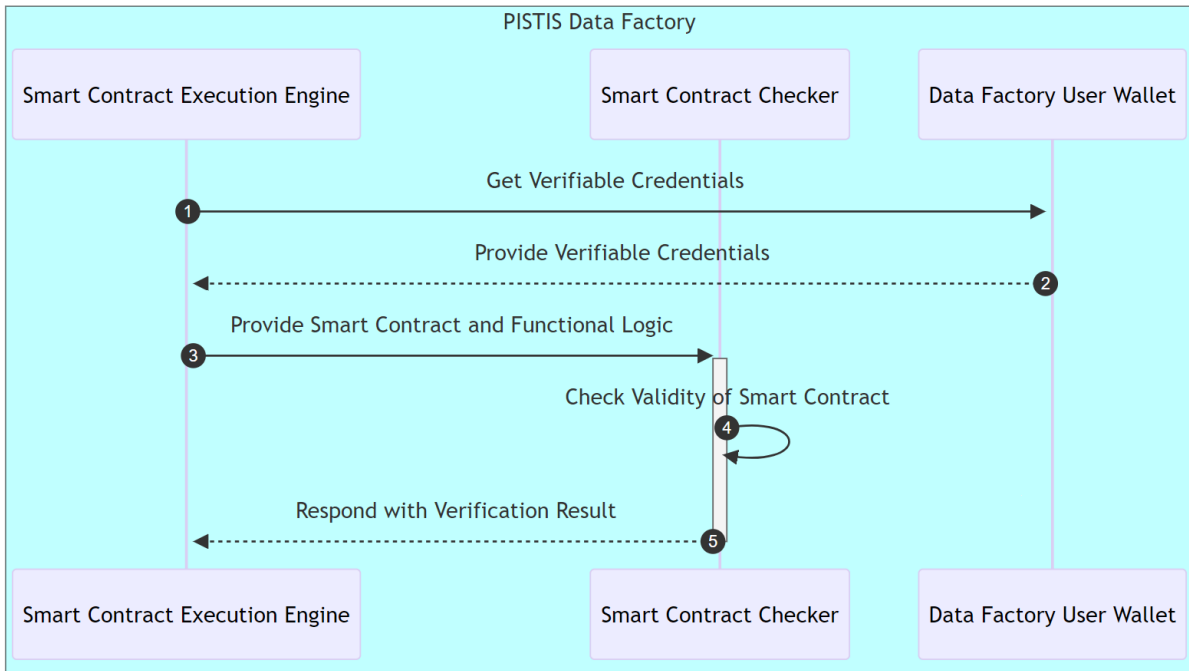


Figure 19: Smart Contract Checker Sequence Diagram

7 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.

7.1 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.

7.1.1 RELATION TO OTHER COMPONENTS

7.1.1.1 Components providing Input to Access Policy Editor

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

7.1.1.2 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

7.1.2 API CALLS DESCRIPTIONS

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

7.1.3 SEQUENCE DIAGRAMS

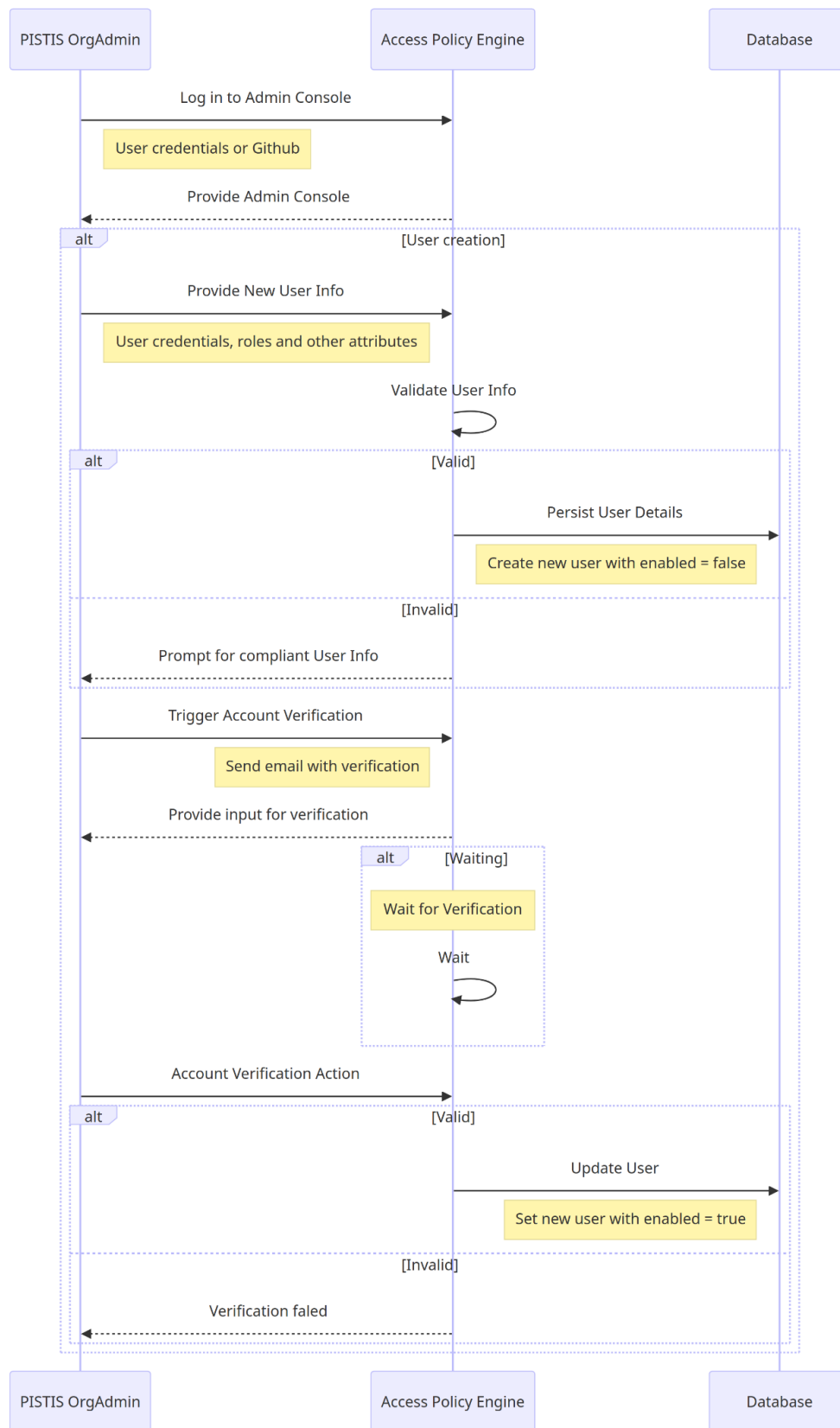


Figure 20: Access Policy Editor Sequence Diagram #1 – User Verification

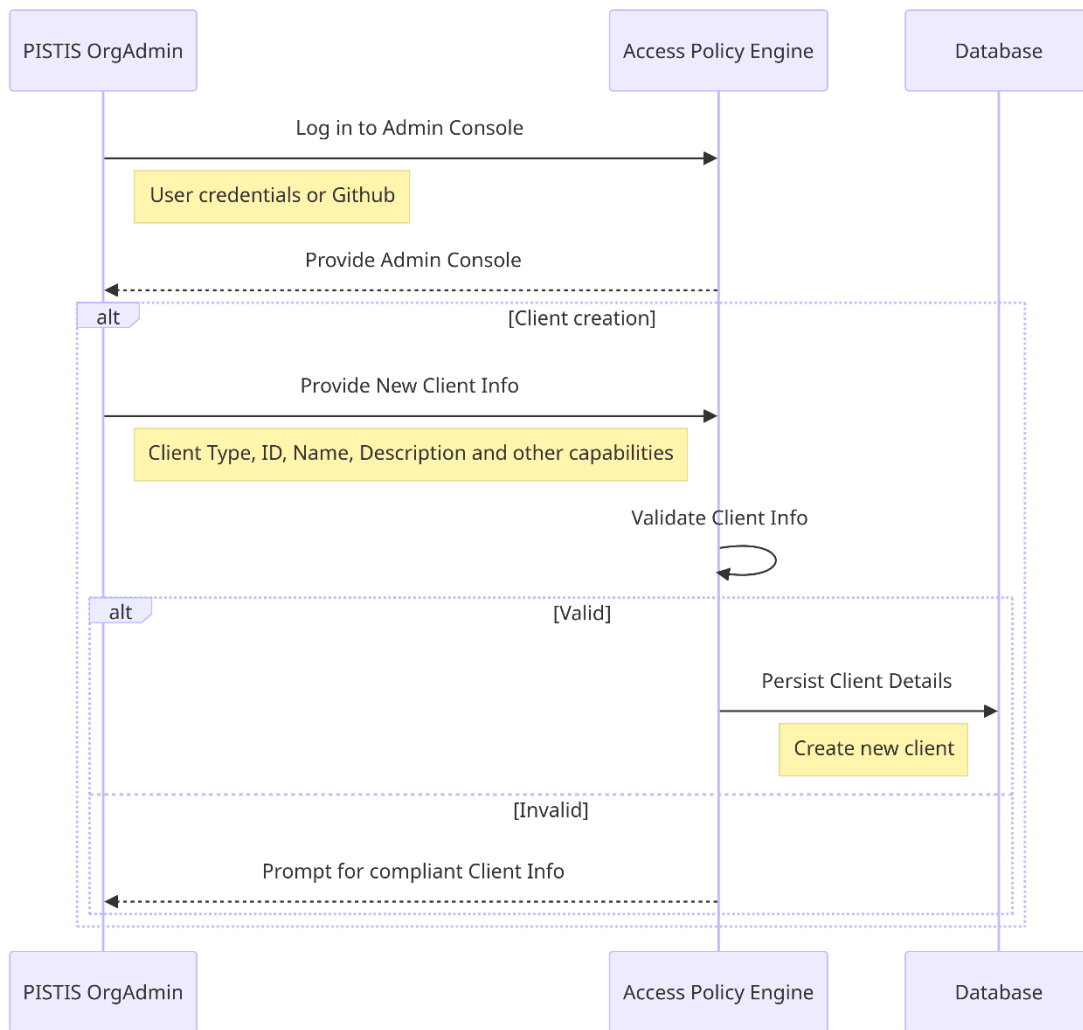


Figure 21: Access Policy Editor Sequence Diagram #2 – Client Creation

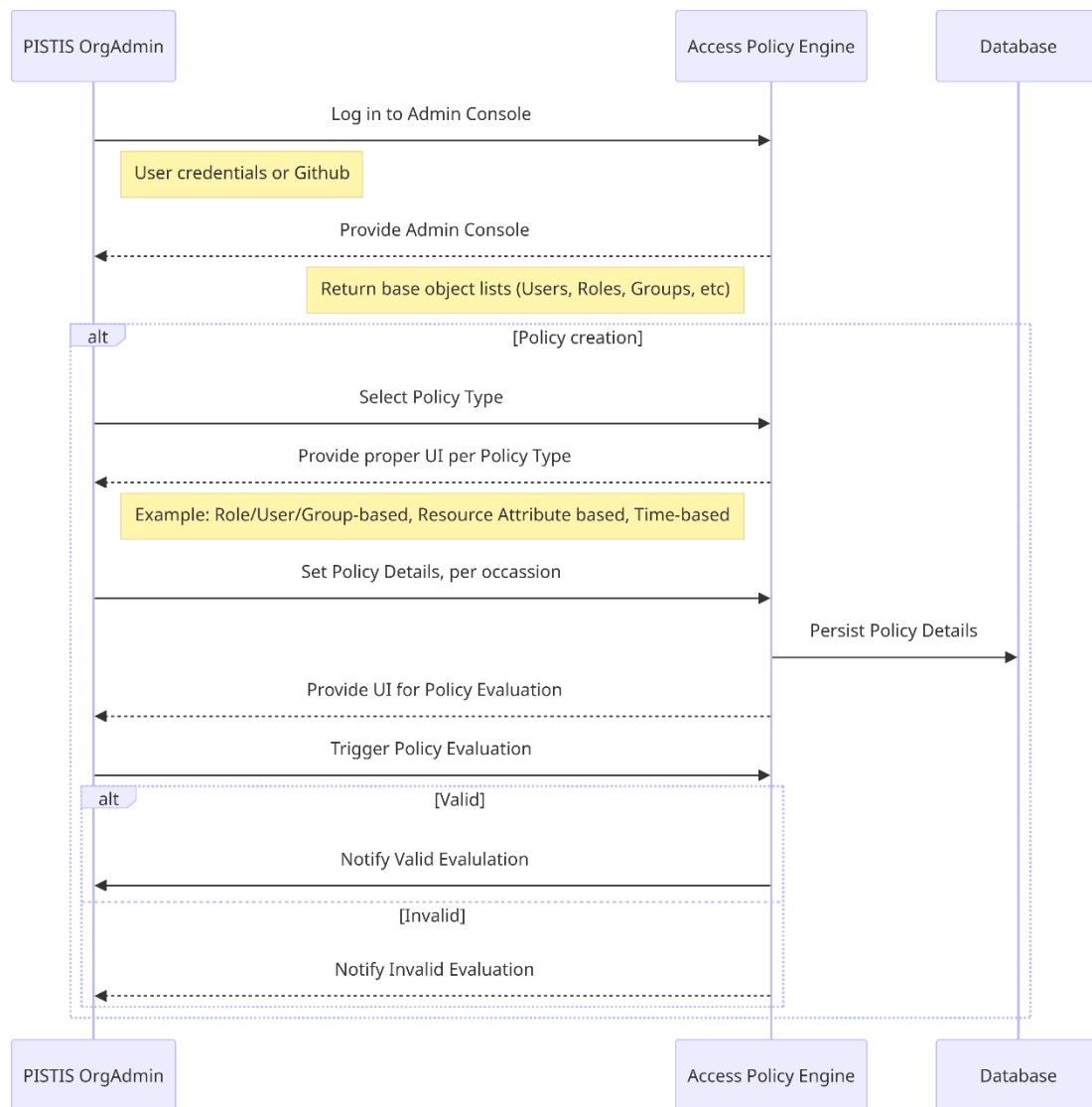


Figure 22: Access Policy Editor Sequence Diagram #3 – Policy Setting and Evaluation

7.2 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.

7.2.1 RELATION TO OTHER COMPONENTS

7.2.1.1 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

7.2.1.2 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

7.2.2 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	

7.2.3 SEQUENCE DIAGRAMS

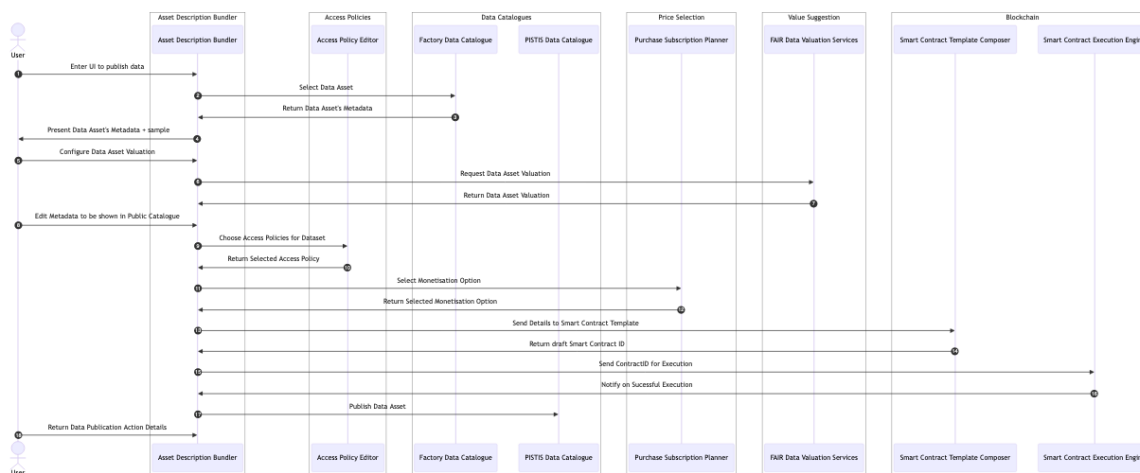


Figure 23: Asset Description Bundler Sequence Diagram

7.3 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).

