TRUSTS

Trusted Secure Data Sharing Space

D3.11 Platform Status Report III

Authors: Ahmad Hemid, Kim Fidomski

December 2022 (M36)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871481

TRUSTS Trusted Secure Data Sharing Space

D3.11 Platform Status Report III

Document Summary Information

Grant Agreement No	871481	**Acronym**	TRUSTS
Full Title	TRUSTS Trusted Secure Data Sharing Space		
Start Date	01/01/2020	Duration	36 months
Project URL	https://trusts-data.eu/		
Deliverable	D3.11 Platform Status Report III		
Work Package	WP3 - TRUSTS Platform implementation		
Contractual due date	31/12/2022	**Actual submission date**	13/02/2023
Nature	Report	**Dissemination Level**	Public
Lead Beneficiary	Fraunhofer (FhG)		
Responsible Author	Ahmad Hemid (FhG), Kim Fidomski (FhG)		
Contributions from	REL, KNOW, RSA, EMC, SWC, LST, FORTH, FhG		

Revision history (including peer reviewing & quality control)

Version	Issue Date	% Complete	Changes	Contributor(s)
v0.1	25/10/2022	5	Initial Deliverable Structure	Ahmad Hemid (FhG), Kim Fidomski (FhG)
v0.2	20/11/2022	10	First version of Chapter 2	Kim Fidomski (FhG)
v0.3	14/12/2022	25	First version of Chapter 3	Nikos Fourlataras (REL), Stefan Gindl (RSA), Dominik Kowald (KNOW), Alan Barnett (EMC), Victor Mireles (SWC)
v0.4	15/12/2022	65	First version of Chapter 1, Chapter 4, Chapter 5, and Chapter 6	Nikos Fourlataras (REL), Stefan Gindl (RSA), Dieter Theiler (KNOW), Victor Mireles (SWC) Evangelos Kotsifakos (LST), Steffen Biehs (FhG), Ioannis Chatzakis (Forth) Ahmad Hemid (FhG), Kim Fidomski (FhG)
v0.5	15/12/2022	65	Pre-review of Chapter 3	Ahmad Hemid (FhG), Kim Fidomski (FhG)
v1.0	19/12/2022	70	Deliverable ready for first peer-review	Ahmad Hemid (FhG), Kim Fidomski (FhG)
v1.1	20/12/2022	70	First peer-review	Gianna Avgousti (eBOS)
v1.2	22/12/2022	70	Revision according to peer-review	Ahmad Hemid (FhG), Kim Fidomski (FhG)
v1.3	23/12/2022	90	Finalization all chapters' content	Ahmad Hemid (FhG), Kim Fidomski (FhG)
v2.0	08/02/2023	92	Final peer-review	Gianna Avgousti (eBOS), Evangelos Kotsifakos (LST)
v2.5	08/02/2023	99	Refinement based on the final peer-review	Ahmad Hemid (FhG), Kim Fidomski (FhG)
v3.0	10/02/2023	100	Final version	Ahmad Hemid (FhG), Kim Fidomski (FhG)

Disclaimer

Copyright message

Table of Conents

List of Figures

List of Tables

Glossary of terms and abbreviations

Abbreviation / Term	Description
ACME	Automated Certificate Management Environment
API	Application Programming Interface
CKAN	Comprehensive Knowledge Archive Network
DAPS	Dynamic Attribute Provisioning System
DevOps	Development and Operations
DMA	Data Market Austria
DSC	IDS Dataspace Connector
EOSC	European Open Science Cloud
ETL	Extract, Transform, Load
FR	Functional Requirement
GA	Grant Agreement
HLF	Hyperledger Fabric
IDS	International Data Spaces
IDSA	International Data Spaces Association
IM	Information Model
ML	Machine Learning
MVP	Minimum Viable Product
NFR	New Functional Requirement
RDF	Resource Description Framework
SDK	Software Development Kit
SMTP	Simple Mail Transfer Protocol
TLS	Transport Layer Security
UC	Use Case
UI	User Interface
UUID	Universally Unique Identifier
WP	Work Package

Executive Summary

This is the third and final status report of the TRUSTS platform. It addresses the technical progress achieved in the last project year (M25-M36) and the status of the TRUSTS platform at the end of the project life cycle in terms of functionality and operational parameters.

D3.11 is based on the previous versions of this deliverable and the progress made. The development of the Minimum Viable Product (MVP) Version 3 and the successor versions was done in accordance with the functional requirement (FR) specification as described in D2.3 and through regular exchange of input with technical and non-technical experts. Accordingly, the latest version of the FRs specification is listed in this report.

MVP.v2, described in more detail in D3.10, was handed over to WP5 to perform the use case (UC) trials. So were the subsequent MVP versions, which are described in more detail in this report. Based on the UC trials conducted for each MVP, WP5 provided regular feedback.

Artefacts and components reused from known projects and initiatives, namely Data Market Austria (DMA) [1], International Data Spaces (IDS) [2], and Comprehensive Knowledge Archive Network (CKAN) are summarised in this report. Artefacts and components developed as part of the TRUSTS project that are included in the platform are also introduced in this report. General information about the component's function as well as technical information like dependencies, interface descriptions, and FRs covered are given. Some of these components were integrated into an MVP version for the first time in the third project year. Corresponding information can also be found for each component.

These components were the basis for the platform architecture. Consequently, adding components led to changes in the platform architecture of the various MVPs. For the platform versions released this year, the corresponding architecture is presented in this report. Furthermore, the functional and technical differences between the MVPs are highlighted.

The end of project version of the TRUSTS platform, named TRUSTS v1.0, covers all aspects and features identified as important for a data marketplace, such as buying and selling data assets, secure communication, recommendation, and a friendly user interface (UI).

This report serves as a snapshot of the platform at various points in the project's final year. The reuse of components as well as building on experiences and best practices from other initiatives and projects, is a first step towards the formation of standards in the field of secured and federated data markets.

1 Introduction

Since the start of the project, work package (WP) 3 has been working, in collaboration with other WPs and in regular exchange of input with technical and non-technical experts to develop the TRUSTS platform that meets the functional and technical requirements defined in the project. This report shares the relevant findings and experiences from that work.

Over the three years of the project, expanded platform versions were continuously released, a fact that is reflected in the total of five MVP versions created during the project. In the first platform status report, D3.9, re-submitted M23, MVP.v0, and MVP.v1 were described. The second platform status report, D3.10, submitted in M24, described MVP.v2. This report, the last platform status report in a series of three, introduces MVP.v3 and MVP.v4 as well as the end of project version of the TRUSTS platform.

Looking at the different platform versions, it becomes clear that the range of functions was increased. An ever-increasing number of requirements and components were considered in each of the MVP versions. MVP.v4, which is described in more detail in this report, integrates for the first time all the components developed during the project or reused from other projects and initiatives. These components formed the building blocks for the MVP architectures.

This report takes the reader through a project year in which major developments were made to the TRUSTS platform. A platform emerged that meets the requirements that have been identified as important for a secured and federated data marketplace.

1.1 Mapping Projects' Outputs

The purpose of this section is to map TRUSTS Grant Agreement (GA) commitments, both within the formal Deliverable and Task description, against the project's respective outputs and work performed.

Table 1: Adherence to TRUSTS GA Deliverable & Tasks Descriptions

TRUSTS Task		**Respective Document Section(s)**	**Justification**
T3.5 Platform Development & Integration	Based on the outcomes of T2.4, this task focuses on the implementation, testing, and deployment of the TRUSTS platform components. Prior to the release of D2.4A, this task is expected to collaborate with T2.1, 2.2, and 2.3 to prepare a smooth start of development in M6. The task makes use of the infrastructure provided by T3.1. Assets from existing platforms (IDS, DMA) will be reused, enhanced, and adapted to cover the specifications of T2.4. This gives the task ahead direction by building on established and proven technologies. While, from an implementation point of view, this task covers general functionality (e.g., datasets and participant registrations), T3.2, 3.3, and 3.4 extend this functionality by providing specific state-of-the-art implementations that address the TRUSTS objectives. To that end, another goal of this work package is to integrate these contributions into a common TRUSTS platform.	Chapter 2 Chapter 3 Chapter 4 Chapter 5 Chapter 6	List of functional requirements, with priority information included. Overview of the platform's artefacts and components. Snapshots of the platform at various times in the project's final year. Workflows of TRUSTS v1.0 and remarkable features. Implementation details, an interoperability experiment, and an effort assessment of core components. Conclusion
TRUSTS Deliverable			
D3.11 Platform Status Report III *This deliverable reports on the current state of the platform regarding its functionality and operational parameters (e.g., number of integrated datasets, transactions, detected events, error rates). While platform releases are happening continuously, following the DevOps principles, these reports act as "snapshot" documentation of the platform.*			

1.2 Consecutive platform status reports in comparison

This deliverable is the third and last report of the TRUSTS platform status report. This section summarises the content-based differences in the second platform status report. The content of Chapters 2 to 6 is considered in Table 2.

Table 2: The content-based differences between this deliverable and the previous one

Section	D3.10 (previous version)	D3.11 (current version)
2	The last version of the FRs specification is presented.	The last version of the FRs specification as well as the FR's priorities are presented.
3	The core artefacts and components from well-known initiatives and projects that are reused in MVP.v2 are introduced.	Artefacts and components from known initiatives and projects, as well as those developed during the project's lifetime that are (re-)used in the end-of-project version of the TRUSTS platform, are introduced.
4	MVP.v2 is presented, including its overview, architecture, and workflows.	An architectural and functional overview of MVP.v3, MVP.v4, and TRUSTS v1.0 is given. The workflows for TRUSTS v1.0 are also presented, as well as the remarkable features of the TRUSTS platform among existing datamarkets.
5	Describing shortly the evolution with agile methods and highlighting the current implementation status with changes to the components and how they cover the updated FRs as well as their implementation status. Snapshots of the landing page are shown.	Briefly describe the project management method. Discussion of the up-to-date changes to the platform components addressing the FRs as well as showing how to simulate an external data market using a TRUSTS clone. Then, it ends with an assessment of the operator's effort for the core components.
6	Summarising D3.10 and giving an overview of the next actions towards D3.11.	Summarising D3.11.

1.3 Deliverable Overview and Report Structure

This report outlines the platform's implementation history in the last year of the project. For this, the status of the platform at different points in time is presented. This report consists of six main chapters. After the introductory chapter, the third and final version of the platform status report is structured as follows:

2. **TRUSTS functional requirements**. The TRUSTS FRs specification is presented in Chapter 2. In D2.3 the TRUSTS FRs specification is explained in detail.
3. **TRUSTS platform components**. Several artefacts and components of the TRUSTS platform, either from well-known projects and initiatives or developed as part of the TRUSTS project, are introduced in Chapter 3. The initiatives and projects mentioned are the DMA, the IDS, and the CKAN project.
4. **The TRUSTS platform**. An architectural and functional overview of the third and fourth version of the MVP is given. In addition, the status of the TRUSTS platform at the end of the project's life cycle is presented. In the latter, the focus is on the architecture and the workflows. Remarkable features of the TRUSTS platform are also highlighted in this chapter.
5. **Implementation**. A brief explanation of the platform's development with agile methods is given. The FRs and the TRUSTS platform components are set in relation while technical information details are provided. Also, an experiment is presented, conducted within the last year of the project, that demonstrates the exchange of metadata between an external operator and the TRUSTS platform. Finally, the effort of a potential TRUSTS platform operator for different components is discussed.
6. **Conclusion**. The report concludes by summarising the report and by highlighting the achievements and lessons learnt during the development of the platform and the further benefits of the same.

2 TRUSTS functional requirements

In D2.3 [5], a comprehensive list of the FRs is specified. This chapter provides a table with the latest version of FRs, which can also be found in D2.3. The following information is provided in Table 3: (1) a unique identifier (ID) for each requirement, (2) a description of the corresponding FR, and (3) the FR's priority. The priority of a FR is either A or B. A and B are placeholders with the following meaning:

A. full implementation
B. partial or best effort implementation

The priority of a FR was defined in collaboration with the whole project consortium. Each WP in the TRUSTS project was asked to assess the priority of each FR. According to the result, the FRs were assigned priorities A and B accordingly.

FR38 - FR44 are not discussed further in this report as they have been clearly assigned to other WPs.