7.3.1 RELATION TO OTHER COMPONENTS

7.3.1.1 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

7.3.1.2 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

7.3.2 API CALLS DESCRIPTIONS

FAIR Data Valuation Service 1.0.11 OAS 3.0

Servers

[Authorize](#)

dataValuation Operations about data valuation objects

[Find out more](#)

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

GET	/user/{userId}/dataValuations Finds data valuations by userID	🔒
------------	--	---

7.3.3 SEQUENCE DIAGRAMS

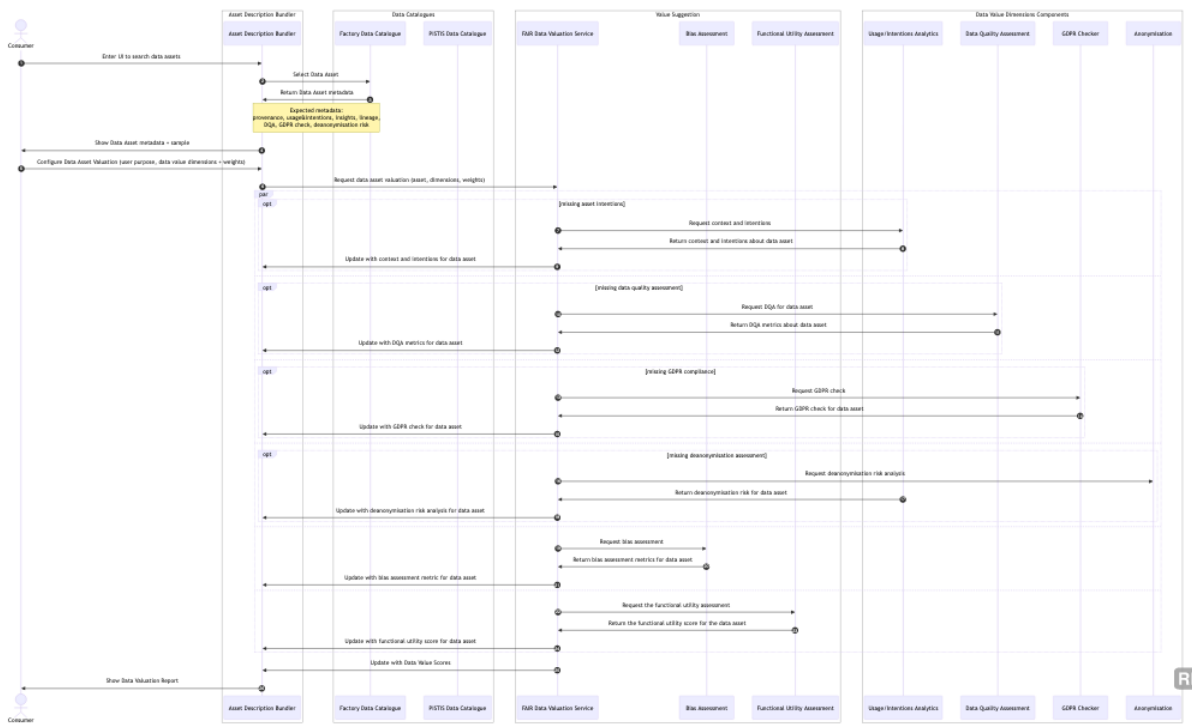


Figure 24: FAIR Data Valuation Service Sequence Diagram

7.4 PISTIS DIGITAL DLT-FIAT WALLET

This component is responsible for facilitating value transactions using cryptocurrencies within a private network, guaranteeing quick and reliable exchanges. It also plays a pivotal role in interfacing with traditional FIAT banking accounts⁵, enabling seamless fund transfers within the system and facilitating conversion between FIAT and cryptocurrencies. Cryptocurrency transactions are processed within a private DLT network maintaining a 1:1 ratio exchange between FIAT money and PISTIS crypto-coins, without the need for a liquidity mechanism.

7.4.1 RELATION TO OTHER COMPONENTS

7.4.1.1 Components providing Input to Digital DLT-FIAT Wallet

Input Required	Component providing the Input	Communication Method
User Authentication Token	Data Factory User Wallet	API
Token Validation Result	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 transactions (address, amount ...)	Data Factory User Wallet	API

⁵ FIAT currencies include the US Dollar (USD), Euro (EUR), and other government-issued currencies that are not backed by a physical commodity like gold or silver

Information about smart contract management (managing terms, validation committee...)	Data Factory User Wallet	API
---	--------------------------	-----

7.4.1.2 Components to which Digital DLT-FIAT Wallet provides input

Output Served	Component ingesting the output	Communication Method
User Authentication Token (Sent towards Keycloak for validation)	Identity Provider	API
Result of FIAT and DLT wallet creation / deletion	Data Factory User Wallet	API
Information about FIAT and DLT wallet balance and digital assets	Data Factory User Wallet	API
Information about FIAT and DLT wallet history/previous actions	Data Factory User Wallet	API
Currency conversion result	Data Factory User Wallet	API
Result of DLT wallet's addresses management (creation / deletion)	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
Information about current active smart contracts	Data Factory User Wallet	API
Information about component's current Health	Data Factory User Wallet	API
Active smart contract information (chains, validation committee, chain quorum)	Data Factory User Wallet	API
Information about the outcome of smart contract management (function execution, validation committee management...)	Data Factory User Wallet	API

7.4.2 API CALLS DESCRIPTIONS

The image shows a Swagger UI interface for the 'pistis_api' (version 0.1.0, OAS 3.0). The API is identified as a 'Management API'. The endpoints are organized into three main sections: Health Checker Endpoint, Transactions, and Wallets. Each endpoint is listed with its HTTP method (GET, POST, DELETE) and the corresponding path. The endpoints are color-coded: blue for GET, green for POST, and red for DELETE. Each entry has a dropdown arrow on the right side.

Method	Endpoint
GET	/api/healthchecker
POST	/api/transaction-create
GET	/api/transactions
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}
DELETE	/api/wallets/{wallet_id}

7.4.3 SEQUENCE DIAGRAMS

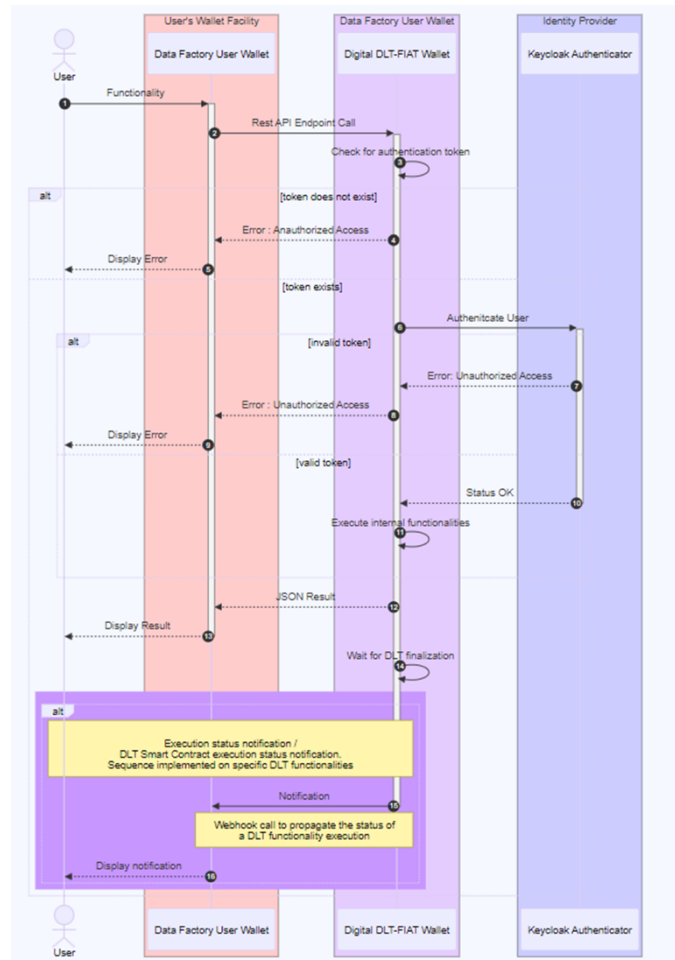


Figure 25: Digital DLT-FIAT Wallet Sequence Diagram - #1

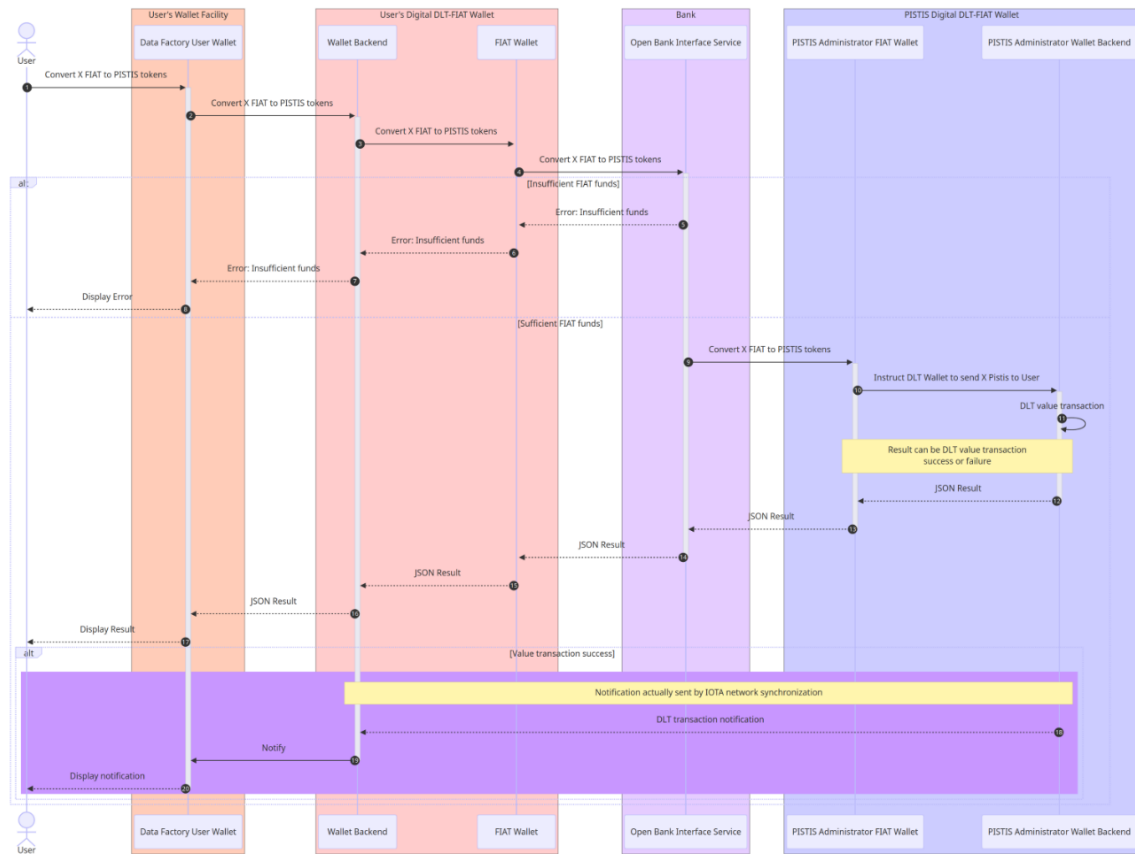


Figure 26: Digital DLT-FIAT Wallet Sequence Diagram - #2

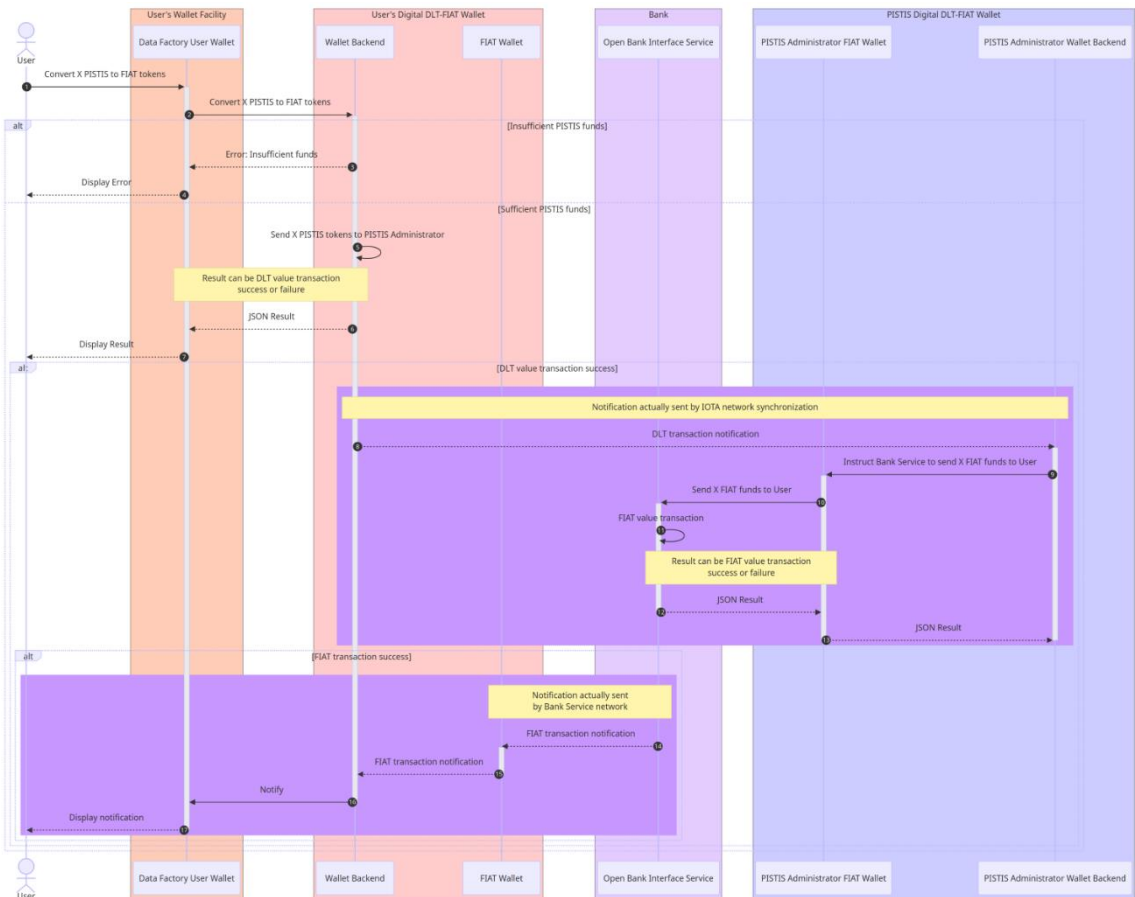


Figure 27: Digital DLT-FIAT Wallet Sequence Diagram - #3

7.5 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.

7.5.1 RELATION TO OTHER COMPONENTS

7.5.1.1 Components providing Input to Data Investment Planner

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

7.5.1.2 Components to which Data Investment Planner provides input

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

7.5.2 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

7.5.3 SEQUENCE DIAGRAMS

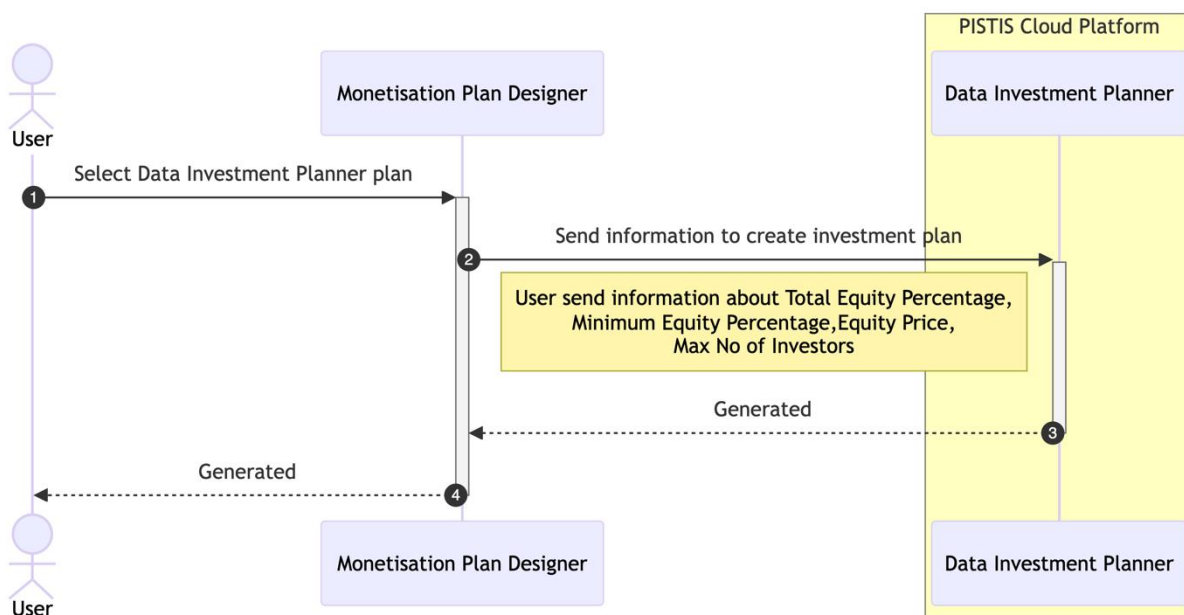


Figure 28: Data Investment Planner Sequence Diagram

7.6 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.

7.6.1 RELATION TO OTHER COMPONENTS

7.6.1.1 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

7.6.1.2 Components to which PISTIS Market Insights provides input

Output Served	Component ingesting the output	Communication Method
N/A		

7.6.2 API CALLS DESCRIPTIONS

Market Insights 1.0 OAS 3.0

Component for the facilitation of Market Insights

Servers
 Authorize

default ^

- POST /request-new-query/ Generate new query v
- POST /save-results/ Save query results v
- GET /run-query/{queryId} Run new Query v

7.6.3 SEQUENCE DIAGRAMS

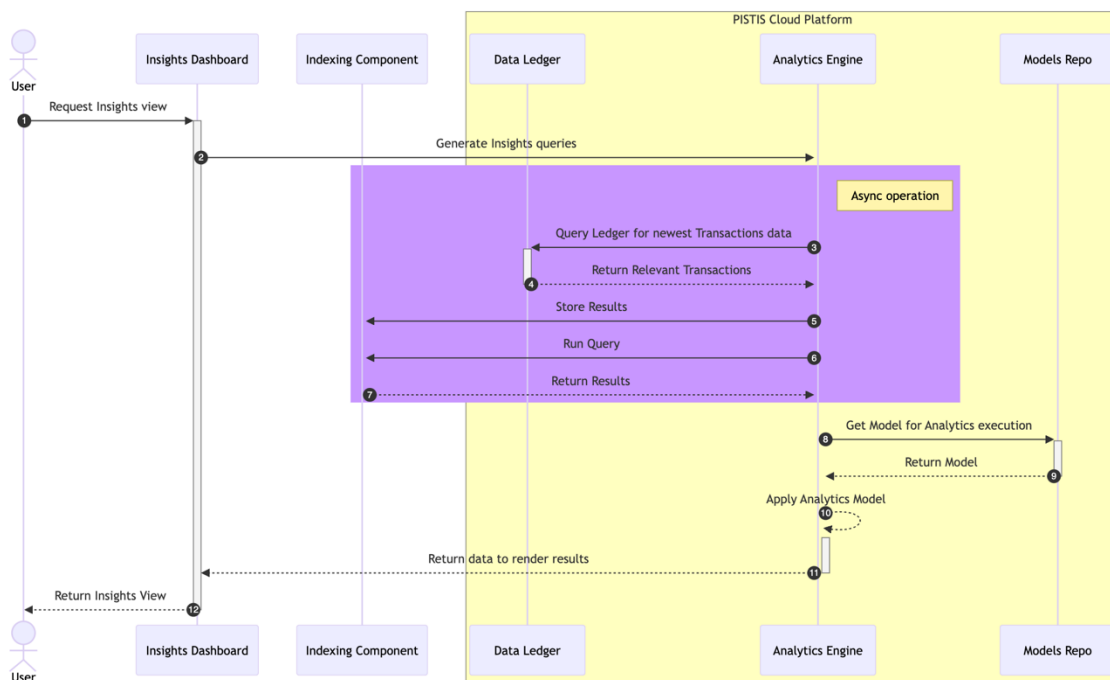


Figure 29: Market Insights Sequence Diagram

7.7 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, in order to verify ownership and be able to trade this NFT for complete ownership change of a Data Asset

This component is triggered by the Monetisation Plan Designer once the user selects to sell a dataset he /she own as an NFT. The Monetisation Plan Designer provides all the required info to the NTF Generator towards generating an NFT, and the latter is using the smart contract facility to generate the NFT and place it on the ledger, and the make it available for sale.

7.7.1 RELATION TO OTHER COMPONENTS

7.7.1.1 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 address returned upon execution	Smart Contract Execution Engine	API

7.7.1.2 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

7.7.2 API CALLS DESCRIPTIONS

NFT Generator Gateway 1.0 OAS 3.0

Gateway for the facilitation of NFT generator

Servers

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

Authorize

External ^

POST /generate/ Request new NFT v

Schemas ^

NFTGeneratorDTO >
←

NFTGeneratorReponse >
←

7.7.3 SEQUENCE DIAGRAMS

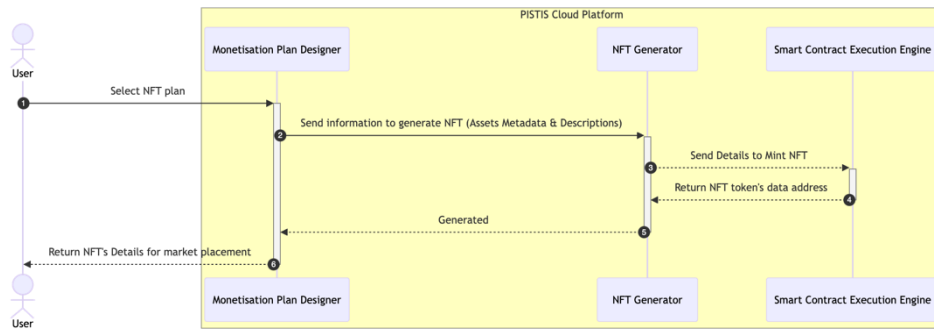


Figure 30: NFT Generator- Sequence Diagram

7.8 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.

7.8.1 RELATION TO OTHER COMPONENTS

7.8.1.1 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

7.8.1.2 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

7.8.2 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 Authorize

default ^

- GET **/asset-id/** Retrieve Asset Id v
- GET **/valuation/{assetId}** Retrieve valuation v
- POST **/bundler/** Add dataset to Asset Description Bundler v

7.8.3 SEQUENCE DIAGRAMS

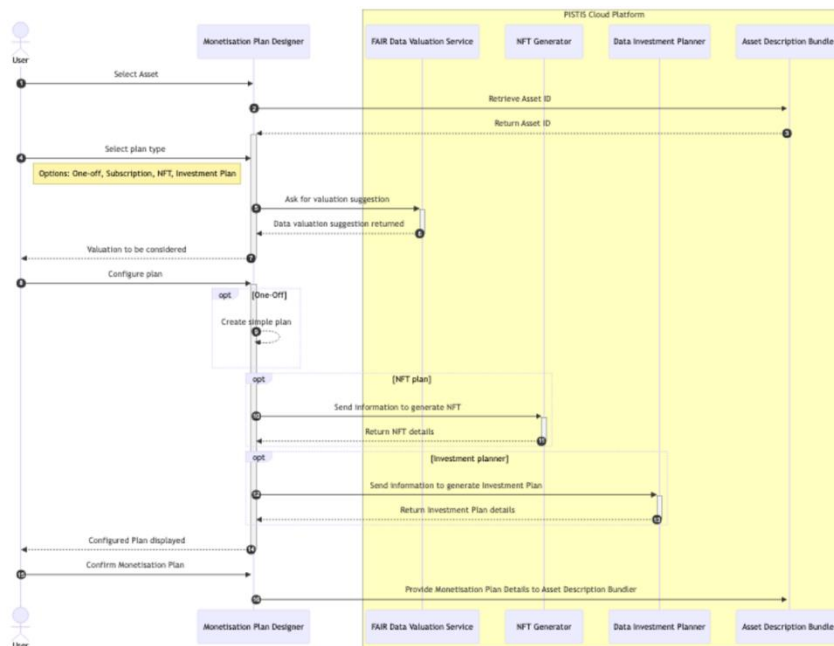


Figure 31: Monetisation Plan Designer - Sequence Diagram

7.9 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 manner in which 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.

7.9.1 RELATION TO OTHER COMPONENTS

7.9.1.1 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

7.9.1.2 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

7.9.2 API CALLS DESCRIPTIONS

Data Usage and Intentions Analytics 1.0 OAS 3.0

Component for the facilitation of Data Usage and Intentions Analytics

Servers: Authorize

default ^

POST
/save-questionnaire/{questionnaireId}
Save questionnaire
v

7.9.3 SEQUENCE DIAGRAMS

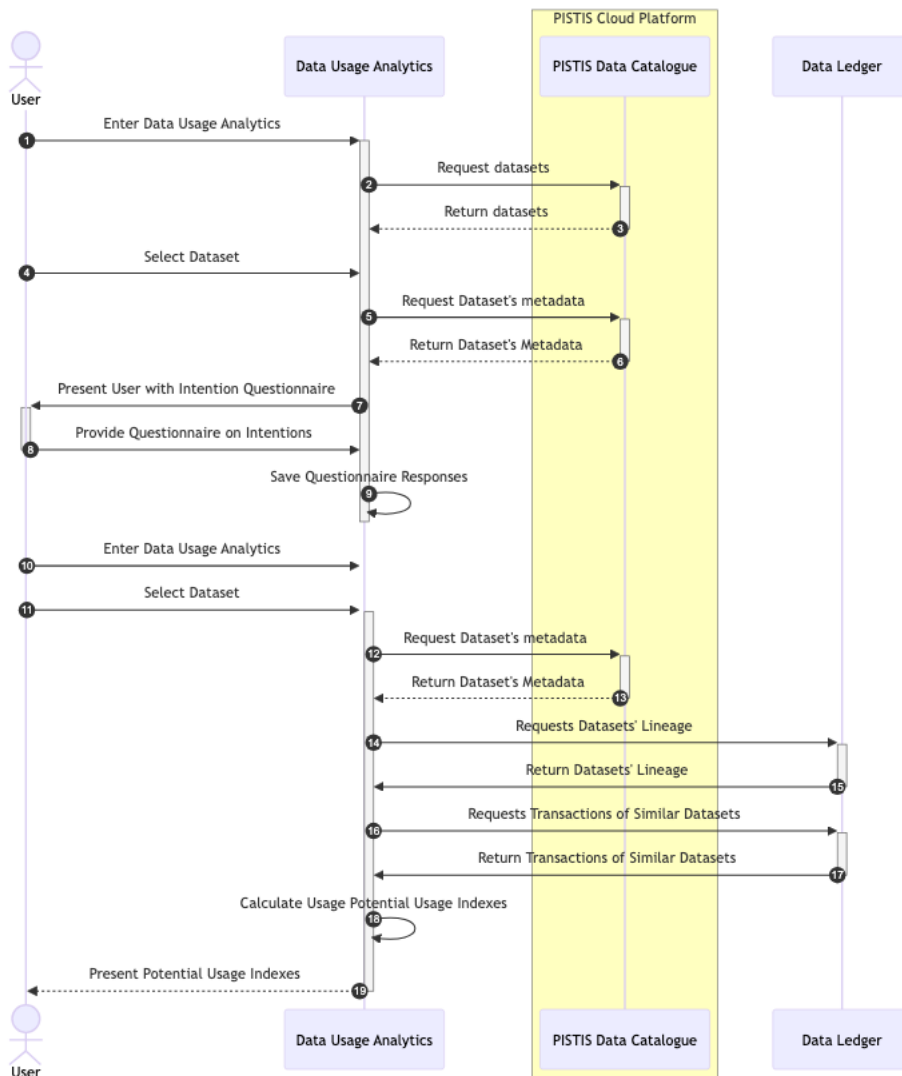


Figure 32: Data Usage and Intentions Analytics- Sequence Diagram

8 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.

8.1 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 is capable of taking 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 final 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 is able to 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.

8.1.1 RELATION TO OTHER COMPONENTS

8.1.1.1 Components providing Input to Anonymizer

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

8.1.1.2 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

8.2 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.

8.2.1 RELATION TO OTHER COMPONENTS

8.2.1.1 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

8.2.1.2 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

8.2.2 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 deletion 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.	∨