Table 3: TRUSTS functional requirement specifications

Req. ID	Description	Priority
Datasets and services onboarding functionality and processes		
FR1	The system should provide standardized API descriptions for enabling providers to onboard their datasets and services	A
FR2	The system should provide APIs that enable its interoperability/federation with other industrial marketplaces and external sources	A
FR3	The system should be able to provide datasets and services descriptions	A
FR4	The system should provide reference mechanisms to open data from 3rd sources, so as to make it available as an option through its data exploration, profiling and provision mechanisms.	A
Intelligent data/service exploration and correlation functionality and processes		
FR5	The system should provide rich search mechanisms across all federated nodes for available datasets and services.	A
FR6	The system should be able to provide datasets and services recommendations to its' users pertaining to their profile and needs	A
FR7	The system should employ matchmaking mechanisms through which hosted datasets are matched with hosted services (e.g., suitable for their analysis or processing / customization) and vice versa.	A

Modified FR8	The system should identify and match related datasets so as to provide combined and enriched data, with services that allow processing / customization of datasets.	A
FR9	The system should be able to improve datasets and services profiles based on extracted information originating from the available data	A
Purchasing transactions and billing		
FR10	The system should provide contract mechanisms as a validation means of sellers/buyers agreements	B
Modified FR11	The system should ensure the integrity and authenticity of the smart contracts transactions signed by its users. A dispute management process may be designed.	B
ModifiedFR12	Smart contracts in the system should be accompanied by a human friendly representation (i.e natural language)	B
Modified FR13	Mechanisms to make signed smart contracts be legally valid, enforceable and interpretable will be investigated	B
Modified FR14	The system should encompass mechanisms for keeping transactions from being infringed	A
Modified FR15	The system should provide the ability to connect to billing mechanisms for enabling consumers to pay providers according to the agreed smart contract	B
FR16	The system must provide alternative and flexible pricing models taking into consideration the diversity of the available datasets and services	B
FR17	The system should provide brokerage mechanisms for addressing the offerings and demands of the hosted datasets and services	A
NFR 1	The system must provide alternative subscription contracts to the TRUSTS platform subscribers.	B
NFR 2	The system should provide a set of templates to the asset owners to describe the T&C for using the respective asset through TRUSTS.	B
NFR 3	Prior to procuring a dataset, information on the dataset should be provided (including data characteristics, attributes, ownership, etc.) including a sample of the data set for pre-purchased testability and preview	B
(Meta-)Data Governance		
FR18	The system should provide explicit metadata information for each dataset or service that is accommodated in the platform	A

FR19	The system should incorporate models, ontologies and taxonomies for the classification and semantic representation of the accommodated datasets and services in the platform.	A
FR20	The system should be able to incorporate well established or standardized ontologies from different scientific, industrial and business domains for the description of the semantic representation of the hosted datasets and services.	A
FR21	The system should provide mechanisms capable of identifying the provenance of the hosted datasets.	A
FR22	The system should provide mechanisms capable to identify the lifecycle of the hosted datasets.	B
FR23	The system should harvest metadata from external sources.	A
FR24	The system should be able to provide semantic information even for unstructured datasets.	A
FR25	The system should be able to keep continuously updated profiles of the hosted datasets and services based on related interactions performed with the system.	A
FR26	Dataset discovery should be based on the FAIR principle.	A
Data as a Service and Subscribers management		
FR27	TRUSTS datasets and services should be provided to the users on demand, regardless of geographic or organisational separation between provider and consumer taking into account all potential territorial legislation/regulatory restrictions.	B
FR28	TRUSTS should be able to be deployed as a federation of distributed, interconnected and interoperable nodes.	B
FR29	The system should enable its users to explore data and services openly, providing public descriptions. However, purchased data and services need to be exchanged point-to-point, between the seller and the buyer.	A
FR30	The system should support mechanisms for users' (producers/consumers) subscription opting different schemes (e.g., annual, monthly, etc.) and authentication.	A
FR31	The system should support corporate accounts that fall under one subscription/enrolment per organisation.	B
FR32	The system should enable authorized users to create, read, update, and delete (withdraw or make unavailable) datasets, services and user profile records.	A
FR33A	The system should provide validation criteria for the new enrolled users.	B

FR33B	The system should provide reputation/rating schemes with regard to available datasets and services.	B
FR34	The system should allow consumers to announce their need for specific datasets / services if there are not any available, already.	B
FR35	The system should provide notifications regarding datasets / services updates to users that have granted access to them.	A
FR36	The system should provide easy to use UIs (ensuring effectiveness, efficiency and user satisfaction) that will help users to accomplish their tasks effectively and prevent them from committing errors.	A
Modified FR37	TRUSTS UIs and workflows must follow a business-wise rational (e.g., one stop shop), for coherently mapping the market's needs. Indicative services to be included: registration, advanced search, buy/sell data, use/provide service, browse data/service catalogue, choose contract, and upload/download datasets.	A
NFR 4	TRUSTS UIs should be personalized.	B
NFR 5	TRUSTS should provide clear help function and documentation.	B
NFR 6	The TRUSTS platform and service should be scalable.	A
Data protection		
Modified FR38	The system should support collaboration between parties while preserving the privacy of the data. Methods that enables data privacy preserving on parties' collaboration will be provided by the system.	B
FR39	The system should provide de-anonymization attack assessment and risk analysis for the private / sensitive datasets to be onboard.	B
Modified FR40	The system should employ anonymization tools and guidelines for data anonymization. Information about anonymization of a dataset should be provided upon client request.	B
FR41	The system should provide means for converting algorithms that might compromise the data privacy into safe privacy preserving ones without harming their functionality.	B
Advanced data analysis based on Machine Learning		
FR42	The system should incorporate well established ML algorithms that can be used by the TRUSTS customers for data analysis and classification.	B
FR43	The system must incorporate a secure infrastructure for the distributed analysis of data based on ML approaches	B

Trusted and legitimate data flows		
FR44	Mechanisms provided by the TRUSTS platform regarding personal data, non-personal data and services exploration, exchange agreements and purchase, should be compliant with the following regulations (when applicable): ● General Data Protection Regulation ● e-Privacy regulation, for electronic communications ● Free Flow of Non-Personal Data Regulation, for data exchange between the TRUSTS platform and subscribers ● Platform-to-Business Regulation, for safeguarding TRUSTS' operational transparency and fairness. Mechanisms provided ensuring that local laws apply to each federated node. Predefined contracts should exist.	A

3 TRUSTS platform components

Chapter 3 briefly presents and summarises the components from known projects and initiatives that have been reused in TRUSTS, as well as the components developed as part of the TRUSTS project. In addition, technical information is provided for each component, including existing dependencies with other components, FRs covered, end-to-end functionalities, interface descriptions, and information about which product version the component is available in.

3.1 Artefacts and components from well-known projects and initiatives

Section 3.1 introduces the components developed in the DMA and IDS projects, as well as in CKAN, that have been reused in TRUSTS. Compared to the previous version of this document, the Dynamic Attribute Provisioning Service and the Vocabulary Service have been added to the list of reused components.

3.1.1 IDS Dataspace Connector

The IDS Dataspace Connector (DSC) is a core component of the TRUSTS platform. In Table 4, the DSC component is presented in short. FRs mentioned are partially covered by this component, but to have full FRs implementations, other components were needed.

The DSC is a core component of the TRUSTS platform. It covers the main functionalities for the platform and has a rich, well documented, and easy to use REST API interface. The DSC can keep all the information about node's catalogues and signed agreements in the local database. Furthermore, this component has the following functionalities:

- Supports IDS Informational Model (IM)
- Creation of access rules (policies) to resources (data assets)
- Creation of catalogues for data assets
- Creation of data assets
- Definition of data assets' metadata
- Creation of an offer for data assets
- Agreement negotiation and conclusion
- Supports secured communication with another DSC and IDS metadata broker
- Sending the connector and catalogue information to Metadata Broker
- Searching for data assets
- Access control to local data assets
- Subscription to data asset modification
- Notification for data asset modification to the consumer
- Easy deployment using docker-compose

Table 4: IDS Dataspace Connector

Connector			
IDS Dataspace Connector			
Dependencies	CKAN, Metadata Broker, DAPS		
Functional requirements	FR1, FR3, FR5, FR18, FR28, FR29, FR32, FR35		
End-to-end functionalities	1.	Security Present certificate for mutual identification and TLS encryption	
	2.	Create catalogue, resource, presentation, offer, agreement, and push to Metadata Broker.	
	3	Contract enforcement (usage data asset control)	
	4	Peer to Peer communication between nodes	
	5	Contract (Agreement) communication with CKAN	
	6	Trigger notification action on data asset update	
	7	Routing to applications and services	
	8	Synchronise vocabularies across the platform	
Interface description	REST API		
Integrated in	MVP.v1	☐ Yes [✓ No]	☐ Updated
	MVP.v2	[✓ Yes] ☐ No	☐ Updated
	MVP.v3	[✓ Yes] ☐ No	☐ Updated
	MVP.v4	[✓ Yes] ☐ No	[✓ Updated]
	TRUSTS.v1.0	[✓ Yes] ☐ No	[✓ Updated]

3.1.2 Metadata Broker and Metadata Storage

The Metadata Broker and Metadata Storage are two closely coupled components that serve as a centralised repository of metadata about assets throughout the platform. The Metadata Storage is implemented as an Apache Jena Fuseki triple store, which has different Resource Description Framework (RDF) graphs corresponding to the assets being served by different nodes in the TRUSTS platform. The Metadata Broker is a lightweight service that wraps the Metadata Storage in an API that is compatible with the IDS information model. Namely, the Metadata Broker can receive IDS messages from the DSC deployed in different nodes, validating their authenticity (via the accompanying DATs) and forwarding them to the underlying triple store.

Table 5: Metadata Broker and Metadata Storage

Metadata Broker and Metadata Storage			
Metadata Broker			
Dependencies			
Functional requirements	FR3, FR5, FR18, FR19, FR20, FR21, FR22, FR25, FR26, FR28, FR29, NFR6		
End-to-end functionalities	1	Onboarding of assets (Dataset, Application or Service) into the TRUSTS platform	
	2	Search for an asset in the TRUSTS platform.	
	3	Make a peer-to-peer connection between a potential buyer of an asset and its provider to negotiate contracting terms.	
Interface description	API compliant with the IDS Communication Protocol 2		
Integrated in	MVP.v1	☐ Yes [✓ No]	☐ Updated
	MVP.v2	[✓ Yes] ☐ No	☐ Updated
	MVP.v3	[✓ Yes] ☐ No	☐ Updated
	MVP.v4	[✓ Yes] ☐ No	☐ Updated
	TRUSTS.v.1.0	[✓ Yes] ☐ No	☐ Updated

3.1.3 Platform Interface (CKAN)

The platform interface is a web-accessible user interface that casual users of the TRUSTS platform can use to access dataset and services being offered. It is designed based on the needs of individual users who do not have the means or the ability to install a full-blown TRUSTS node on their premises, and who have no need to offer services or applications on the platform, nor consume the latter. It is based on the well-known CKAN data management platform, with a series of extensions and customizations that allow it to interact with the rest of the TRUSTS components.

Table 6: Platform Interface

Platform Interface		
CKAN		
Dependencies	DSC, Recommender System, Vocabulary Service, Smart Contract Component	
Functional requirements	FR1, FR2, FR3, FR5, FR6, FR10, FR15, NFR3, FR18, FR19, FR20, FR21, FR22, FR24, FR25, FR26, FR27, FR28, FR29, FR31, FR35, FR36, FR37, NFR4, NFR6	
End-to-end functionalities	1	Onboarding of datasets into the TRUSTS platform through a user-friendly interface
	2	Searching for assets in the TRUSTS platform

	3	Defining access policies for assets (e.g., number of times it can be accessed, validity period, etc.)	
	4	Receive recommendations about assets in the TRUSTS platform	
	5	Accept contracts for assets in the TRUSTS platform	
	6	Receive access details for services acquired through the TRUSTS platform	
	7	Consume datasets acquired through the TRUSTS platform	
Interface description	A web-accessible user interface, and a OpenAPI-documented REST API.		
Integrated in	MVP.v1	☐ Yes ☑ No	☐ Updated
	MVP.v2	☐ Yes ☑ No	☐ Updated
	MVP.v3	☑ Yes ☐ No	☐ Updated
	MVP.v4	☑ Yes ☐ No	☐ Updated
	TRUSTS.v.1.0	☑ Yes ☐ No	☐ Updated

3.1.4 Corporate Interface (CKAN)

The corporate interface is a web-accessible application that is to be deployed on the premises of organizations participating in the TRUSTS platform and accessible only through their private networks. It serves as the main point of access for users belonging to said organisation into the TRUSTS platform, and it is through this component that they can onboard, offer, search for, and consume assets, as well as receive recommendations for further assets, and inspect the lifecycle of their uploaded assets.
It is based on the well-known CKAN data management platform, with a series of extensions and customizations that allow it to interact with the rest of the TRUSTS components.