8.2.3 SEQUENCE DIAGRAMS

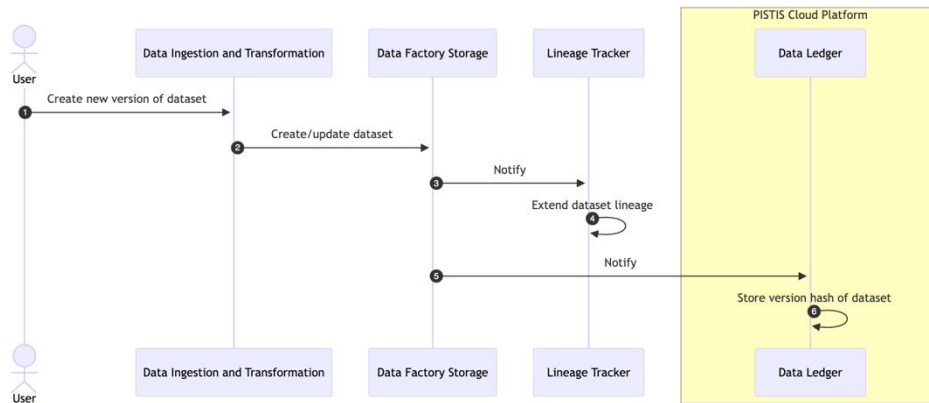


Figure 34: Lineage Tracker Sequence Diagram #1

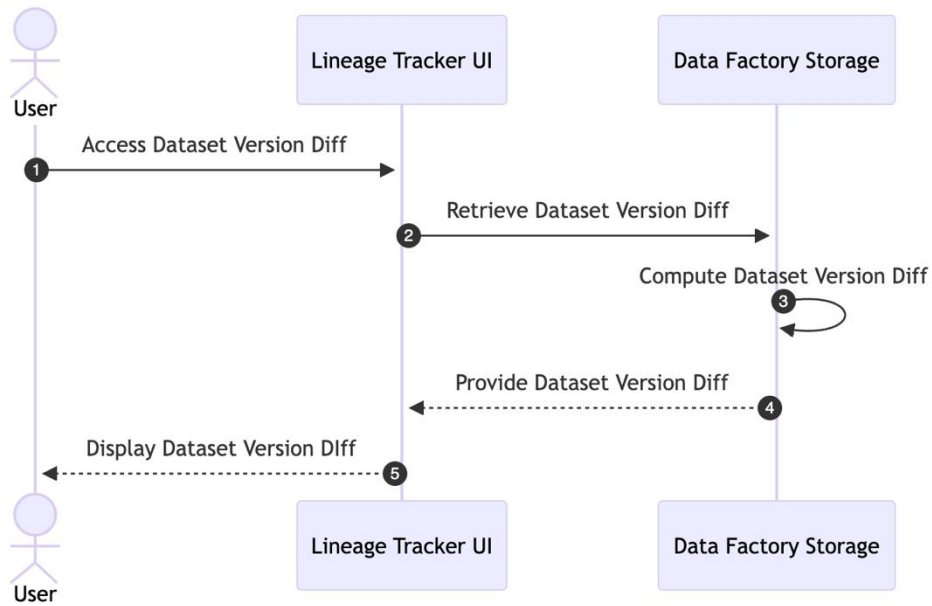


Figure 35: Lineage Tracker Sequence Diagram #2

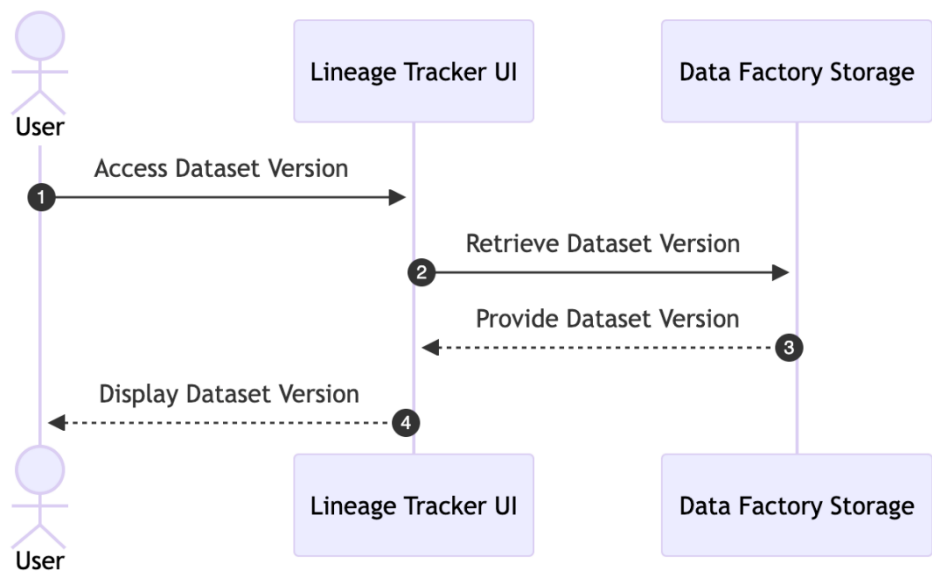


Figure 36: Lineage Tracker Sequence Diagram #3

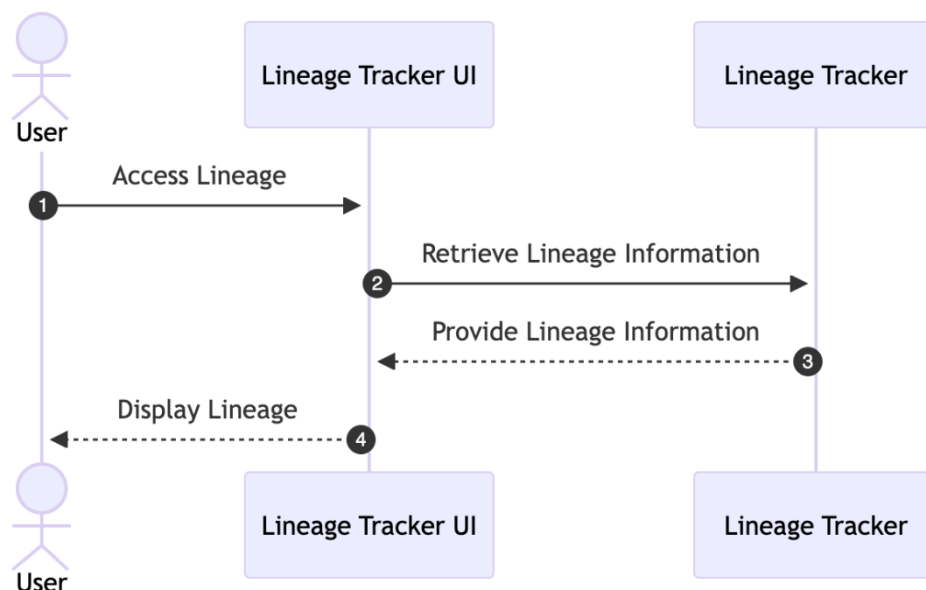


Figure 37: Lineage Tracker Sequence Diagram #4

8.3 GDPR CHECKER

GDPR checker component is responsible for ensuring compliance with the EU General Data Protection Regulation (GDPR). In essence, the GDPR checker not only verifies compliance regarding anonymity but also assesses adherence to the user's privacy profile, which includes factors such as anonymity, linkability, observability, and more. More specifically, such a privacy profile will be checked against the data that is going to be uploaded to the PISTIS platform and can essentially be exchanged through the platform to be compliant. The necessary level of privacy is relevant to the sensitivity and the nature of the data as well as the application requirements that produces these data. These privacy profiles, based on the characteristics of the data, can capture the required level of privacy protection. These measures (privacy profile) are expressed in as GDPR-alignment policies. Policies need to be in the form of *"if type of data X and type of data Y is exposed then this can have an impact on the privacy property k (e.g., anonymity) of the user"*.

Considering this, the core functionality of GDPR checker is to generate a type of smart contract that can act as certification of GDPR compliance (including the user's consent) for further processing upon the data. The component will consider the labels of the data (from Data Quality Assessment component) and the privacy profiles (to be defined by Organisation's Legal entity) and will provide either a certification of compliance with GDPR or static mitigation suggestions to the PISTIS Platform UI. Figure 38 presents the high-level architecture of the GDPR Checker including the internal components of the tool. GDPR checker involves various components and steps to ensure data compliance with GDPR.

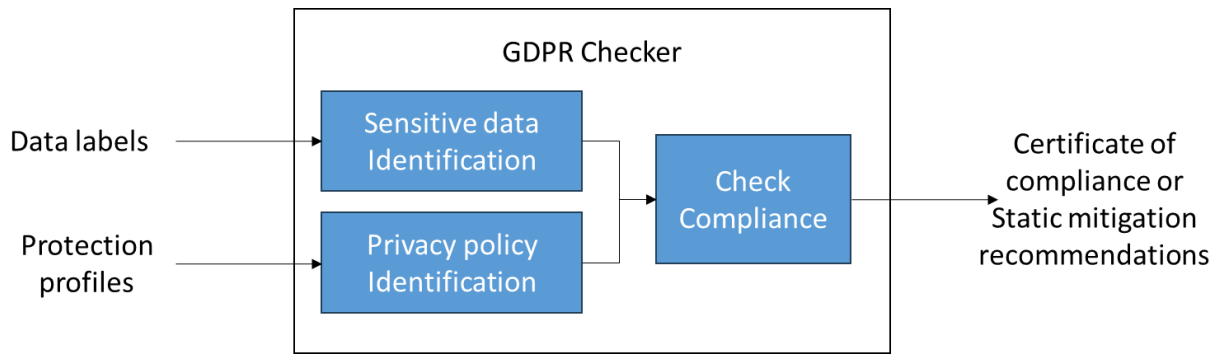


Figure 38: GDPR Checker high-level Architecture

8.3.1 RELATION TO OTHER COMPONENTS

8.3.1.1 Components providing Input to GDPR checker component

Input Required	Component providing the Input	Communication Method
Data labels and metadata	Data Quality Assessment	API
Privacy Policies	Organisation’s Legal entity	API
Data-related Smart Contract to be checked for GDPR compliance	Smart Contract Execution Engine	API

8.3.1.2 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
Checked GDPR violations	Smart Contract Execution Engine	API

8.3.2 API CALLS DESCRIPTIONS

GDPR Checker ^

GET	/getDataLabels Retrieve data labels	v
PUT	/updatePolicy Update an existing policy	v
GET	/getPolicy Retrieve a policy	v
DELETE	/deletePolicy Delete a policy	v
POST	/checkDataCompliance Check data compliance	v

8.3.3 SEQUENCE DIAGRAMS

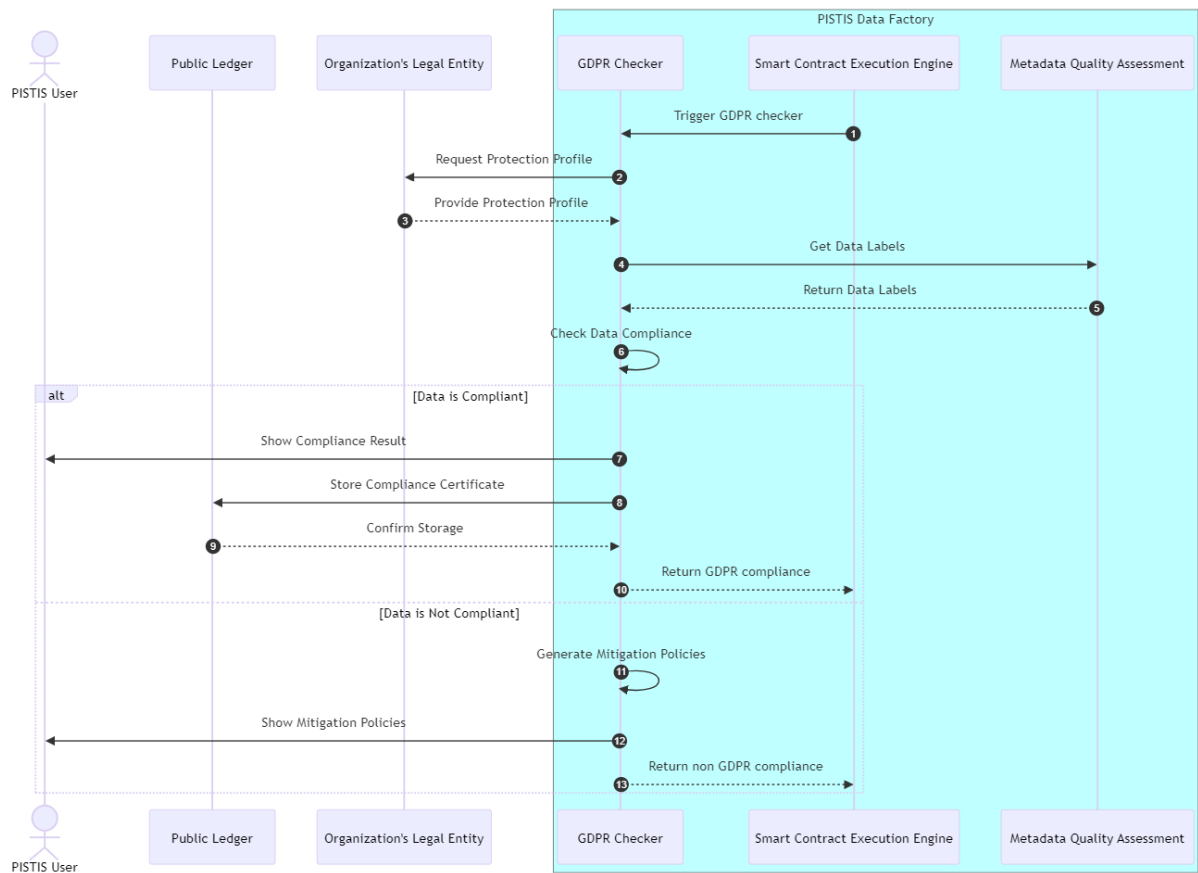


Figure 39: GDPR checker Sequence Diagram

8.4 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 in order 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 40 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.

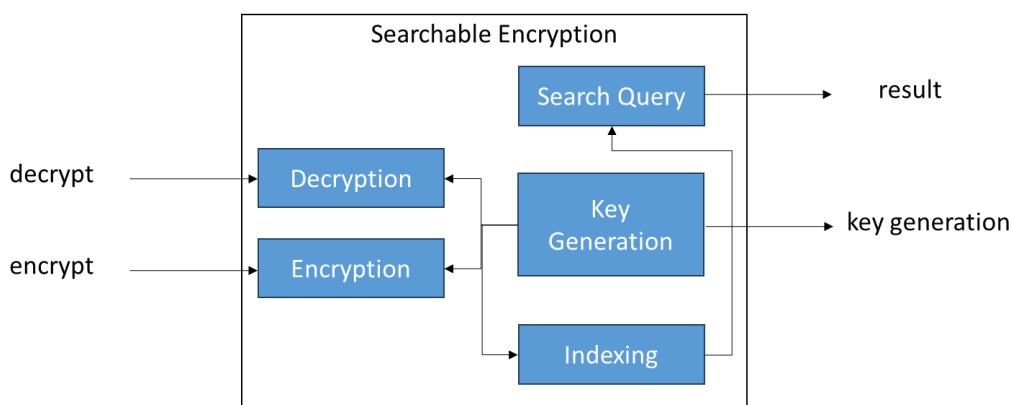


Figure 40: Searchable Encryption high-level Architecture

8.4.1 RELATION TO OTHER COMPONENTS

8.4.1.1 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

8.4.1.2 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

8.4.2 API CALLS DESCRIPTIONS

Searchable Encryption ^

- POST /initEncryption Encrypt v
- GET /search search data v

8.4.3 SEQUENCE DIAGRAMS

Figure 41 depicts the sequence of actions for the Searchable Encryption component.

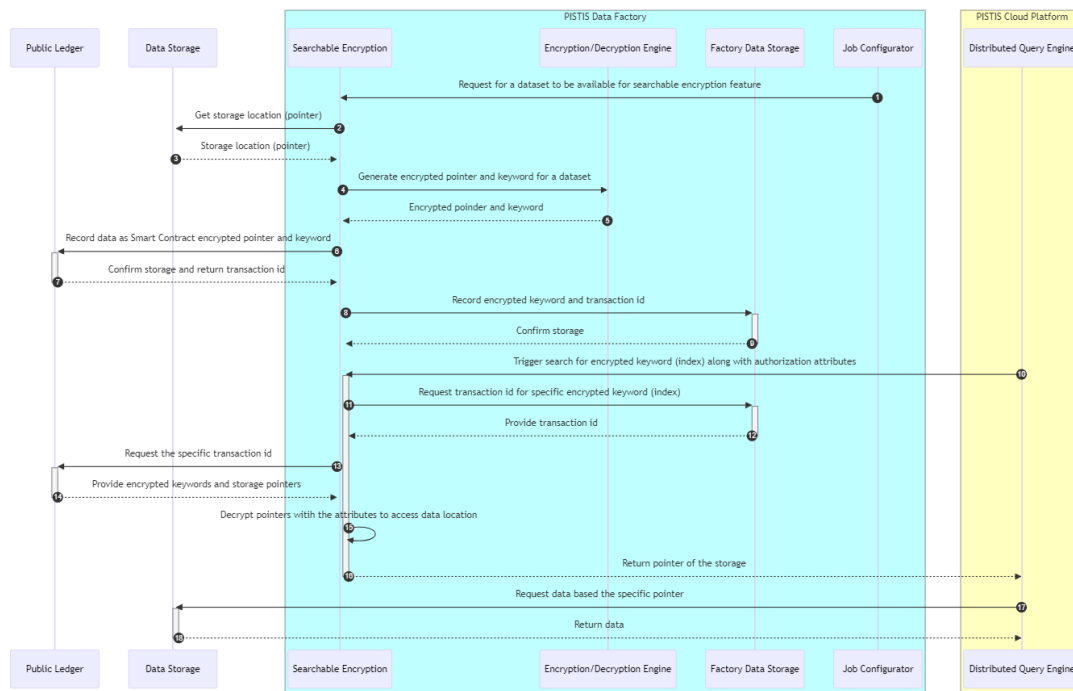


Figure 41: Searchable Encryption Sequence Diagram

8.5 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 42 represents the high-level architecture of the Encryption/Decryption component including the internal components of the tool.

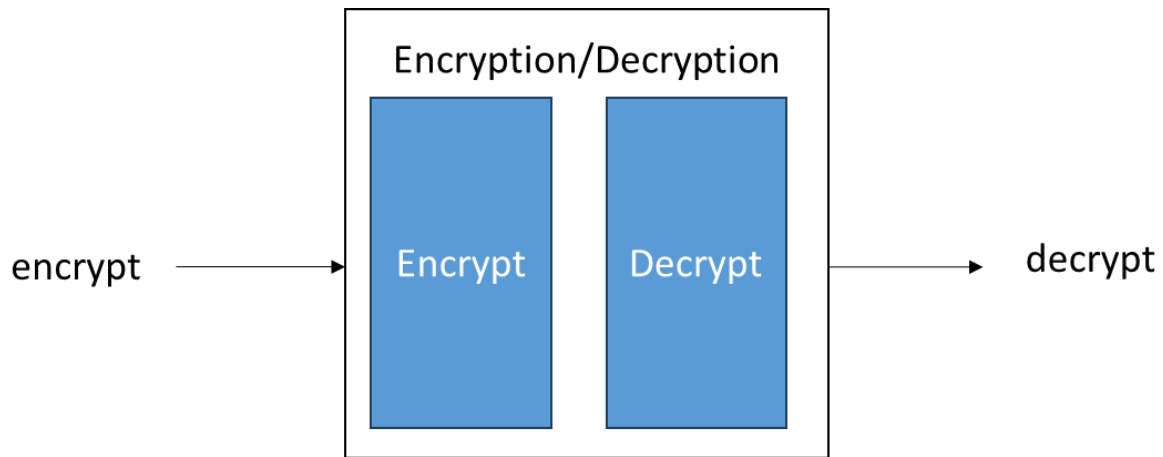


Figure 42 Encryption/Decryption high-level Architecture

8.5.1 RELATION TO OTHER COMPONENTS

8.5.1.1 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)
Encryption/Decryption functionality	Factory Data Storage	API

8.5.1.2 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)
Encryption/Decryption functionality	Factory Data Storage	API

8.5.2 API CALLS DESCRIPTIONS

Encryption/Decryption ^

POST /**encrypt** Encrypt Data v

POST /**decrypt** Decrypt Data v

8.5.3 SEQUENCE DIAGRAMS

The next figure depicts the sequence of actions for the Encryption/Decryption component. In a nutshell, a PISTIS Component triggers Encryption or Decryption functionality, where the Encryption/Decryption component will interact with the Data Factory User Wallet to get the appropriate credentials (e.g., the attributes stored in the form of verifiable credentials or any

necessary key or certificate). In the case of encryption, the component also releases attributes needed for decryption.

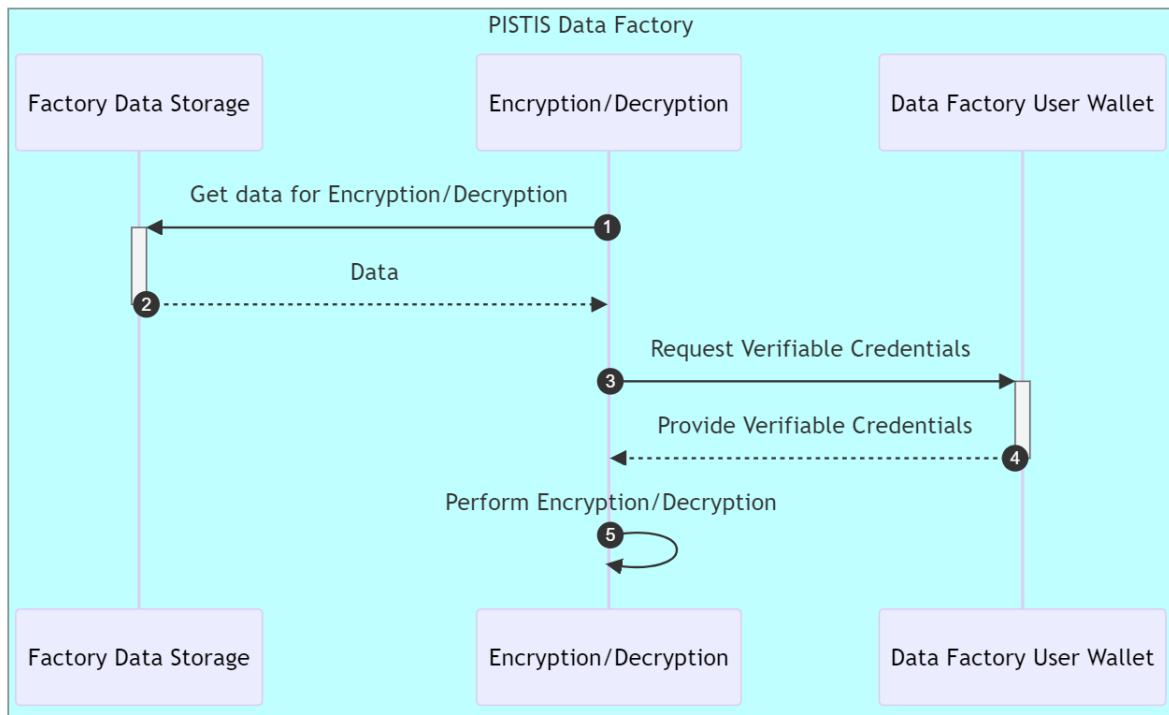


Figure 43: Encrypt/Decrypt Sequence Diagram

9 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.

9.1 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 gathered by the lineage tracking component in conjunction with Provenance tracking to fulfil its core functions. It facilitates the retrieval of usage tracking information, active contracts, and access rights. Additionally, it plays a crucial role in determining whether there is a violation of the usage contract, helping to ensure the adherence to contract agreements and maintain data integrity within the platform.

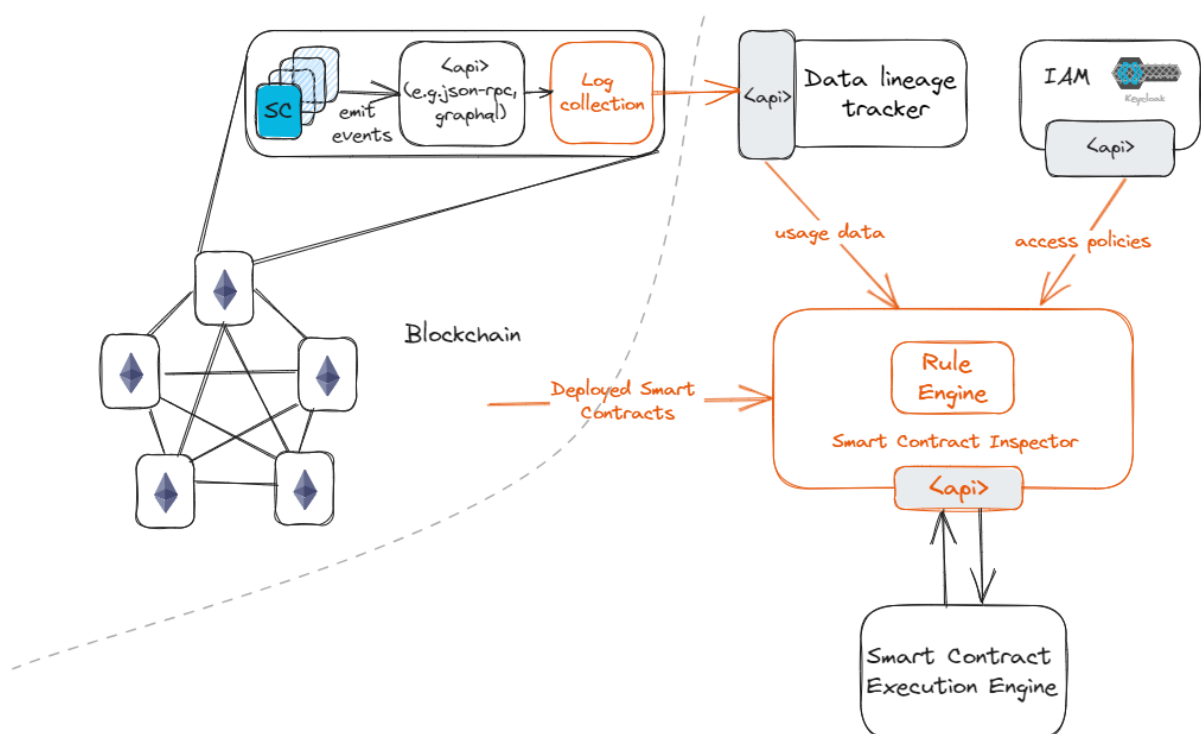


Figure 44: On chain platform inspection

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.

9.1.1 RELATION TO OTHER COMPONENTS

9.1.1.1 Components providing Input to On/Off Platform Contract Inspector

Input Required	Component providing the Input	Communication Method
User, resource, action to be performed	Access Policy Engine	API
Get usage tracking info, who did what and when for a specific resource	Lineage Tracker	API
Data format, data location	Data Check-In	API
Transaction Details	Data Blockchain	API

9.1.1.2 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

9.1.2 API CALLS DESCRIPTIONS

on on platform inspection ^

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

off off platform inspection ^

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

9.1.3 SEQUENCE DIAGRAMS

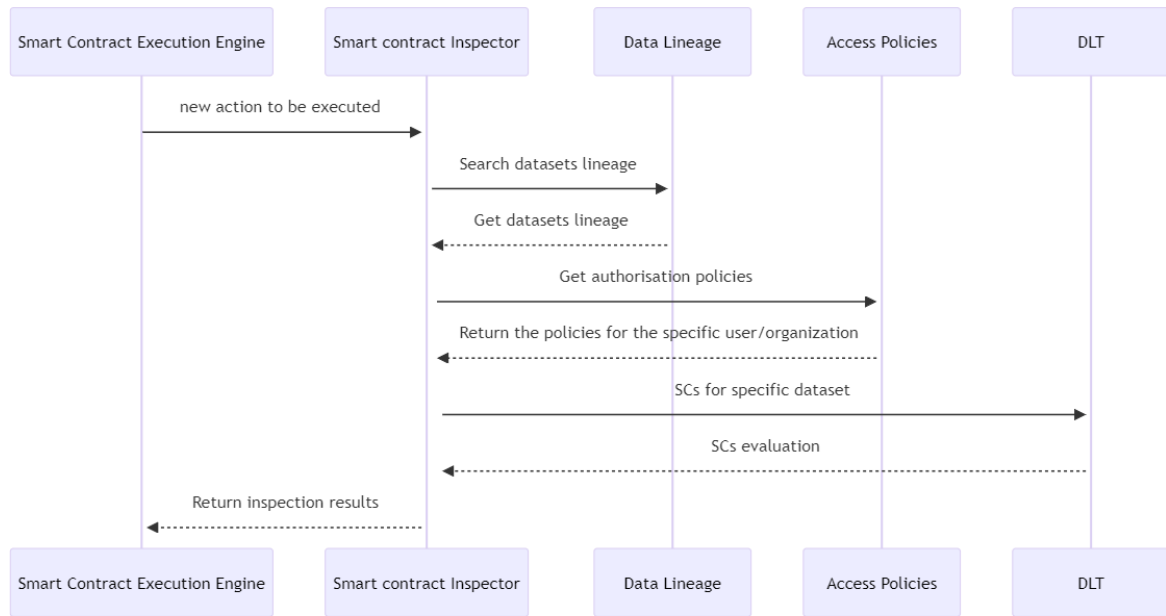


Figure 45: On platform contract inspection

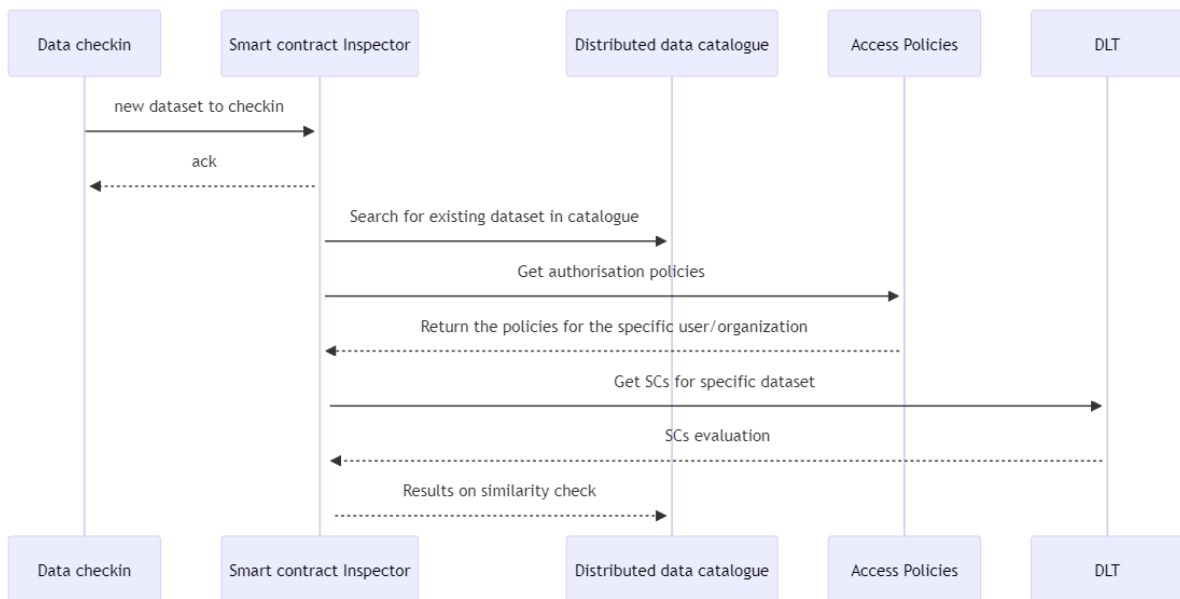


Figure 46: Off platform contract inspection

9.2 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.

9.2.1 RELATION TO OTHER COMPONENTS

9.2.1.1 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

9.2.1.2 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

9.2.2 API CALLS DESCRIPTIONS

Transactions Auditor 1.0 OAS 3.0

Component for the facilitation of Transactions Auditor

Servers

[http://localhost:7000/apl/ - Server](http://localhost:7000/apl/) Authorize

default ^

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

9.2.3 SEQUENCE DIAGRAMS

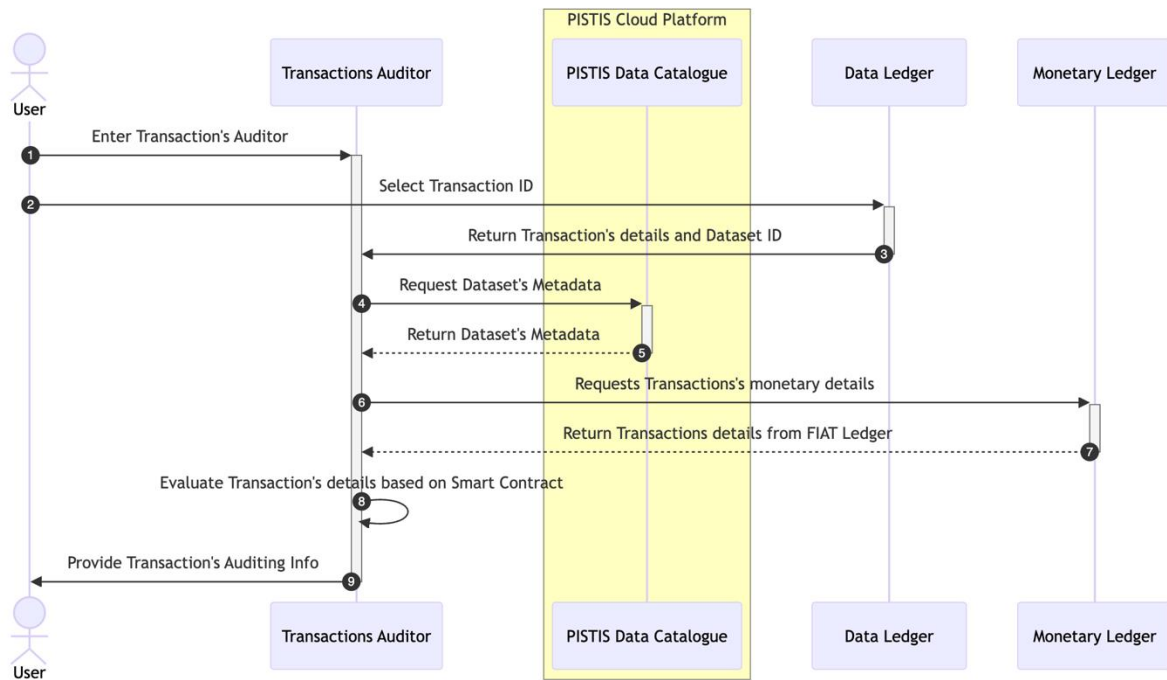


Figure 47: Transactions Auditor Sequence Diagram

9.3 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 is able to 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.

9.3.1 RELATION TO OTHER COMPONENTS

9.3.1.1 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

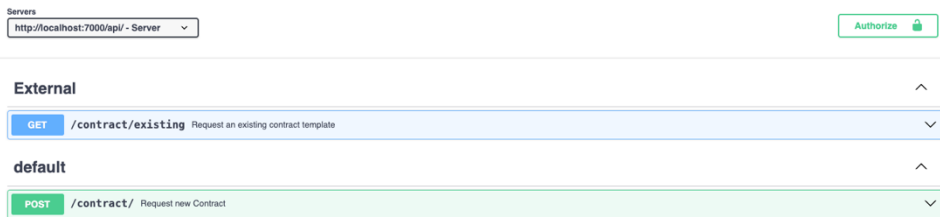
9.3.1.2 Components to which the Smart Contract Template Composer provides input

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

9.3.2 API CALLS DESCRIPTIONS

Smart Template Composer Gateway ^{1.0} ^{OAS 3.0}

Gateway for the facilitation of Smart Template Composer



9.3.3 SEQUENCE DIAGRAMS

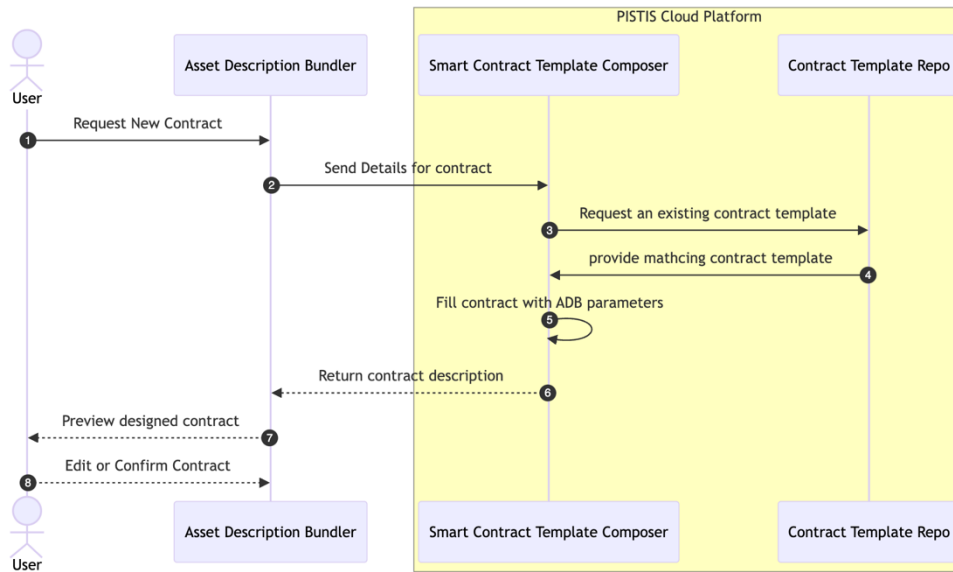


Figure 48: Smart Contract Template Composer - Sequence Diagram

9.4 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 the Transaction Auditor 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 49 presents the high-level architecture of the Smart Contract Execution Engine including the internal components of the tool.