Table 7: Corporate Interface

Corporate Interface		
CKAN		
Dependencies	DSC, Recommender System, Vocabulary Service, Smart Contract Component	
Functional requirements	FR1, FR2, FR3, FR5, FR6, FR10, FR15, NFR3, FR18, FR19, FR20, FR21, FR22, FR24, FR25, FR26, FR27, FR28, FR29, FR31, FR35, FR36, FR37, NFR4, NFR6	
End-to-end functionalities	1	Onboarding of datasets, services, and applications into the TRUSTS platform through a user-friendly interface
	2	Onboarding of large amounts of assets into the TRUSTS platform through an API
	3	Searching for assets in the TRUSTS platform

	4	Defining access policies for assets (e.g., number of times it can be accessed, validity period, etc.)	
	5	Receive recommendations about assets in the TRUSTS platform	
	6	Accept contracts for assets in the TRUSTS platform	
	7	Receive access details for services acquired through the TRUSTS platform	
	8	Consume datasets acquired through the TRUSTS platform	
	9	Download configuration and deployment parameters (docker-compose files) for applications acquired through the TRUSTS platform.	
Interface description	A web-accessible user interface, and a OpenAPI-documented REST API.		
Integrated in	MVP.v1	☑ Yes ☐ No	☐ Updated
	MVP.v2	☑ Yes ☐ No	☐ Updated
	MVP.v3	☑ Yes ☐ No	☐ Updated
	MVP.v4	☑ Yes ☐ No	☐ Updated
	TRUSTS.v.1.0	☑ Yes ☐ No	☐ Updated

3.1.5 Landing Page

The Landing Page provides the user with initial information about the TRUSTS platform. It is the first page a user interacts with when visiting the TRUST platform. It is vital that this page is easy to use and pleasing to the eye, since first impressions are very important on how a user will end up perceiving the entire platform later. Thus, a great amount of time has been dedicated to designing such a page, and even more on redesigning it after receiving feedback from the Use Cases.

Table 8: Landing Page

Landing Page		
CKAN		
Dependencies	DSC, Recommender System, Vocabulary Service, Smart Contract Component	
Functional requirements	FR36, FR37	
End-to-end functionalities	1	Onboarding of datasets, services, and applications into the TRUSTS platform through a user-friendly interface
	2	Searching for assets in the TRUSTS platform through a visually pleasing UI.

	3	Provide ways to define access policies for assets (e.g. number of times it can be accessed, validity period, etc.) via dropdown menus	
	4	Receive recommendations about assets in the TRUSTS platform in the form of carousel cards while searching for other assets.	
	5	Provide UI to accept contracts for assets in the TRUSTS platform	
	6	Receive access details for services acquired through the TRUSTS platform in a user-friendly UI.	
	7	Consume datasets acquired through the TRUSTS platform by using the download links/buttons provided.	
	9	Manage user account and access token	
	10	Manage corporate node's visuals and key configuration parameters.	
	11	Provide a help-page to guide the user through well written documentation accompanied with step-by-step pictures.	
	12	Visually appealing dashboards that help the user paint a complete picture over his account and the status of it.	
Interface description	TRUSTS UIs		
Integrated in	MVP.v1	☐ Yes ☑ No	☐ Updated
	MVP.v2	☑ Yes ☐ No	☐ Updated
	MVP.v3	☑ Yes ☐ No	☐ Updated
	MVP.v4	☑ Yes ☐ No	☑ Updated
	TRUSTS.v.1.0	☑ Yes ☐ No	☐ Updated

3.1.6 IDSA DAPS

IDSA DAPS (Dynamic Attribute Provisioning Service) has a very specific role in TRUSTS. This component controls the communication between main parts of the platform (DSC and Metadata Broker). Any communication in the platform cannot start without authentication through DAPS.

DAPS checks the validity of participant certificates and is used for mutual authentication between participants. Certificates, that are issued for any TRUSTS participant, are the entry ticket to the platform and are used for secured encrypted communication in the platform. See Table 9 for more technical details on this component.

Table 9: IDSA DAPS

Dynamic Attribute Provisioning Service			
IDSA DAPS			
Dependencies	DSC, Metadata Broker		
Functional requirements	FR11, FR21, FR32		
End-to-end functionalities	1.	Security Present certificate for mutual identification and TLS encryption	
	2.	Participants authentication	
Interface description	REST API		
Integrated in	MVP.v1	☑ Yes ☐ No	☐ Updated
	MVP.v2	☐ Yes ☑ No	☐ Updated
	MVP.v3	☑ Yes ☐ No	☐ Updated
	MVP.v4	☑ Yes ☐ No	☐ Updated
	TRUSTS.v1.0	☑ Yes ☐ No	☐ Updated

3.1.7 Vocabulary Service

The Vocabulary Service is a central point of vocabulary management for the TRUSTS platform. It allows for operators to define, in a single point, the different vocabularies that will be used to describe the different assets in the TRUSTS platform, such as themes, data types, keywords, etc. It is designed around the linked data principles, in the sense that each concept in the vocabularies is an RDF resource which can be linked to other resources in the Linked Data Cloud. The vocabularies managed through this service are distributed to the different platform interfaces via either text-serialised RDF graphs (e.g., Turtle files), or a SPARQL endpoint. Using either of these, platform interfaces can keep an updated version of the vocabularies to display to users when they onboard assets, as well as when they do faceted search of them.

Table 10: Vocabulary Service

Vocabulary Service		
Vocabulary Service		
Dependencies		
Functional requirements	FR19, FR20, FR21, FR24, FR26	
End-to-end functionalities	1	Search for assets in the TRUSTS platform
	2	Onboard assets
Interface description	Web accessible UI, REST Interface, SPARQL Endpoint	

Integrated in	MVP.v1	☐ Yes ☑ No	☐ Updated
	MVP.v2	☐ Yes ☑ No	☐ Updated
	MVP.v3	☑ Yes ☐ No	☐ Updated
	MVP.v4	☑ Yes ☐ No	☐ Updated
	TRUSTS.v.1.0	☑ Yes ☐ No	☐ Updated

3.2 Artefacts and components developed as part of the TRUSTS project

Section 3.2 introduces the components developed during the lifetime of the TRUSTS project. The components described in the following subsections were developed in the tasks of WP3. They are essential results of the same and are explained in more detail in the associated task deliverables.

3.2.1 Recommender System

The recommender system component enables brokerage activities within the TRUSTS platform. Thus, via personalised recommendations, the different entities in the platform (i.e., users, datasets, services) are interlinked. Please note, that a user could also be a corporate entity, and a service could also be an application entity. We provide three different types of recommendation algorithms, namely Most Popular, Collaborative Filtering, and Content-based Filtering, that are further described in D3.13. Additionally, our recommender system supports six different recommendation use cases or end-to-end functionalities, which are listed in the following table:

Table 11: Recommender System

Recommender System		
Recommender System		
Dependencies	DSC, Metadata Broker, CKAN	
Functional requirements	FR6-9, FR17, FR25-26, FR34	
End-to-end functionalities	1	Recommendation of Datasets to Users
	2	Recommendation of Services to Users
	3	Recommendation of Datasets to Services
	4	Recommendation of Services to Datasets
	5	Recommendation of Datasets to Datasets
	6	Recommendation of Services to Services
Interface description	REST API	

Integrated in	MVP.v1	☐ Yes ☑ No	☐ Updated
	MVP.v2	☐ Yes ☑ No	☐ Updated
	MVP.v3	☑ Yes ☐ No	☐ Updated
	MVP.v4	☑ Yes ☐ No	☐ Updated
	TRUSTS.v.1.0	☑ Yes ☐ No	☐ Updated

3.2.2 Smart Contract Component

The Smart Contract Executor component enables use of blockchain and smart contract functionality over a remote API. This was built on top of the Hyperledger Fabric (HLF) development kit, which does not expose a remote API by default. Therefore, a remote API was built on the Hyperledger gateway as a REST interface. Some Hyperledger Fabric sample smart contracts are included in the component package, as well as custom-made smart contracts to allow for user scoring, such as ratings and reviews of users and assets. A graphical user interface named Explorer UI is included in the component package to view the blockchain in a more human-readable way. The component also implements three separate, distinct blockchain search mechanisms; one included as part of the Explorer UI, the other included as one of the HLF sample smart contracts, and the third is highly portable - developed on top of the remote API which provides granular search functionality independent of the underlying blockchain technology. Rapid deployment scripts and an installation package including comprehensive documentation allows the component to be installed in approximately one hour on a new system by a user with general technical experience, and the containerized nature of the component supports platform portability.

Table 12: Smart Contract Executor

Smart Contract Executor		
Smart Contract Executor		
Dependencies	TRUSTS Dependencies N/A, External Dependencies Hyperledger Fabric	
Functional requirements	FR10, FR11, FR14	
End-to-end functionalities	1	Standard blockchain & ledger operations (read / write etc)
	2	Explorer graphical user interface for blockchain with blockchain search
	3	Blockchain query, basic asset transfer, secured agreement smart contracts
	4	User scoring smart contracts, assign ratings / reviews to assets & users
	5	Custom remote API built on Hyperledger SDK with API blockchain search
	6	Rapid installation procedure with containerized system
Interface description	REST API, GUI, command terminal (on same host)	

Integrated in	MVP.v1	☐ Yes [✓ No]	☐ Updated
	MVP.v2	☐ Yes [✓ No]	☐ Updated
	MVP.v3	☐ Yes [✓ No]	☐ Updated
	MVP.v4	[✓ Yes] ☐ No	☐ Updated
	TRUSTS.v.1.0	[✓ Yes] ☐ No	☐ Updated

3.2.2.1 Payment Compatibility Demonstrator

The Payment Compatibility Demonstrator was developed to showcase how the TRUSTS system can connect to external payment systems. This was created as an expansion module to the Smart Contract Executor remote API, meaning it is independent of the underlying blockchain technology. The remote API search is leveraged for the asset ID input to this component. If the asset exists, its information is loaded into a web UI to be displayed to the user. If the metadata displayed is satisfactory to the user, the user can select an external payment provider using radial buttons and click the purchase button. The user will then be brought to the external payment system, and once they have completed the purchase, they load the payment Universally Unique Identifier (UUID) into the blockchain of the Smart Contract Executor. The transfer of the asset then proceeds, and the process is complete. The Payment Compatibility Demonstrator satisfies FR15.

3.2.3 Interoperability Component

The interoperability component enables exchange of metadata with third-party providers, for either external data markets or the European Open Science Cloud (EOSC). It has two major components, the “TRUSTS platform client” and “connectors” for two EOSC initiatives. For interoperability with external data markets, we provided an API, the before-mentioned TRUSTS platform client. Operators of external data markets can use this client to programmatically upload their metadata, i.e., to batch-load metadata of large amounts of datasets into TRUSTS.

EOSC is not a single platform, instead it is a multitude of different initiatives coming together under the umbrella of the EOSC. Each initiative has its own technology stack and interfaces. Consequently, we decided to exemplarily focus on two important EOSC initiatives and selected OpenAIRE and Europeana, for which we built connectors. These connectors use an ETL process (extract/transform/load) to first acquire metadata from either OpenAIRE or Europeana, subsequently transform it into a TRUSTS-compatible format, and lastly load it into TRUSTS. The connectors also use the before-mentioned TRUSTS platform client to finally load the data into TRUSTS.

Additionally, it should be mentioned that TRUSTS can also interoperate with other CKAN-based platforms, since it is itself based on CKAN. There are ready-built harvesting extensions in the CKAN ecosystem that provide this functionality [6].

Table 13: Interoperability Component

Interoperability Component			
Interoperability Component			
Dependencies	CKAN, IDS-IM		
Functional requirements	FR2, FR4, FR20, FR23		
End-to-end functionalities	1	TRUSTS platform client to load metadata into TRUSTS	
	2	Connector to harvest metadata from Europeana and OpenAIRE	
Interface description	Python library, REST API		
Integrated in	MVP.v1	☐ Yes [✓ No]	☐ Updated
	MVP.v2	☐ Yes [✓ No]	☐ Updated
	MVP.v3	[✓ Yes] ☐ No	☐ Updated
	MVP.v4	[✓ Yes] ☐ No	☐ Updated
	TRUSTS.v.1.0	[✓ Yes] ☐ No	☐ Updated

3.2.4 Notification Service

The Notification Service is used for sending emails to users that are defined in any data asset contract who should be informed when the data asset was changed.

The DSC keeps information about existing contracts and who should be informed when the asset is changed. Changes to data asset trigger request from the DSC to the note-service endpoint for email sending to corresponding user's email address.

The notification service in response sends email through the email service.