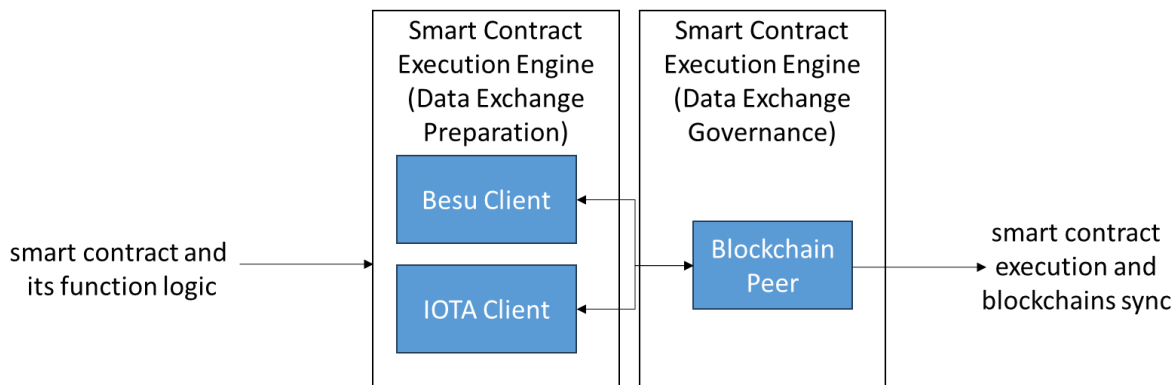


Figure 49: Smart Contract Execution Engine High-Level Architecture

9.4.1 RELATION TO OTHER COMPONENTS

9.4.1.1 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)
Deployed smart contract to be executed and its functional logic	PISTIS Data Ledger	Internal function (offered by PISTIS Data Ledger)
Check GDPR violations	GDPR Checker	API
Check violations, functional logic, and validity of the Smart Contract	Smart Contract Checker	API
PISTIS Cloud Platform		
Check authenticity (get VC attributes)	Digital DLT-Fiat Wallet	Internal function
Deployed smart contract to be executed	PISTIS Data Ledger	Internal function
Contract to be deployed	Asset Description Bundler	API
Inspection result prior to execution	On/Off Platform Inspector	API

9.4.1.2 Components to which Smart Contract Execution Engine component provides input

Output Served	Component ingesting the output	Communication Method
---------------	--------------------------------	----------------------

Organisation Boundaries		
Smart contract to be executed (including data synchronisation between the two ledgers). This will trigger all the necessary checks (e.g., GDPR violations and smart contract check) and then sequentially both ledgers (e.g., HyperLedger Besu and IOTA Tangle) to execute an already deployed smart contract.	Data Ledger - Monetary ledger	API
PISTIS Cloud Platform.		
Smart contract to be executed (including data synchronisation between the two ledgers). This will trigger all the necessary checks (e.g., on/off platform inspection) and then sequentially both ledgers (e.g., HyperLedger Besu and IOTA Tangle) to execute an already deployed smart contract.	Data Ledger - Monetary ledger	API

9.4.2 API CALLS DESCRIPTIONS

Smart Contract Execution Engine (Organization Boundaries) ^

POST /executeOrganizationTransaction Execute Data or Monetary Transaction v

Smart Contract Execution Engine (Pistis Cloud Platform) ^

POST /executePlatformTransaction Execute Data or Monetary Transaction v

9.4.3 SEQUENCE DIAGRAMS

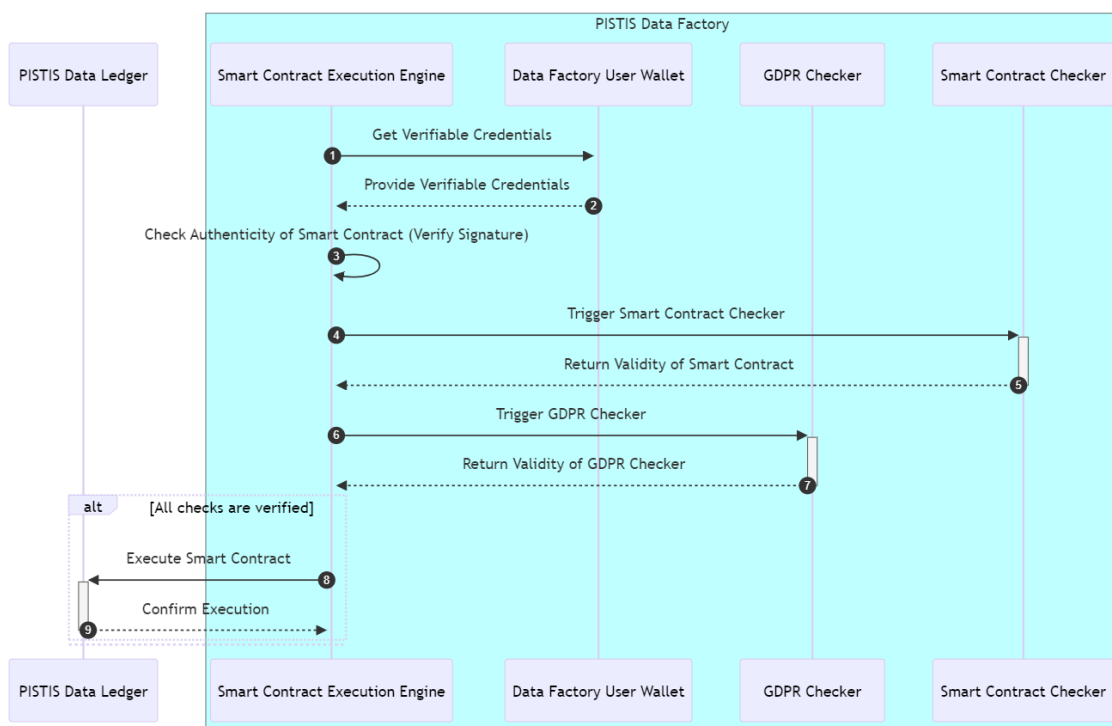


Figure 50: Smart Contract Execution Engine Sequence Diagram (Organisation Boundaries)

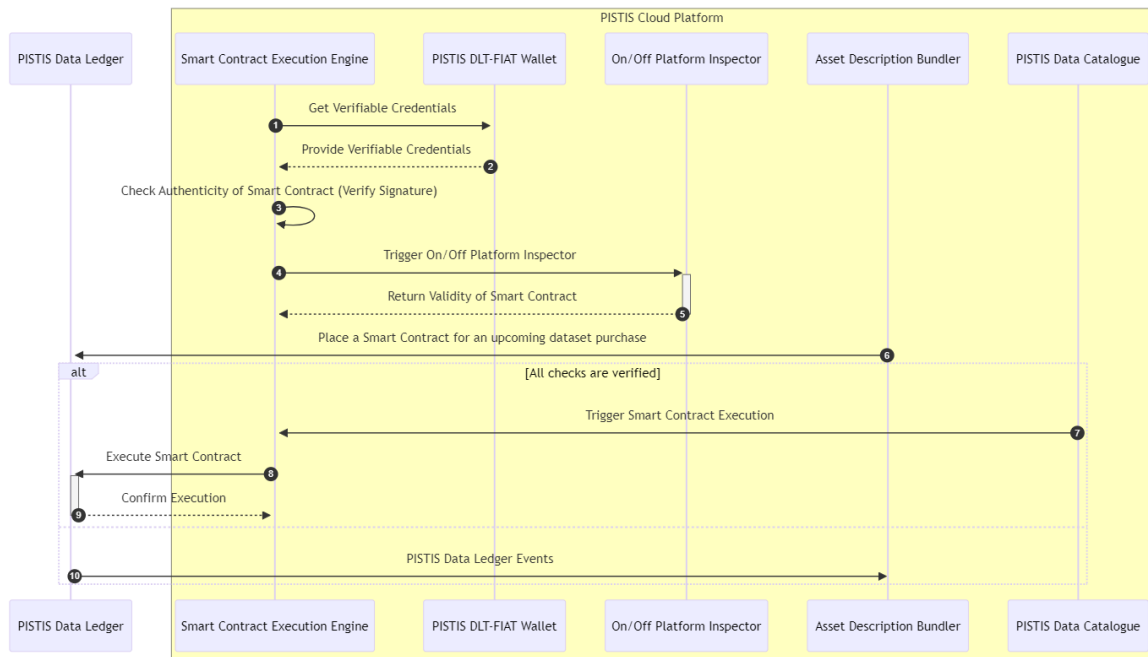


Figure 51: Smart Contract Execution Engine Sequence Diagram (PISTIS Cloud Platform)

9.5 DATA FACTORY USER WALLET

The Data Factory User Wallet hosts the identification credentials and plays a critical role within the PISTIS framework by securely storing verifiable credentials for self-sovereign identity (SSI) purposes. This component is a cornerstone in the architecture of the user identity management and authentication, and resides in the user side. Figure 52 presents the high-level architecture of the Identity Wallet including the internal components of the tool.

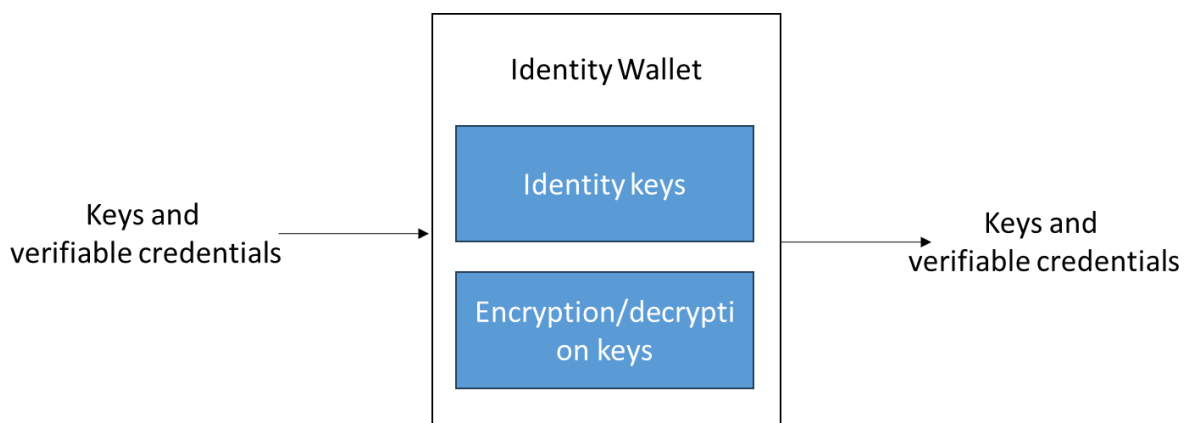


Figure 52: Data Factory User Wallet High-Level Architecture

9.5.1 RELATION TO OTHER COMPONENTS

9.5.1.1 Components providing Input to Data Factory User Wallet component

Input Required	Component providing the Input	Communication Method
Keys and Verifiable Credentials	PISTIS components	API
Monetary transactions information	Digital DLT-FIAT Wallet	API

9.5.1.2 Components to which Data Factory User Wallet component provides input

Output Served	Component ingesting the output	Communication Method
Keys and Verifiable Credentials	PISTIS components	API
Data transactions information	Digital DLT-FIAT Wallet	API

9.5.2 API CALLS DESCRIPTIONS

Data Factory User Wallet ^

GET	/getVC	Get Verifiable Credentials Attributes	v
POST	/storeVC	Get Transaction Data	v
POST	/createVC	Create Verifiable Credentials based on Attributes	v
POST	/syncMonetaryTransactionToken	Synchronization with Monetary Transaction (e.g., Token)	v

9.5.3 SEQUENCE DIAGRAMS

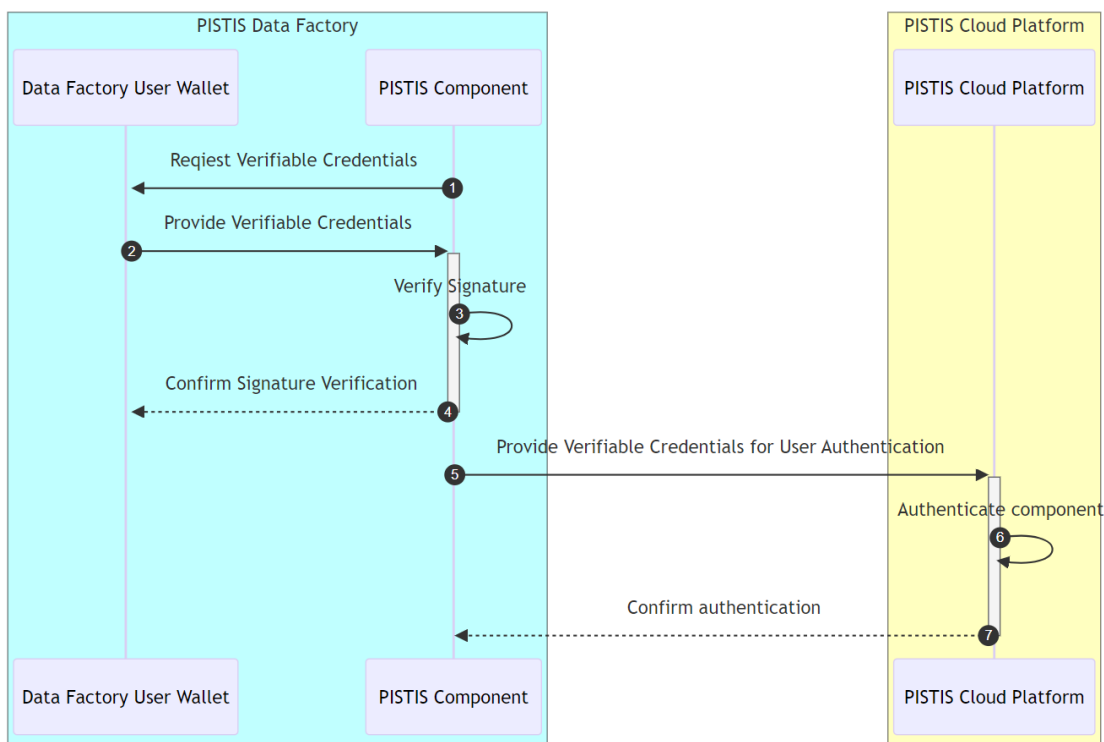


Figure 53: Data Factory User Wallet Sequence Diagram

10 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.

10.1 PISTIS DATA LEDGER

Distributed Ledger Technologies (DLT) have received growing attention in recent years as an innovative method of storing data. It is a fully decentralised record-keeping system used to record information. DLTs are rapidly gaining popularity, transforming the way organisations operate in the digital world. It is a system for storing and managing data and digital assets in a secure, transparent and decentralised manner. It ensures that data are tamper-proof and allows for a high level of transparency by making information available to everyone on the network, offering several advantages over traditional databases.

Thus, this technology holds paramount importance for storing smart contracts, lineage information, and data transactions within the framework of PISTIS. The adoption of Blockchain in PISTIS symbolizes a commitment to leveraging cutting-edge technology for secure, transparent, and efficient data management. It aligns with modern trends in digital transformation, where trust, security, and decentralisation are paramount. The integration of DLT not only enhances the PISTIS platform's operational efficiency but also fortifies trust among its users, making PISTIS a pioneering platform in the realm of secure and transparent data exchange and management. In addition, PISTIS Data Ledgers component includes a private channel for storing sensitive data such as pointers that need to be protected under access control policies defined by the user or the organisation and are attached to the specific data bundle.

10.1.1 RELATION TO OTHER COMPONENTS

10.1.1.1 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	API

10.1.1.2 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	API

10.1.2 API CALLS DESCRIPTIONS

PISTIS Data Ledger ^

POST /storeData Store Data v

POST /queryData Get Data v

10.1.3 SEQUENCE DIAGRAMS

Figure 54 depicts the sequence of actions for the PISTIS Data Ledger component.