Table 14: Notification Service

Notification Service			
note-service			
Dependencies	DSC, SMTP email service		
Functional requirements	FR35		
End-to-end functionalities	1	Receive requests for email from DSC	
	2	Create and send email to the email address from the incoming message	
Interface description	REST API, SMTP		
Integrated in	MVP.v1	☐ Yes	☐ Updated

		☑ No	
	MVP.v2	☐ Yes ☑ No	☐ Updated
	MVP.v3	☐ Yes ☑ No	☐ Updated
	MVP.v4	☑ Yes ☐ No	☐ Updated
	TRUSTS.v.1.0	☑ Yes ☐ No	☐ Updated

4 The TRUSTS platform

Chapter 4 presents snapshots of the platform at various times during the final year of the project. Section 4.1 refers to the MVPs of the TRUSTS platform. The MVP versions, developed in the first and second year of the project are briefly explained, while more information about these versions can be found in the earlier versions of this report: D3.9 [13] and D3.10 [14]. MVP.v3 was released in July 2022 (M31). The release of MVP.v4 followed in September 2022 (M33). Both versions are introduced as well. The respective architecture, differences from previous versions, and core components are highlighted in the corresponding subsections. Section 4.2 highlights the state of the TRUSTS platform at the end of the project. An overview is given, the architecture, and respective workflows of the platform are described. Lastly, Section 4.3 shows the unique characteristics and features of TRUSTS among the existing data markets.

4.1 Minimum Viable Products of the TRUSTS platform

An MVP [3, 4] is a version of a product with an elementary set of implemented FRs to obtain feedback quickly and with the least possible effort. Accordingly, feedback can be considered, and the product can be promoted to (potential) customers at an early stage of development.

During the project lifetime, five MVP versions were developed. Each with an extended set of functionalities. The publication periods can be taken from Figure 1.

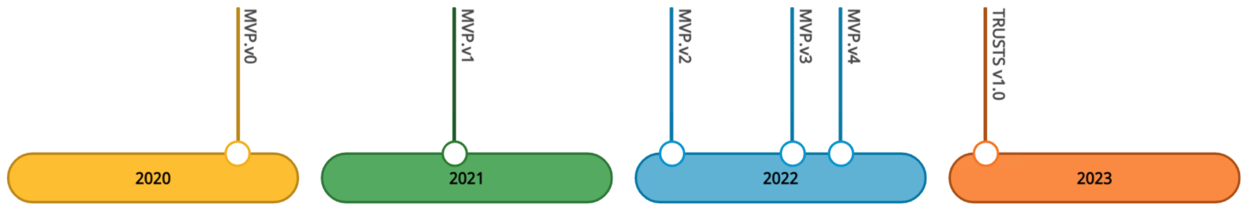


Figure 1: MVP releases timeline

In the third and final year of the project, the period covered by this status report, MVP.v3 and MVP.v4 were released and are further explained, in the following subsections, along with how these differ from their respective predecessor versions. In addition, the previous versions (MVP.v0, MVP.v1, and MVP.v2) are briefly outlined. D3.9 and D3.10 provide more detailed information on these.

4.1.1 Previous MVP versions

This subsection briefly presents snapshots of the platform before starting the development of MVP.v3. The interested reader is referred to D3.9 and D3.10 for more detailed information on the earlier versions.

MVP.v0. The initial MVP version, MVP.v0, consisted of two nodes: Server Node and Client Node. It also consisted mainly of the following three components: IDS Trusted Connector, IDSA DAPS, and Metadata Mapper. MVP.v0 was developed with the aim of defining a roadmap for further development, identifying core components, and gathering initial experience, especially with components to be reused from known projects and initiatives.

MVP.v1. With MVP.v1, the platform was presented to WP5 for the first time. WP5 was responsible for conducting the UC trials. This platform version had two types of nodes: Central Node and Corporate Node. Nodes communicate through the IDS Trusted Connector. In MVP.v1, a small set of features for secure data exchange were implemented, these included: offer datasets and services, secure communication, and the technical foundation for UI components. Main components of this version were the IDS Trusted Connector, IDSA DAPS, and CKAN.

MVP.v2. Even MVP.v2 was built as a network between independent nodes. It is distinguished between three types of nodes: Central Node, Corporate Node, and User Portal Node. The nodes are connected via secured communication channels based on mutual Transport Layer Security (TLS) authentication. A figure of the respective architecture of MVP.v2 is given in D3.10. In MVP.v2 the IDS Trusted Connector was replaced by the DSC. The range of functions were expanded compared to the predecessor version. New features and enhancements in MVP.v2 included, for example, the extended definition of data assets, creation of data asset catalogues, creation of data asset offers, searching mechanism, and data asset access control.

4.1.2 MVP.v3

MVP.v3 was presented as a more mature version of the TRUSTS platform. A couple of the core and main features and services of the TRUSTS project were implemented in this MVP. In the following, its overview will be discussed, differences from its preceding version are presented, and finally its architecture is demonstrated.

MVP.v3 Overview:

The development of the MVP.v3 started in February 2022 (M26) and concluded in July 2022 (M31). To extend what has been outlined about the types of nodes in the above MVP.v2 text, Figure 2 shows the MVP.v3 architecture which contains three types of nodes:

1. **Corporate Node**. Is used by partners of TRUSTS that have complex infrastructure and participate in TRUSTS. They can have many users and many services and applications that are provided or consumed via the TRUSTS platform. They can set any authorization system and communication structure inside their nodes. The TRUSTS Operator is not responsible for setting up or supporting instances of corporate nodes but will provide detailed instruction manuals and all necessary software. Among the components being set up, is the Corporate Interface, a web application through which the organisation’s users can interact with the TRUSTS platform.

2. **User Portal Node**. Is created to cover the needs of individual users of the TRUSTS platform. It is set up and maintained by the TRUSTS Operator. By accessing this node, individual users can benefit from some of the functionalities of the TRUSTS platform, without the complexity of setting up a corporate node. Additionally, it is the entry point for all organisations intending to join the TRUSTS platform, as it provides landing pages, legal information, and setup instructions. One of the components included in this node is the platform Interface, which acts as the main point of access to the above-mentioned functionalities. In principle, the TRUSTS platform could have more than one User Portal Node.

3. **Central Node**. Exists to support the operation of the whole TRUSTS platform, playing the role of authorization, monitoring, smart contract executor, catalogue, application repository, among others. This node is created and maintained by the TRUSTS operator.

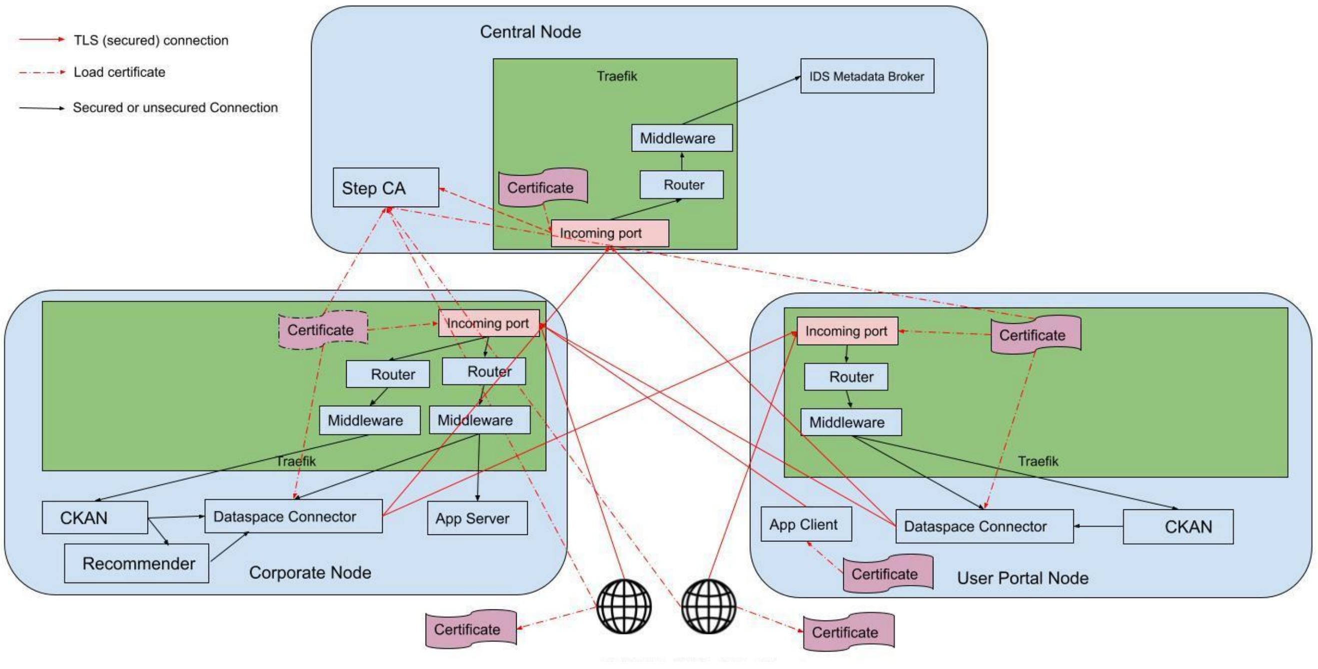


Figure 2: MVP.v3 architecture

MVP.v3 Architecture Description:

Figure 2 demonstrates the MVP.v3 architecture which can have three types of nodes: Central Node, Corporate Node, and User Portal Node. In the meanwhile, the first two types are mandatory and valuable to allow the communication between the TRUSTS operator (operates the Central Node) and the other business partners (have their Corporate Nodes). In the following text, components of each node will be discussed as well as the offered secure communication between them.

- **Components of Central Node:** The main component included in this node is the Metadata Broker acting as a centralised repository of the metadata about the digital assets to be shared and exchanged within TRUSTS. Step CA an online Certificate Authority (CA) for secure, automated X.509 and SSH certificate management, was tested during the MVP.v3 lifecycle for generation of SSL certificates but in the meanwhile the provided IDS certificates which were given by IDSA are the only ones which used in the IDS components practically.

- **Components of Corporate Node:** This is the bigger between the other nodes. It contains several services and components: DSC that supports metadata for assets and their offers and consulting the contracts between TRUSTS stakeholders; Recommendation and Brokerage Services; Application Server to serve data applications; Corporate Interface using CKAN to onboard datasets and services as well as enable searching them.

- **Components of User Portal Node:** Similar components names can be seen with those in the Corporate Node, meanwhile the focus in this node goes to the end-user. Therefore, the Portal Interface using CKAN is used to allow interaction between those users and TRUSTS. Dataspace Connecter as well is required to allow secure communications with the other nodes. Moreover, every application server provided in the Corporate Node can use its client version to allow human interaction.

- **Communication between Nodes:** IDS certificates can be created using the IDS DAPS component or it can be given by IDSA directly for research purposes. The cross-communication between the TRUSTS nodes is made using IDS secure communication using IDS certificates.

<u>MVP.v3 Updates and changes:</u>

MVP.v3 has more advanced features and services other than the ones in the MVP.v2. In the following, those updates and changes will be given.

- Providing implementations to achieve data harvesting from the EOSC (e.g., OpenAIRE and Europeana).
- UI improvements with taking in consideration to have the systems as business-wise rational.
- Enabling of a rich search mechanism across all federated nodes for the available datasets, services, and applications.
- The recommender system was integrated in MVP.v3 with its ability to fetch datasets/services/apps from the broker. Also, the CKAN UI was extended to show recommender results.
- Management of contracts in CKAN in accordance with IDS-contracts.
- Smart Contract was implemented but was not visible from the TRUSTS UI.
- Provision of vocabulary hub component which includes models, ontologies, and taxonomies for the classification and semantic representation of the accommodated data assets.
- Development of a REST API service for notification when the data asset has been updated.

4.1.3 MVP.v4

This is the successor version of the MVP.v3. The following text explores its overview and the difference between it and MVP.v3.

<u>MVP.v4 Overview:</u>

The development of MVP.v4 started in July 2022 (M31) and concluded in September 2022 (M33). Upgrading the MVP to version 4 served to integrate additional components, namely the interoperability component as well as the smart contract executor. In addition, according to the findings after the tests performed on MVP.v3, improvements have been made.

<u>MVP.v4 Architecture Description:</u>

Like MVP.v2 and MVP.v3, MVP.v4 consists of three interconnected nodes: Central Node, Corporate Node, and User Portal Node. Each node is owned and operated by a different organisation. The description of these nodes remains the same as in Section 4.1.2.1. To support the various functionalities of the platform, each node runs a standard set of components. The nodes are connected via instances of the DSC. The DSC can be seen as the central component of the architecture, as it distributes information throughout the network. Communication between DSCs is controlled by DAPS.