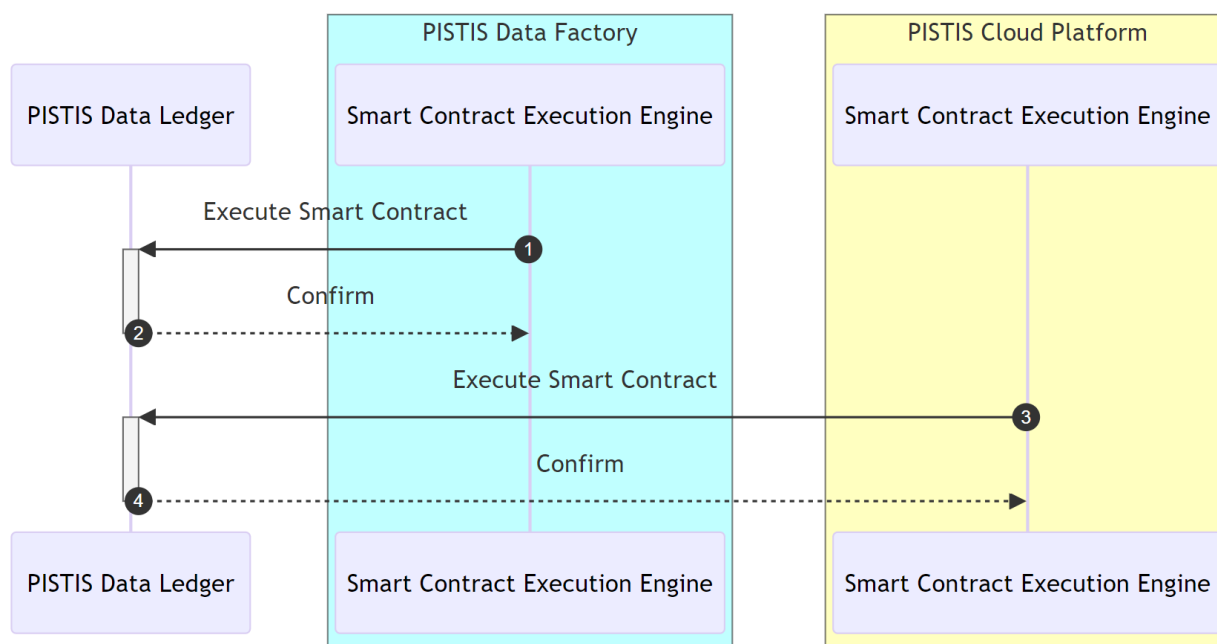


Figure 54: PISTIS Data Ledger Sequence Diagram

10.2 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 exactly 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.

10.2.1 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

10.2.1.1 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

10.2.2 API CALLS DESCRIPTIONS

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

10.2.3 SEQUENCE DIAGRAMS

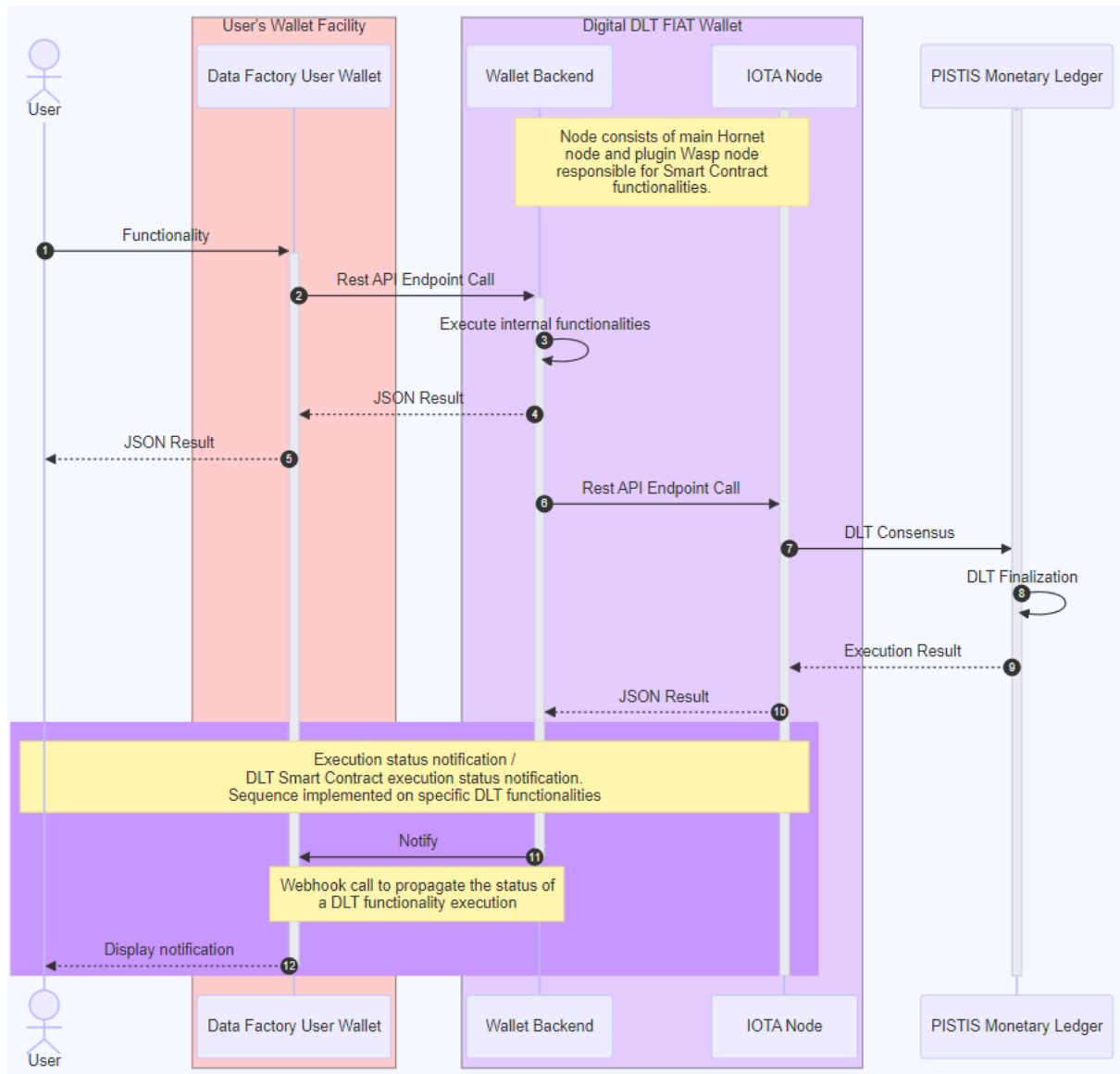


Figure 55: PISTIS Monetary Ledger Sequence Diagram

11 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.

11.1 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 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 organisation that is managing the overall PISTIS infrastructure that maintains the models and is able to upload new artefacts and fill-in new metadata information on the existing ones.

11.1.1 RELATION TO OTHER COMPONENTS

11.1.1.1 Components providing Input to the PISTIS Models Repository

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

11.1.1.2 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

11.1.2 API CALLS DESCRIPTIONS

PISTIS Models Repository ^{1.0} ^{OAS 3.0}

Component for the facilitation of models repo

Servers

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

default ^

- GET** /models/ Retrieve Available models
- GET** /models/metadata Retrieve Metadata of Available models
- GET** /single-model/{modelId} Retrieve model by Id
- GET** /single-model/{modelId}/metadata Retrieve model metadata by Id
- PUT** /model/{modelId} Update model
- DELETE** /model/{modelId} Delete model
- GET** /model/{modelId} Download model
- PUT** /model/{modelId}/metadata Update model metadata
- DELETE** /model/{modelId}/metadata Delete model metadata
- GET** /model/{modelId}/metadata Download model metadata
- POST** /model/ Add model
- POST** /model/upload Upload new model
- POST** /model/upload/metadata Upload new model metadata

S

11.1.3 SEQUENCE DIAGRAMS

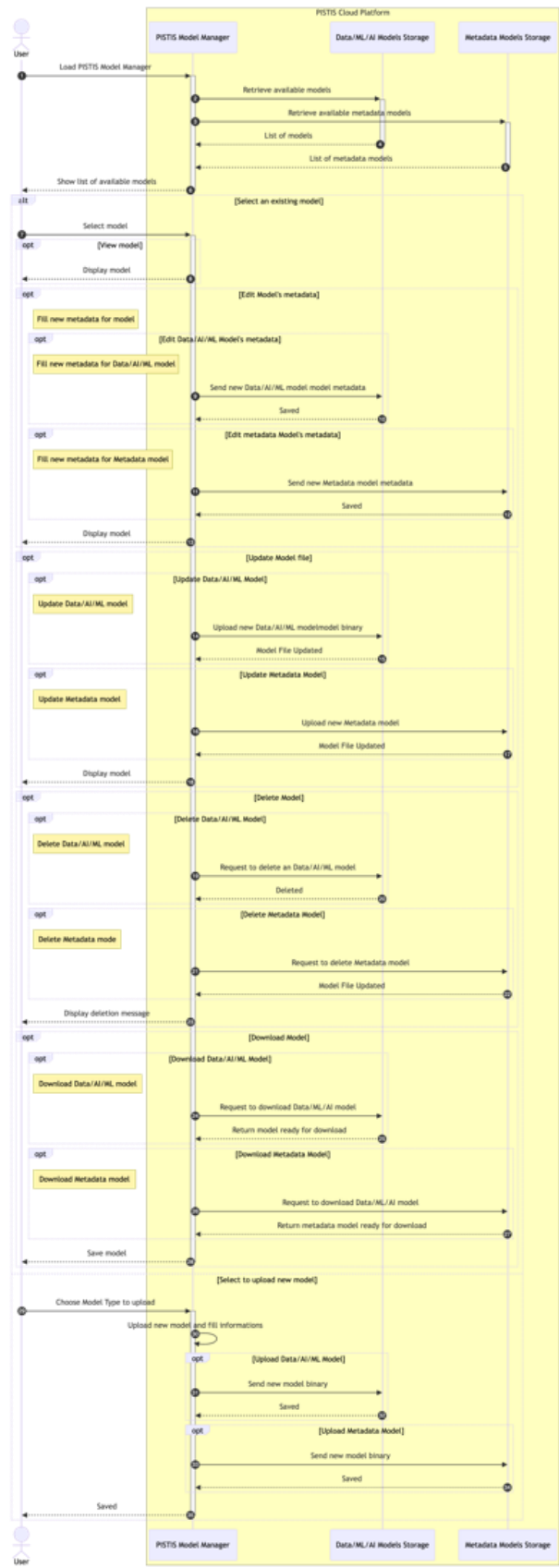


Figure 56: PISTIS Models Repository - Sequence Diagram

11.2 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.

11.2.1 RELATION TO OTHER COMPONENTS

11.2.1.1 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

11.2.1.2 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

11.2.2 API CALLS DESCRIPTIONS

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

11.2.3 SEQUENCE DIAGRAMS

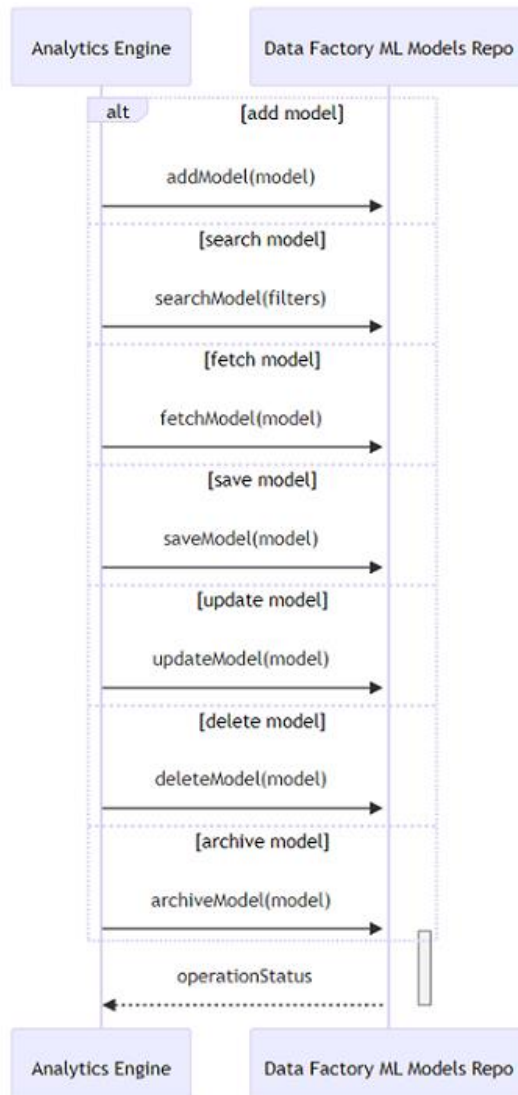


Figure 57: Factory ML Models Repo Sequence Diagram

11.3 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.

11.3.1 RELATION TO OTHER COMPONENTS

11.3.1.1 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

11.3.1.2 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

11.3.2 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

11.3.3 SEQUENCE DIAGRAMS

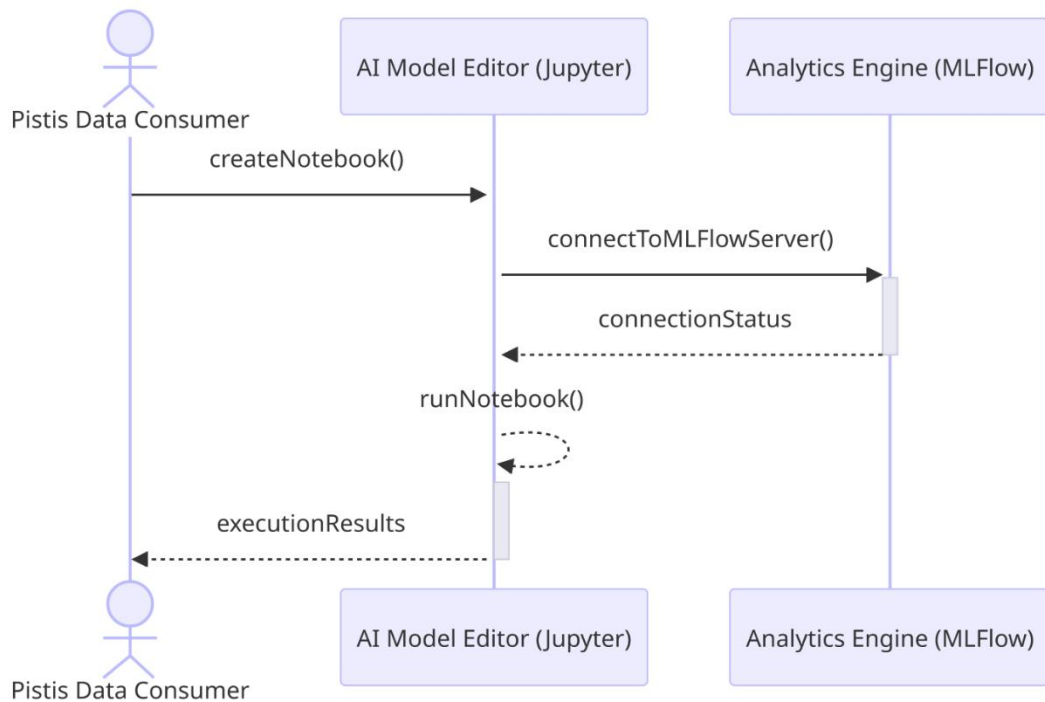


Figure 58: AI Model Editor Sequence Diagram

12 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.

12.1 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 OpenIDC⁶ 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 eIDAS⁷ 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.