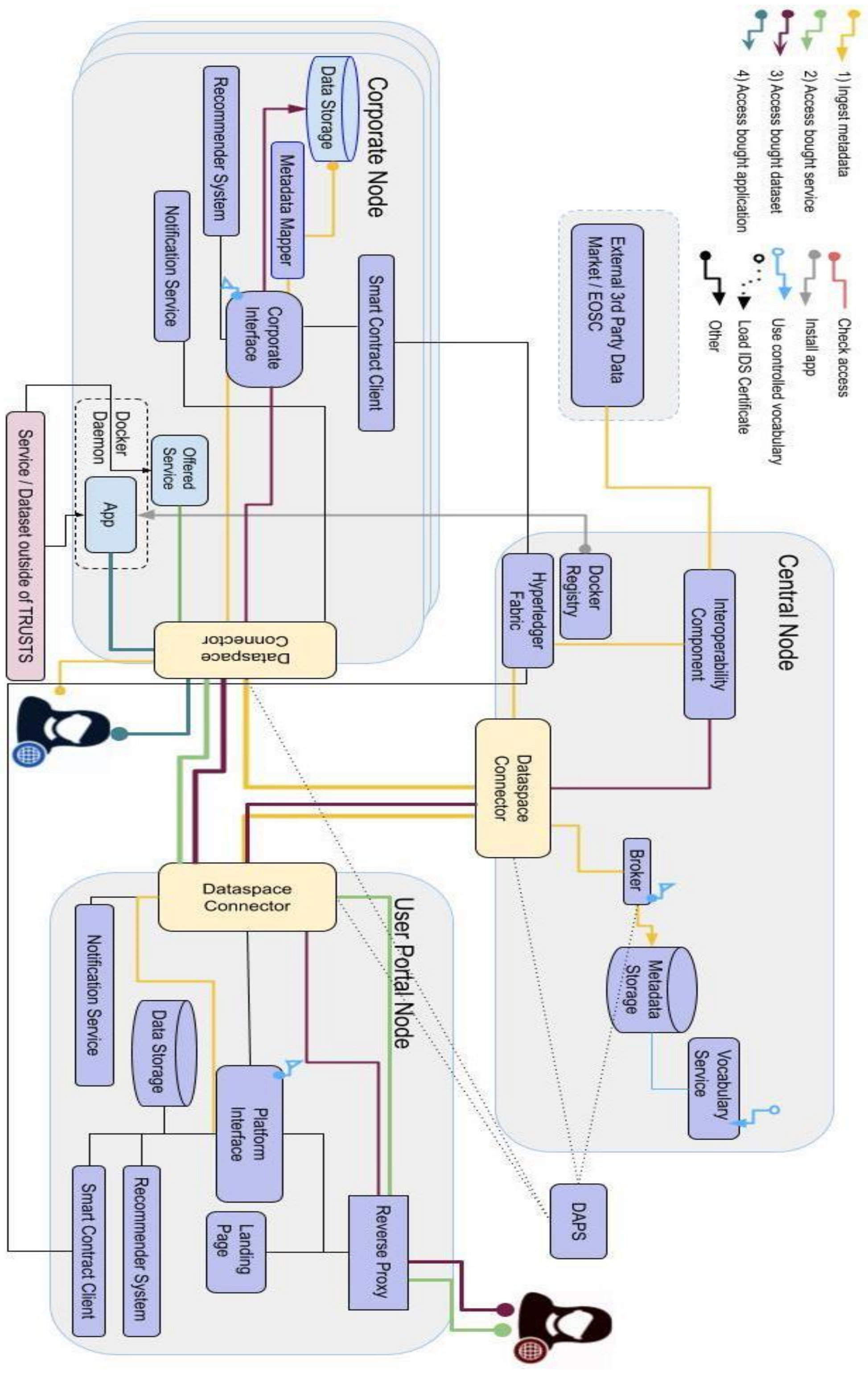


Figure 3: MVP.v4 architecture

The following is the new changes of the architecture of MVP.v4 comparing to its previous preceding version:

- **Components of the Central Node:** Metadata Storage component and its services are added to this node to store the TRUSTS metadata and to provide REST services for CRUD operations as well as to enable search and query functionalities. DSC is required as well to enable its main asset onboarding and to enable secure communication within TRUSTS nodes. Additionally, to enable exchange of metadata with third-party providers, for either external data markets or the EOSC and to use blockchain and smart contract functionality over a remote API, Interoperability component and Hyperledger Fabric are integrated, respectively. Besides, having Docker Registry to allow storing of the data applications.

- **Components of Corporate Node:** This is the bigger node to be installed in the business patterns' premises containing mandatory components and optional ones. The mandatory components are DSC, Recommender System, Smart Contract Client, Corporate Interface using CKAN, and Notification Service. Meanwhile, the optional components include Metadata Mapper for the internal data storage, Offered Services, and Data Apps provision.

- **Components of User Portal Node:** A couple of same components from the Corporate Node are added which have either same or similar functions, e.g., Smart Contract Client, Recommender System, Notification Service, and Data Storage.

- **Communication between Nodes:** IDS protocol is kept used to allow the secure communication between the TRUSTS nodes. Furthermore, Restful Web Services are used either to offer asset services or to allow further interaction between backend servers and their clients.

<u>MVP.v4 Updates and changes:</u>

The lifetime of the MVP.v4 was very short, around one and half a month. But that version has taken in consideration WP5 learnt lessons which was the result of testing the MVP.v3. Various of those issues and problems reported during that time have been tackled, and solutions for them have been identified. In the following, the updates and changes in this version are presented:

- Integration of the Smart Contract within the platform as well as storing the metadata in that component.
- Finalising the notification service.
- Providing a final version of the interoperability component that external partners can use.
- Simulation of the interoperability of TRUSTS with the external data markets using another version of TRUSTS called “TRUSTS Clone” implementation.
- Improving user friendliness of the platform.
- Continuation of work on the usage policies and solving certain related challenges to allow all the usage policy options given in the DSC.
- Development of a payment component demonstrator to show the compatibility of the platform with payment services.

4.2 TRUSTS v1.0

In this section, the end of the project version of the platform will be presented. In the first subsection the overview about the TRUST v1.0 and the new updates and changes in either the architecture or the development will be presented. Meanwhile, the second subsection explorers the workflow of the core and main functions, including their sequence diagrams. Finally, the last subsection tackles the unique characteristics and features of TRUSTS among the existing data marketplaces.

4.2.1 TRUSTS v1.0 - The Operator's TRUSTS Version

The following subsection lays out an overview about TRUST v1.0 (which was agreed by WP3 partners to be the delivered version to the operator), describes the one difference between its architecture and the previous version and finally, presents the updates and changes regarding the development and technical activities.

TRUSTS v1.0 overview:

After the end of WP5 UC trials, it was decided to give a new name for the final version as "TRUSTS v1.0" as has been explained above. The name of the version "TRUSTS v.1.0" was taken from the name of the project itself to highlight itself because of the project as well as to carry a newer name without the prefix "MVP". Even though it has a new name, it still provides the basic functionality of the platform since it needs further development to be at the market readiness level. In addition, to be in a correct stage, this version has not many differences with its predecessor MVP.v4. It can be seen as a refined version only of MVP.v4.

TRUSTS v1.0 architecture description:

At the end of the project, the final architecture of the TRUSTS project is proudly offered, with integrated services and components clearly shown. Figure 4 is the basic software including the basic functionalities which the project's operator can live with during the initial phase of the project start-up. Compared to its predecessor MVP.v4, the only component "Reserve Proxy" is added. Therefore, that only which can be reported about its architecture and how it differs with MVP.v4.

TRUSTS v1.0 updates and changes:

This version is considered as the final version by the end of the project. Few refinements took place. No further core or main developments were done. What can be identified meanwhile, the following refinements and changes:

- Refinement of the UI profile and administration pages to allow user-friendly interfaces.
- Proving open API descriptions of the platform's interfaces to allow easy understanding of each provided API and service in TRUSTS components.
- Finalising the user subscription from assets to allow sending of user notification when there is an update or changes to their subscribed digital assets.
- Planning of Reserve Proxy integration for all the nodes in the architecture to add more safety and security layers while allowing the communication between the TRUSTS nodes.

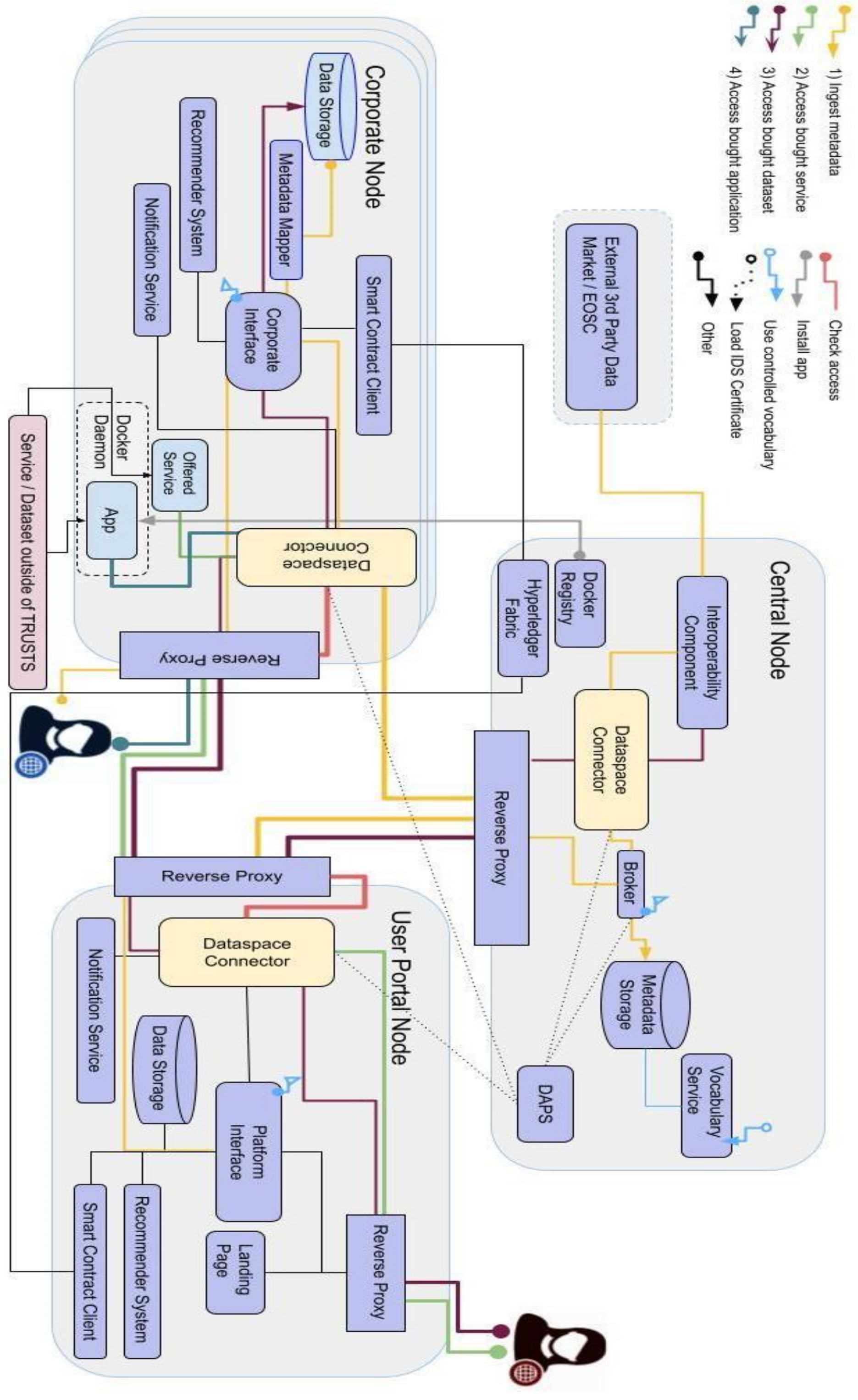


Figure 4: TRUSTS v1.0 architecture

4.2.2 Workflows and sequence diagrams

In the following subsections, workflows of the TRUSTS platform are illustrated using sequence diagrams and explained step by step. The main workflows are shown as well as workflows concerning specific components.

4.2.4.1 The main workflow functionalities

To show the main internal and external components' communications in the TRUSTS platform, the following covers the main eight workflow functionalities. Each functionality has a title to explain its actual process and a sequences diagram and its explanation:

Data Asset Onboarding

1. Data asset onboarding process starts from the creation of the User in the Provider node definition of data asset using CKAN UI.
2. CKAN in response to this user's action creates all metadata for data assets in the appropriate catalogue and then creates an offer for this data asset inside the Provider DSC.
3. User presses button "Publish" to promote the offer of data asset to the Metadata Broker
4. CKAN initiates the process of publishing an offer to the Broker, by sending a push command to the Provider DSC.
5. The Provider DSC requests security token from DAPS.
6. DAPS responses with token or error.
7. The Provider DSC publishes information about itself and an offer to IDS Metadata Broker, presenting the token.
8. The Metadata Broker checks The Providers Token. If it is OK, the Broker creates records about Provider's DSC and its offers and sends back confirmation. Otherwise, the Metadata Broker sends an error message back to the Providers DSC.
9. The Provider DSC forwards the Metadata Broker response to CKAN.
10. CKAN shows the result of the Metadata Broker to the User's screen.

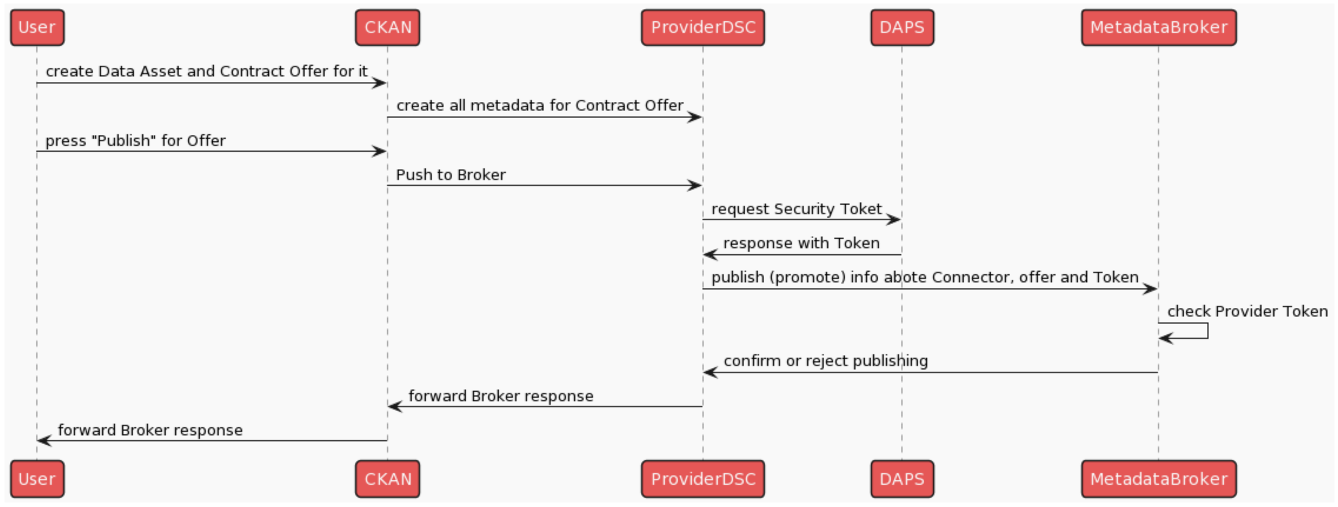

User
CKAN
ProviderDSC
DAPS
MetadataBroker
create Data Asset and Contract Offer for it
create all metadata for Contract Offer
press "Publish" for Offer
Push to Broker
request Security Toket
response with Token
publish (promote) info abote Connector, offer and Token
check Provider Token
confirm or reject publishing
forward Broker response
forward Broker response

Figure 5: Data Asset Onboarding

Data Asset Searching

A precondition to start a search for a data asset is an existing offer for this data asset in the Metadata Broker.

1. The User in the Consumer node inputs a search request for a data asset via the CKAN UI or selects one data asset from the recommended.
2. CKAN forwards the searching request to Consumer DSC.
3. The Consumers DSC requests a security token from DAPS.
4. DAPS sends back a token or error if it doesn't authenticate DSC.
5. The Consumer DSC transforms the search request to a SPARQL query, adds its token and sends the request to the Metadata Broker.
6. The Metadata Broker checks the Consumers Token and if it's correct executes the SPARQL query.
7. The Metadata Broker returns the searching result or error to the Consumer DSC.
8. Consumer DSC forward the Broker's response to CKAN.
9. CKAN shows the search result on the User's screen.

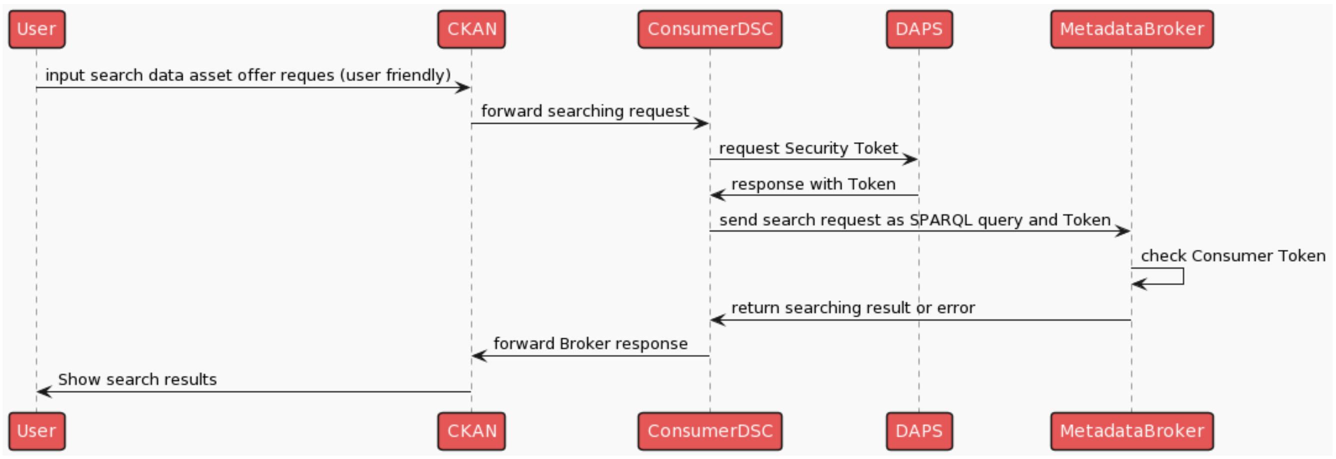


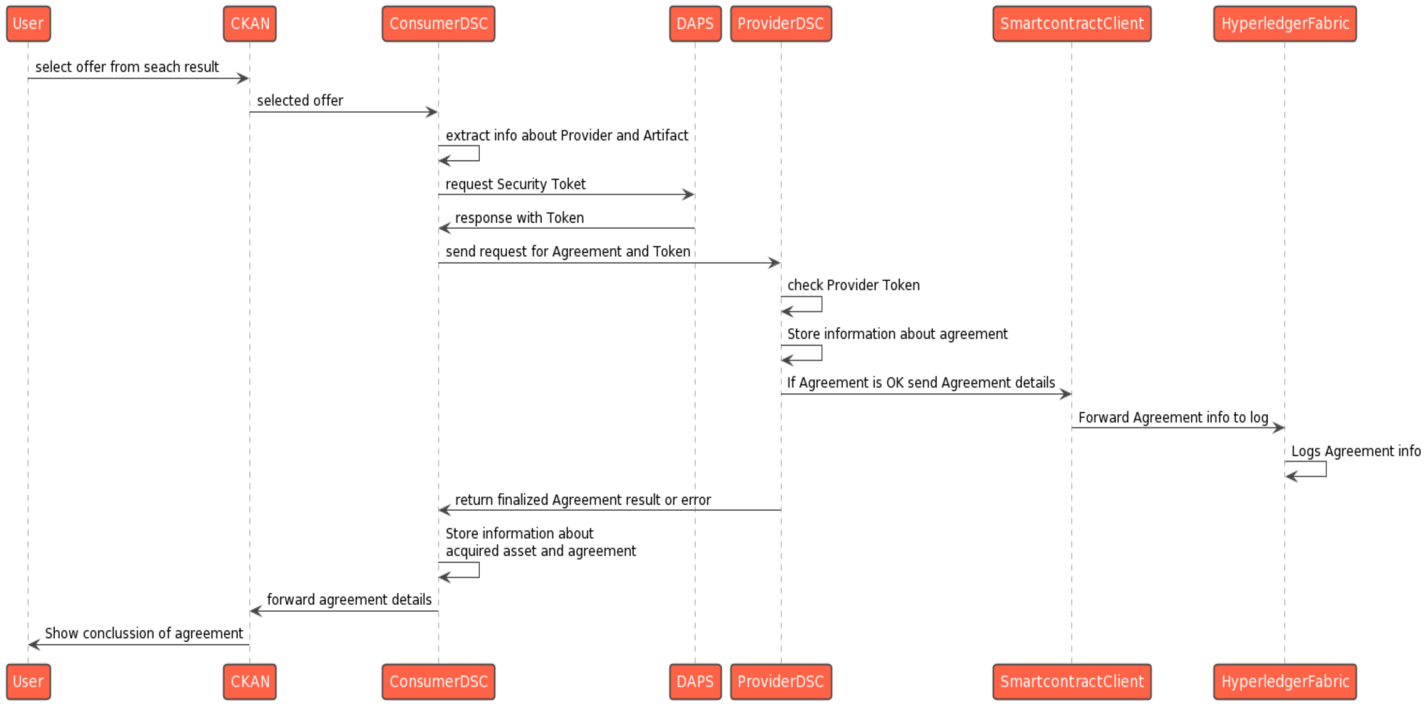
Figure 6: Data Asset Searching

Data Asset Agreement Conclusion

Data asset agreement conclusion can be fulfilled after searching in Metadata Broker.

1. The User in the Consumer node should select from the searching result suitable asset offer and press on the link for corresponding offer. CKAN shows detailed information about offer and the User to confirm the agreement should press "Accept Contract".
2. CKAN forwards the selected offer to Consumer DSC.
3. The Consumer DSC extracts information about location of the data asset.
4. The Consumer DSC requests a token from DAPS.
5. DAPS sends token or error back to Consumer DSC
6. The Consumer DSC sends a request for agreement and Token to Provider DSC.
7. The Provider DSC checks the Provider token if it is correct then stores information about agreement.
8. The Provider DSC returns the finalised agreement or an error to the Consumer DSC.
9. If the Provider DSC accepts the Agreement, then it sends information about the Agreement to the Smart Contract Client.

10. The Smart Contract Client sends the Agreement information to Hyperledger Fabric Server. The HLF logs the Agreement information.
11. The Consumer DSC stores the agreement information.
12. The Consumer DSC forwards the agreement details to CKAN.
13. CKAN shows the agreement details in the User's screen.


User
CKAN
ConsumerDSC
DAPS
ProviderDSC
SmartcontractClient
HyperledgerFabric
select offer from seach result
selected offer
extract info about Provider and Artifact
request Security Toket
response with Token
send request for Agreement and Token
check Provider Token
Store information about agreement
If Agreement is OK send Agreement details
Forward Agreement info to log
Logs Agreement info
return finalized Agreement result or error
Store information about acquired asset and agreement
forward agreement details
Show conclussion of agreement
Figure 7: Data Asset Agreement Conclusion

Data Asset (Dataset) Access

Users of a Consumer node can download a dataset from the provider only if previously the Consumer has created an agreement for access to this dataset with the provider.

1. The User of the Consumer node selects from the list of agreements (contracted data assets) the dataset for downloading and presses the download button.
2. The Consumer CKAN forwards the information about the selected agreement (dataset) to the Consumer DSC.
3. The Consumer DSC extracts the information about the Provider for dataset to route request.
4. The Consumer DSC requests a token from DAPS.
5. DAPS responses with a token or an error.
6. The Consumer DSC sends a request for the dataset accompanied with its token to the provider.
7. The Provider DSC checks the consumer token.
8. The Provider DSC checks access rights for the consumer to the resource.
9. If 6 and 7 succeed, the Provider DSC requests the dataset from the Provider CKAN.
10. The Provider CKAN returns the dataset to the Provider DSC.
11. The Provider DSC returns the dataset to the Consumer DSC.
12. The Consumer DSC forwards the dataset to the Consumer CKAN.