12.1.1 RELATION TO OTHER COMPONENTS

12.1.1.1 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

12.1.1.2 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

⁶ <https://github.com/OpenIDC>

⁷ <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

12.1.2 API CALLS DESCRIPTIONS

Authentication ^

POST /token Get an access token 🔒 ▼

Authorization ^

POST /authorize Authorization 🔒 ▼

GET /authorize/{username}/{audience}/{scope} Check authorization for a given user, audience and scope 🔒 ▼

Resource ^

POST /resource Registers a new Data Asset in IAM 🔒 ▼

GET /resource/{id} Read a Data Asset by ID 🔒 ▼

PUT /resource/{id} Update a Data Asset's attributes by ID 🔒 ▼

DELETE /resource/{id} Delete a Data Asset from IAM by ID 🔒 ▼

GET /resource/{id}/policies Retrieves access policies for a Data Asset 🔒 ▼

Factory ^

POST /factory Registers a new Data Factory in IAM 🔒 ▼

12.1.3 SEQUENCE DIAGRAMS

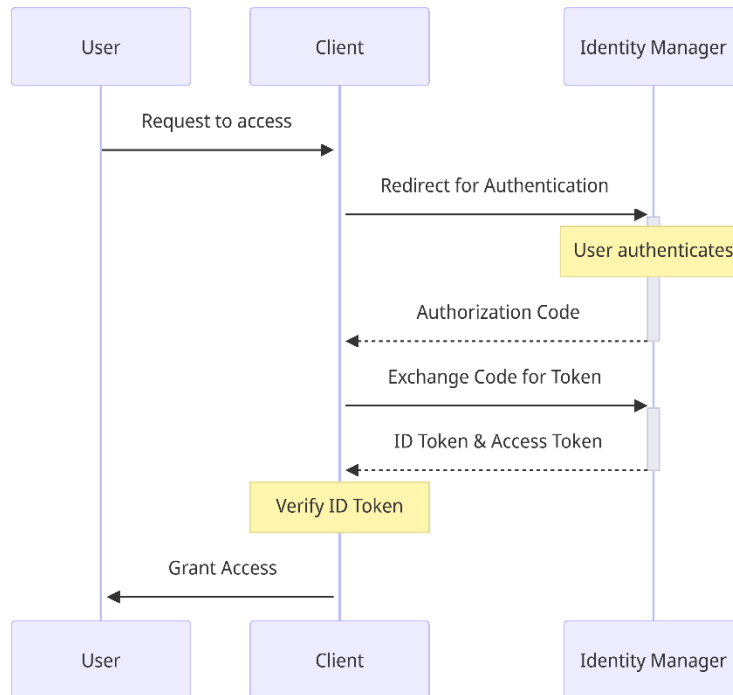


Figure 59: Identity Manager Authentication Sequence Diagram

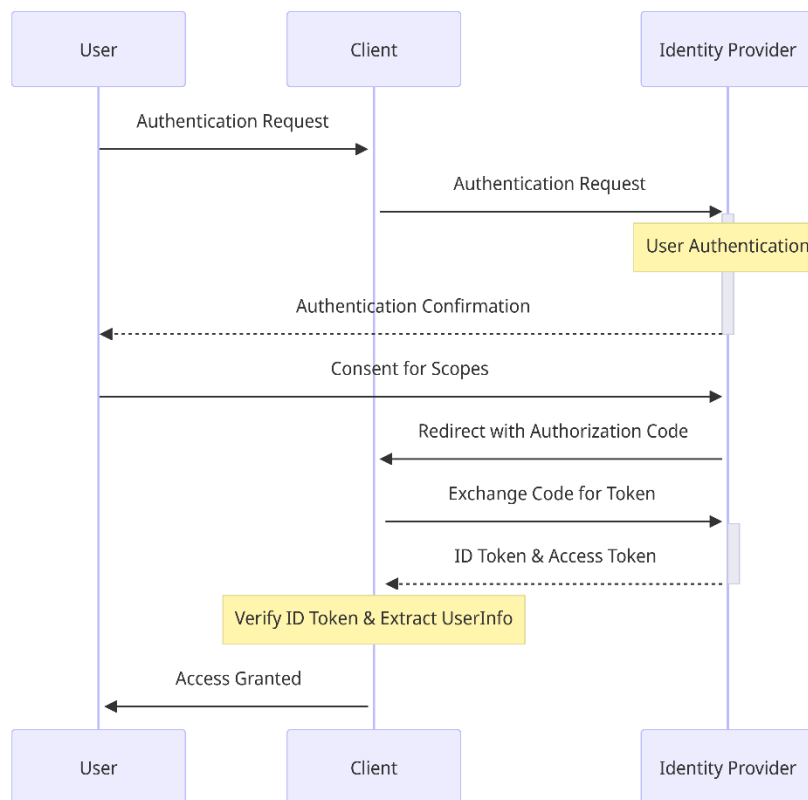


Figure 60: Identity Manager Authorisation Sequence Diagram

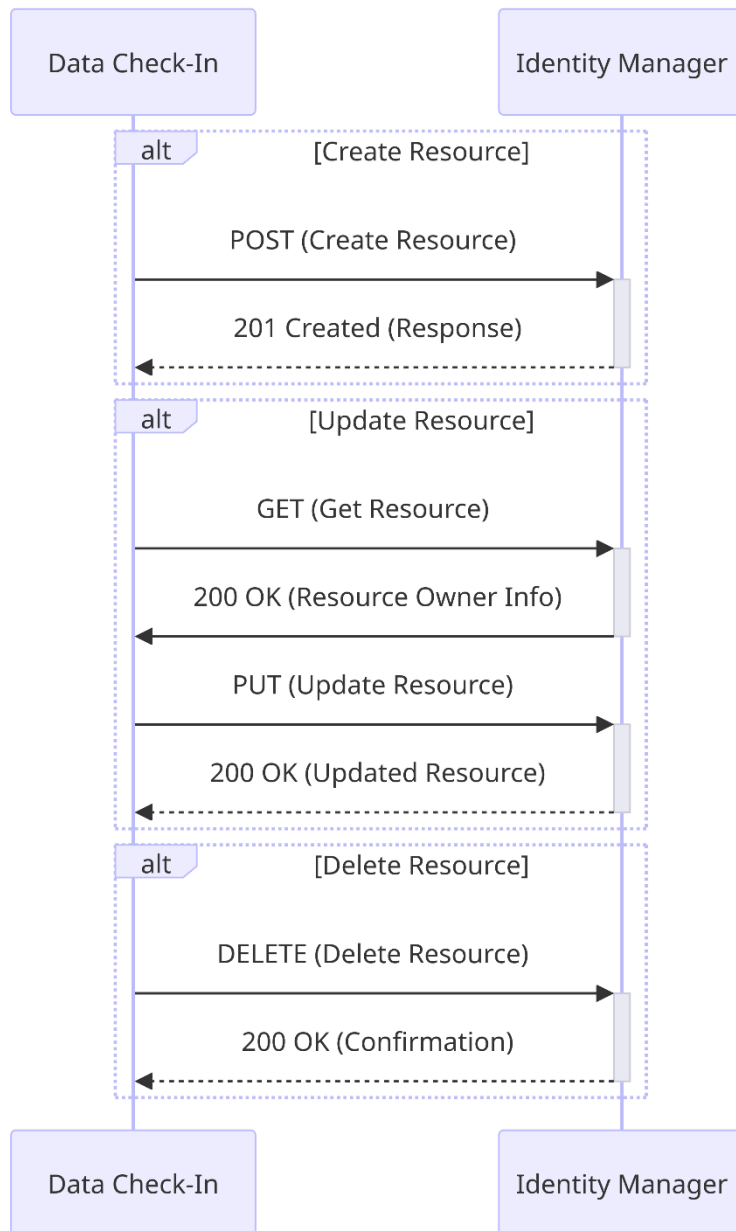


Figure 61: Identity Manager Resource management Sequence Diagram

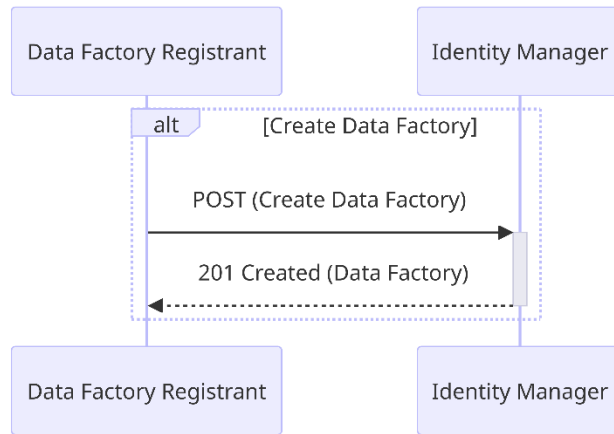


Figure 62: Identity Manager Data Factory registration Sequence Diagram

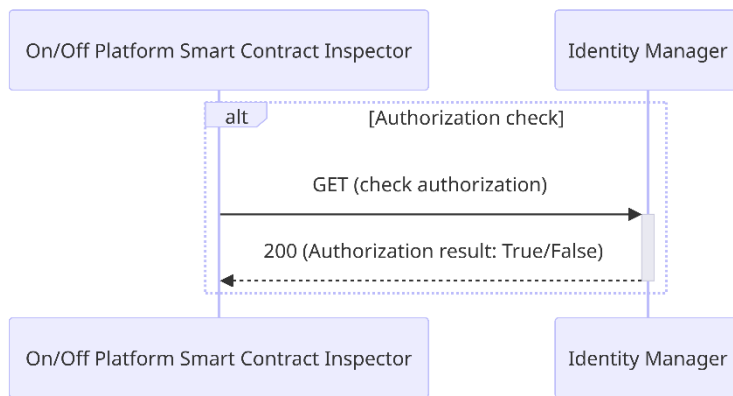


Figure 63: Identity Manager Authorisation check for user, audience, and scope Sequence Diagram

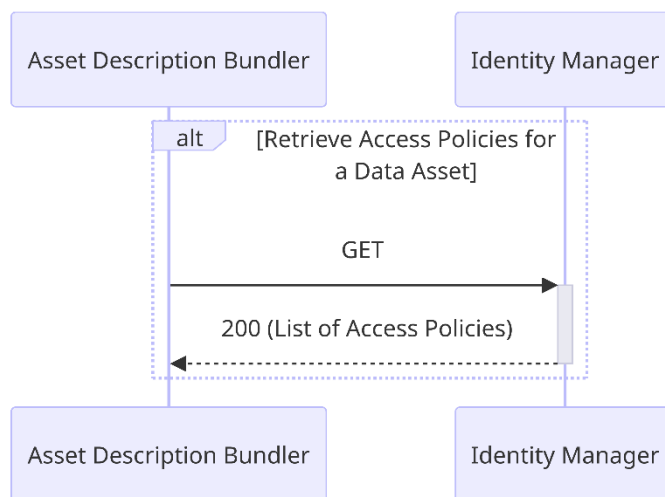


Figure 64: Identity Manager Access Policies retrieval for a Data Asset Sequence Diagram

13 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.

13.1 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.

13.1.1 RELATION TO OTHER COMPONENTS

13.1.1.1 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

13.1.1.2 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

13.1.2 API CALLS DESCRIPTIONS

Data Factories Registrant 1.0 OAS 3.0

Component for the facilitation of registration new factories

Servers: Authorize

User ^

- POST /register-new-factory/ Register new factory v
- GET /retrieve-ip/{factoryId} Retrieve the ip of the other factory v

Admin ^

- PUT /notifyAdmin/{factoryId} Accept/Deny new factory v
- GET /retrieve-status/{factoryId} Retrieve the status of factory v

13.1.3 SEQUENCE DIAGRAMS

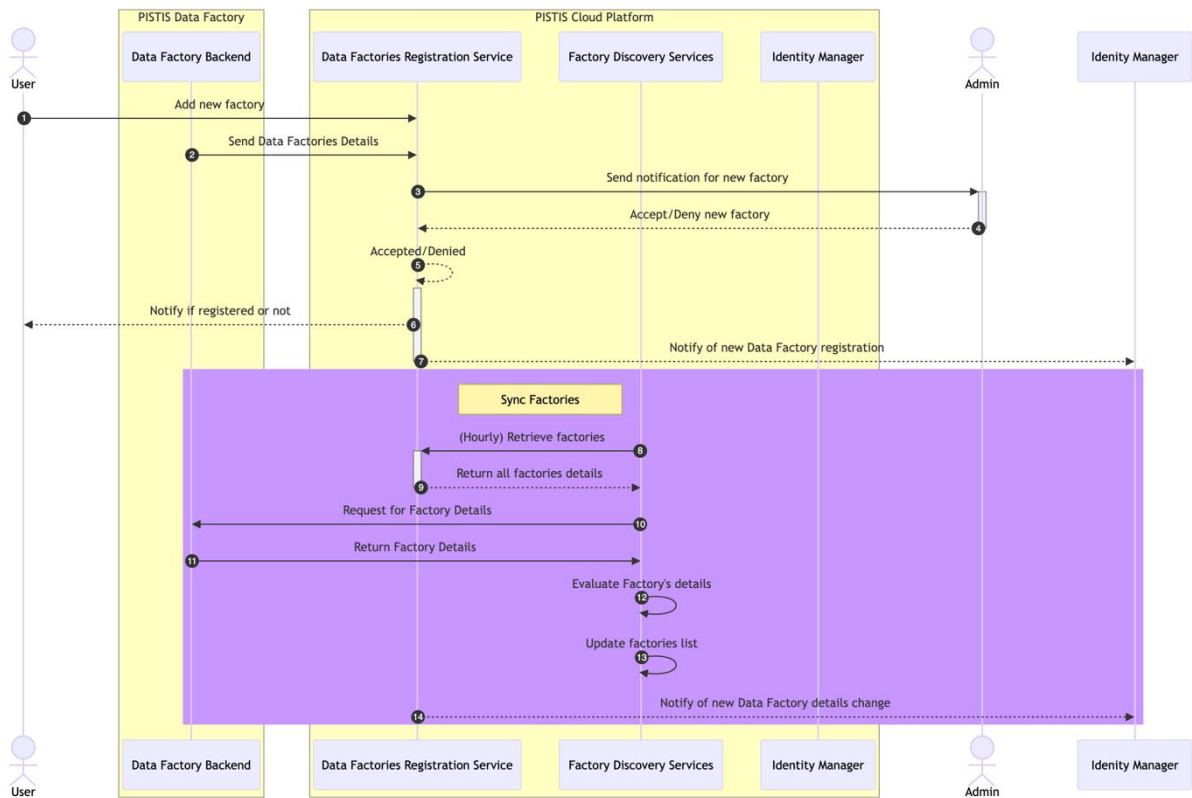


Figure 65: Data Factories Registrant Sequence Diagram

13.2 SYSTEM AND ACTIVITIES MONITOR

The 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.

13.2.1 RELATION TO OTHER COMPONENT

13.2.1.1 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

13.2.1.2 Components to which the System and Activities Monitor provides input

Output Served	Component ingesting the output	Communication Method
N/A		

13.2.2 API CALLS DESCRIPTIONS

N/A

13.2.3 SEQUENCE DIAGRAMS

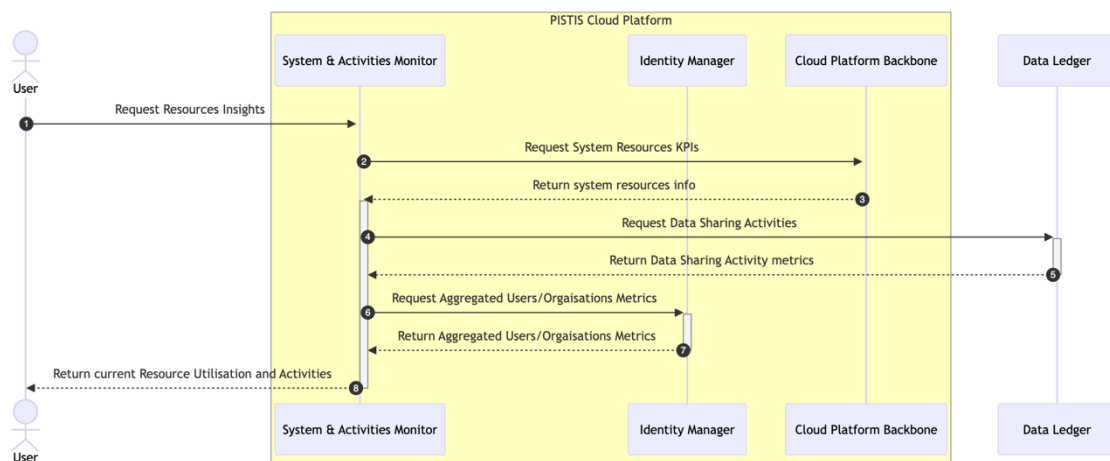


Figure 66: Resources and Activities Monitor- Sequence Diagram

14 CONCLUSIONS

The main objective of the deliverable is to provide the first version of the PISTIS platform's architecture. This includes describing the main interactions between the components that form the different bundles, documenting their APIs, and showcasing the sequence diagrams of the components.

In this, first version of the architecture the focus is on the functionalities of the components required to implement the previously defined usage scenarios and the PISTIS MVP. The overall platform is defined as a combination of the PISTIS Data Factory deployments that is operated by the Data Providers, and the PISTIS Cloud Platform that is used to govern the monetary transactions and provide the monetisation services for the overall ecosystem.

The current architecture will be used as a reference for the technical work in the project to be delivered as components. Their detailed descriptions and code implementation will be done under WP2 and WP3. Then, they will be integrated into the overall PISTIS platform by WP4, following the agile development methodology for software delivery. Accordingly, the architecture presented in this document is expected to evolve, and there is a possibility of change in specific components. Thus, updated versions of the current architecture are expected to be released alongside the launch of the different prototype versions of the platform, and all these changes will be documented in the upcoming WP4 deliverables.