13. The Consumer CKAN downloads the dataset and shows it in the User browser.

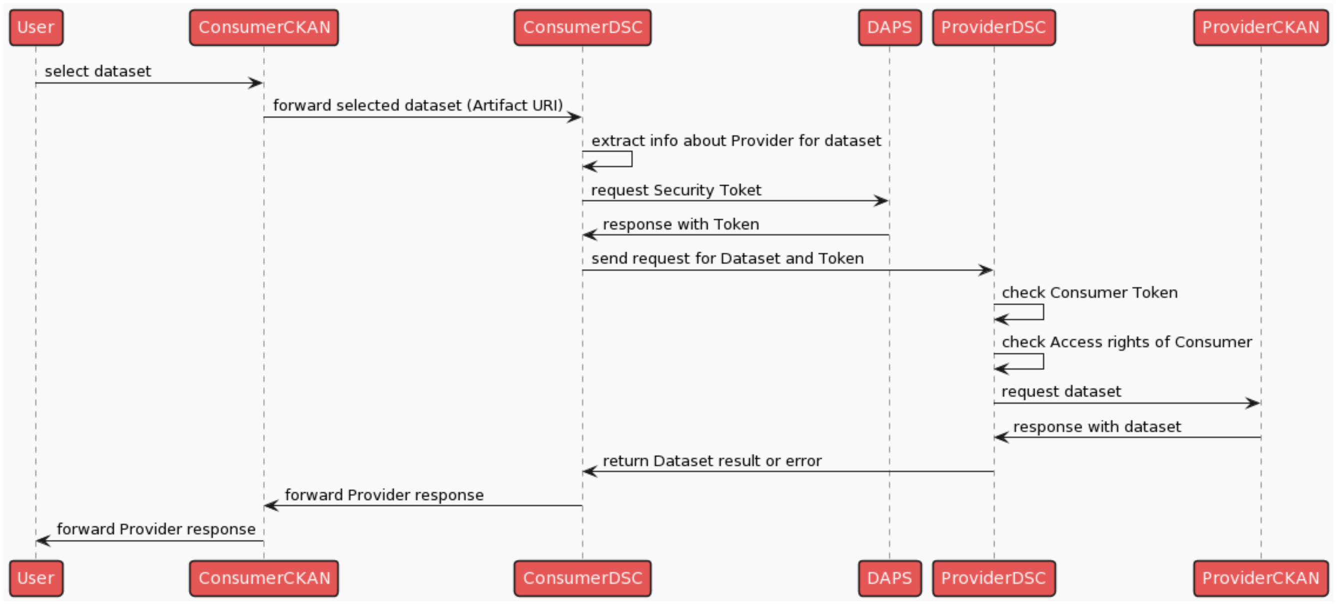

User
ConsumerCKAN
ConsumerDSC
DAPS
ProviderDSC
ProviderCKAN
select dataset
forward selected dataset (Artifact URI)
extract info about Provider for dataset
request Security Toket
response with Token
send request for Dataset and Token
check Consumer Token
check Access rights of Consumer
request dataset
response with dataset
return Dataset result or error
forward Provider response
forward Provider response

Figure 8: Data Asset (Dataset) Access

Data Asset (Service) Access

Users of a Consumer node can set up an application client to the provider resources only if previously the Consumer has created an agreement for access to these resources with the provider.

1. The User of the Consumer node selects from the list of agreements (contracted data assets) the services to access, then the user downloads from the Consumer CKAN all the needed information for setting up an app client to access the provider services.
2. The Consumer CKAN responses with setting info for the app client to the user.
3. The User configures the app client and runs it.
4. The app client issues an action request for the application server through the Consumer DSC.
5. The Consumer DSC extracts the information about the Provider of the services.
6. The Consumer DSC requests a token from DAPS.
7. DAPS responses with a token or an error.
8. The Consumer DSC sends a request to the service accompanied with its token to the provider.
9. The Provider DSC checks the consumer token.
10. The Provider DSC checks the access rights for the consumer to the resource.
11. If 6 and 7 succeed, the Provider DSC forwards the request to the AppServer.
12. The AppServer response with data to the Provider DSC.
13. The Provider DSC returns the data result or an error to the Consumer DSC.
14. The Consumer DSC forwards the provider response to the AppClient.

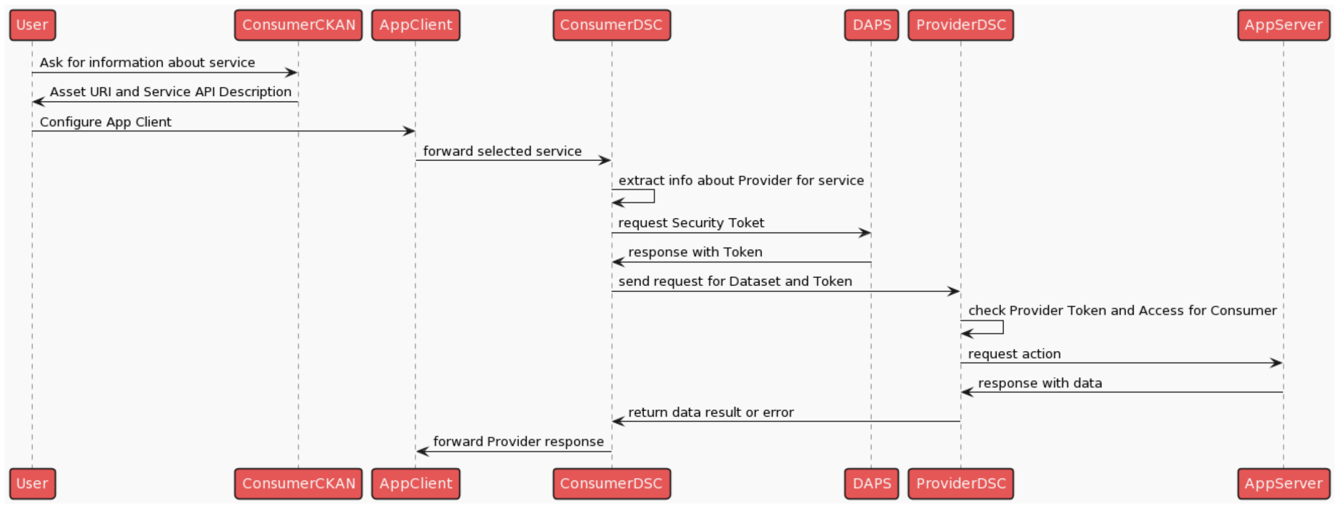

User
ConsumerCKAN
AppClient
ConsumerDSC
DAPS
ProviderDSC
AppServer
Ask for information about service
Asset URI and Service API Description
Configure App Client
forward selected service
extract info about Provider for service
request Security Toket
response with Token
send request for Dataset and Token
check Provider Token and Access for Consumer
request action
response with data
return data result or error
forward Provider response

Figure 9: Data Asset (Service) Access

Data Asset (Application) Access

Users of a Consumer node can set up and run an application in a Consumer node only if the Consumer has previously created an agreement for access to these resources with the provider.

1. The User of the Consumer node selects from the list of agreements (contracted data assets) the application to access, then the user downloads from Consumer CKAN all needed information for setting up an app client to access provider services.
2. The Consumer CKAN forwards a request for the selected application to the Consumer DSC.
3. The Consumer DSC extracts the information about the Provider of the services.
4. The Consumer DSC requests a token from DAPS.
5. DAPS responses with token or error.
6. The Consumer DSC sends a request for the application accompanied with its token to the provider.
7. The Provider DSC checks the consumer token.
8. THe Provider DSC checks the access rights for the consumer to the resource.
9. If 6 and 7 succeed, the Provider DSC responds with an application setup information to the Consumer DSC.
10. The Consumer DSC forwards the Provider response to the Consumer CKAN.
11. The Consumer CKAN shows the User application setup information to the User.
12. The User sends a request to download the application from the TRUSTS Docker images repository.
13. The Docker Images Repository sends the docker image of the application to the User.
14. The User sets up the application.
15. The User runs the application in the Consumer Node.

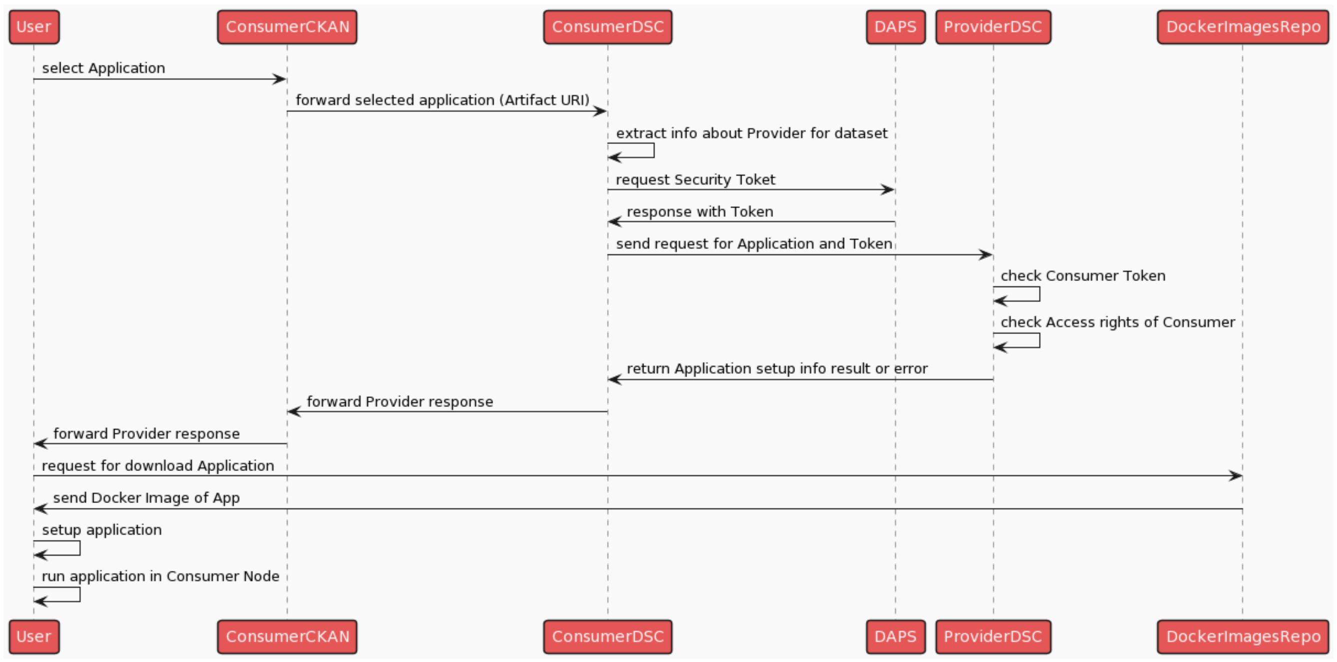

User
ConsumerCKAN
ConsumerDSC
DAPS
ProviderDSC
DockerImagesRepo
select Application
forward selected application (Artifact URI)
extract info about Provider for dataset
request Security Toket
response with Token
send request for Application and Token
check Consumer Token
check Access rights of Consumer
return Application setup info result or error
forward Provider response
forward Provider response
request for download Application
send Docker Image of App
setup application
run application in Consumer Node

Figure 10: Data Asset (Application) Access

Subscription to data asset update

When the User in the Consumer node accepts the contract, this action triggers:

1. After signing the contract, the User of the Consumer node presses the button for subscription to the data asset updates.
2. The Consumer CKAN sends a request for IDS subscription to the Consumer DSC
3. The Consumer DSC sends a subscription message to the provider of the artefact.
4. The Consumer CKAN sends a request for non-IDS subscription to the Consumer DSC.

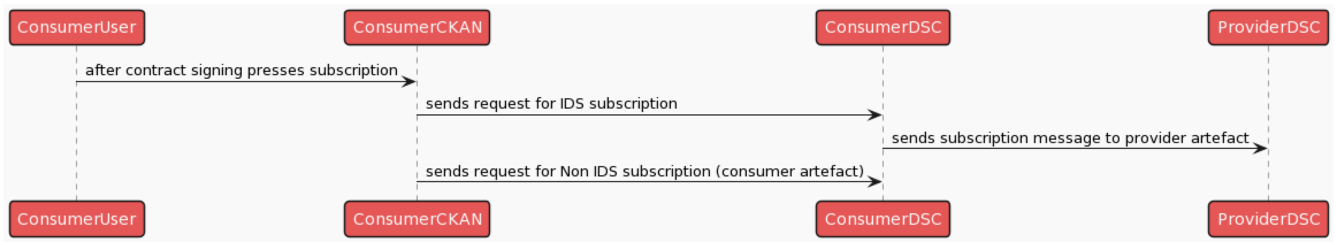

ConsumerUser
ConsumerCKAN
ConsumerDSC
ProviderDSC
after contract signing presses subscription
sends request for IDS subscription
sends subscription message to provider artefact
sends request for Non IDS subscription (consumer artefact)

Figure 11: Subscription to data asset update

User notification on asset update

This is a description of how the Consumer User will be notified when the contracted asset was updated.

1. The User of the Provider node updates the contracted artefact.
2. The Provider CKAN posts an update request for the corresponding artefact into the Provider DSC.
3. The Provider DSC sends to the Consumer DSC an update message if IDS subscription exists for the artefact.
4. The Consumer DSC updates the local information about the corresponding artefact.

5. The Consumer DSC posts to the local noteService a request for the email if non-IDS subscription for the artefact exists.
6. NoteService sends email through the email service to the Consumer User who signed to the artefact subscription.

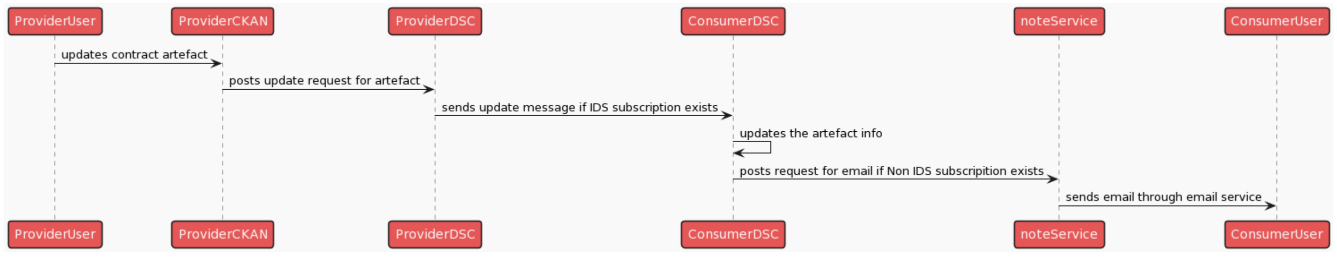

ProviderUser
ProviderCKAN
ProviderDSC
ConsumerDSC
noteService
ConsumerUser
updates contract artefact
posts update request for artefact
sends update message if IDS subscription exists
updates the artefact info
posts request for email if Non IDS subscripition exists
sends email through email service

Figure 12: User notification on asset update

4.2.3.2 Interoperability between TRUSTS and other external data markets/EOSC

A software module was created, the TRUSTS platform client, that allows programmatic batch-upload of metadata of datasets into TRUSTS, and itself accesses the REST-API that is provided by CKAN, the data management platform used as the basis for TRUSTS. Figure 13 shows the ETL process we use to load metadata from the two EOSC initiatives OpenAIRE and Europeana. First metadata was acquired from them (the "extract" part of the ETL process), then it was transformed into the TRUSTS-IM (the "transform" part) and subsequently serialised into the data storages of TRUSTS (the "load" part). The last step leverages the before-mentioned TRUSTS platform client.

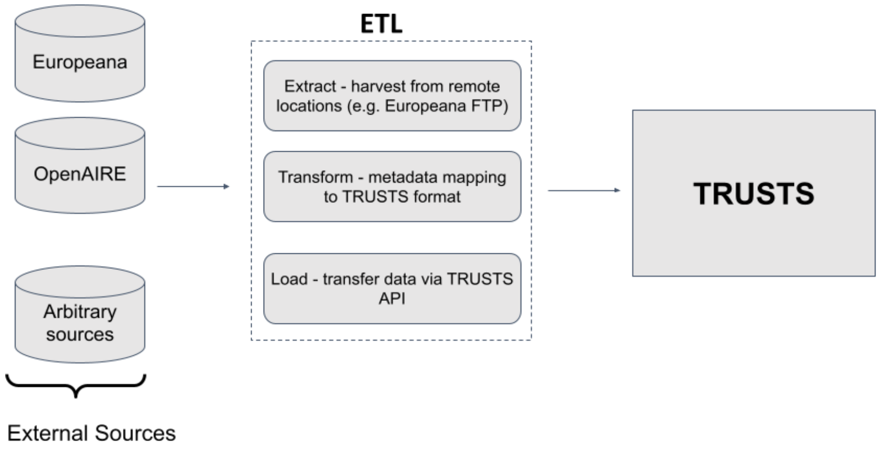

Europeana
OpenAIRE
Arbitrary sources
External Sources
ETL
Extract - harvest from remote locations (e.g. Europeana FTP)
Transform - metadata mapping to TRUSTS format
Load - transfer data via TRUSTS API
TRUSTS

Figure 13: The ETL process used by the two EOSC connectors

4.2.3.3 Smart Contracts functionalities

Asset Purchase

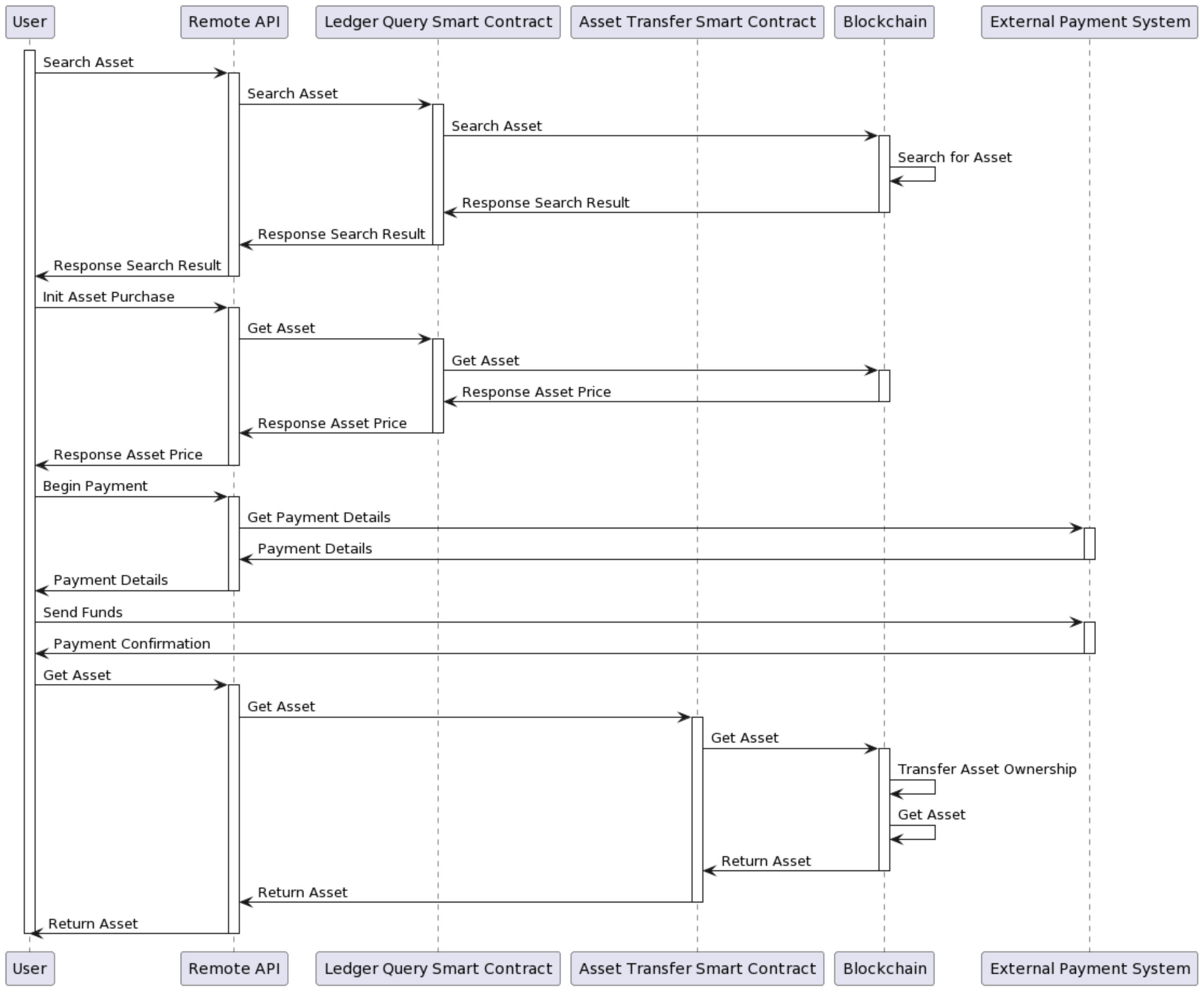


Figure 14: Asset Purchase

In Figure 14, a process by which a user purchases an asset is outlined. The user begins by searching the blockchain for a desired asset using a ledger query smart contract and receives a response. The user then initialises the asset purchase operation and gets the price of the desired asset. Payment details for the seller or provider of the asset are retrieved next and following this the user makes a payment to them. Finally, with the payment complete the user gets ownership of their newly purchased asset (or instance of it) committed to the blockchain. This concludes the asset transfer operation example shown in Figure 14.

Asset Rating and Reviewing

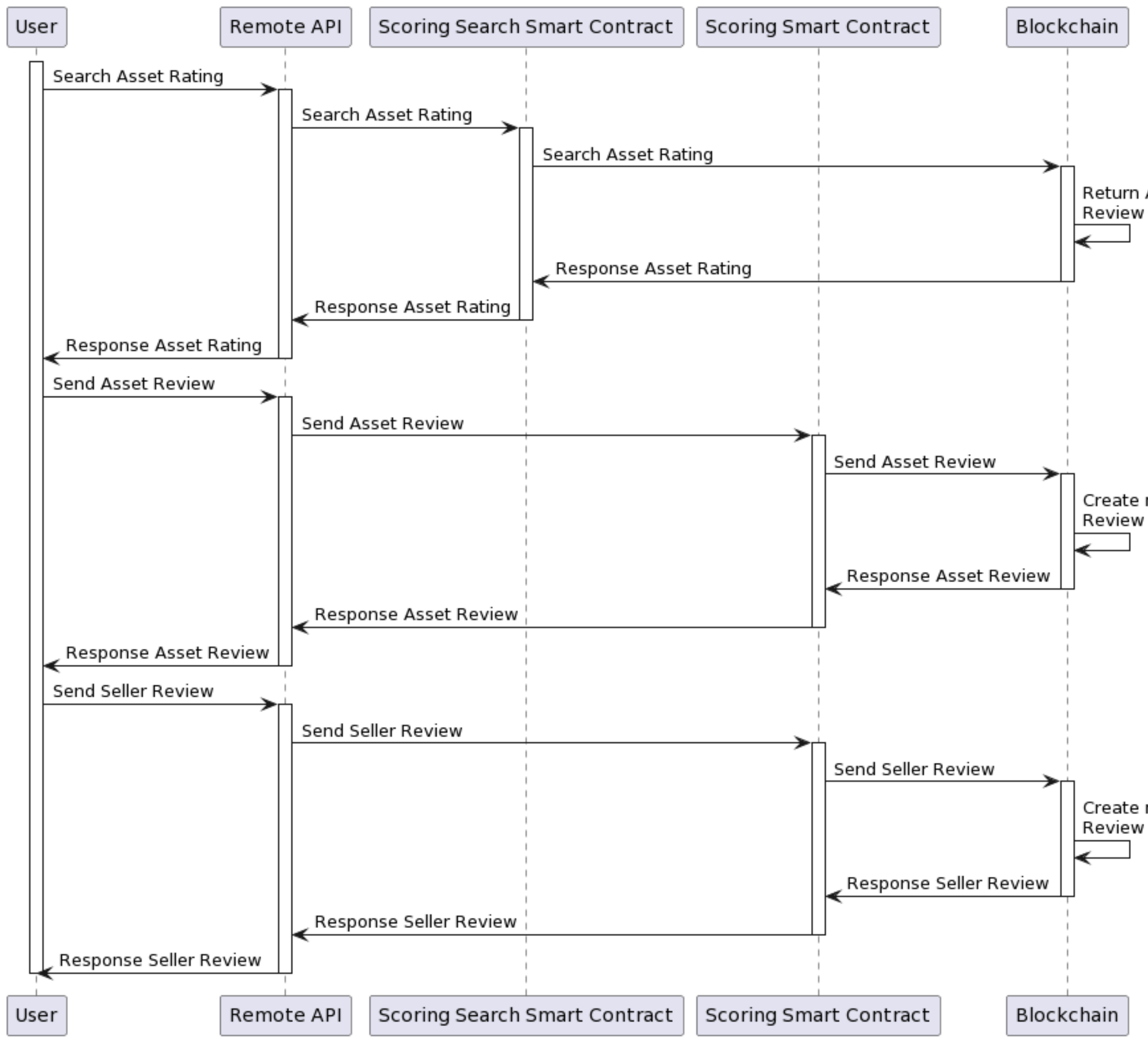


Figure 15: Asset Rating and Reviewing

In Figure 15 the process of viewing and making asset reviews and ratings, and seller reviews and ratings. The first step outlined in Figure 15 is the user searching for ratings of a certain asset. These ratings objects can also provide text (i.e., a review) in a description field. The blockchain responds to the request with rating objects relating to the searched asset ID. Next the user creates their own asset review object using the user scoring smart contract. The asset review request is committed to the blockchain with a reference to the target asset being reviewed. The user is then notified of the operation's success. The user then creates a review object of a seller using the same user scoring smart contract. The request is placed, and the information is committed to the blockchain using the same process, but this time the reference points to a seller or data provider being rated. The user receives a confirmation of the successful process, ending the operation.

4.2.3.4 Recommender

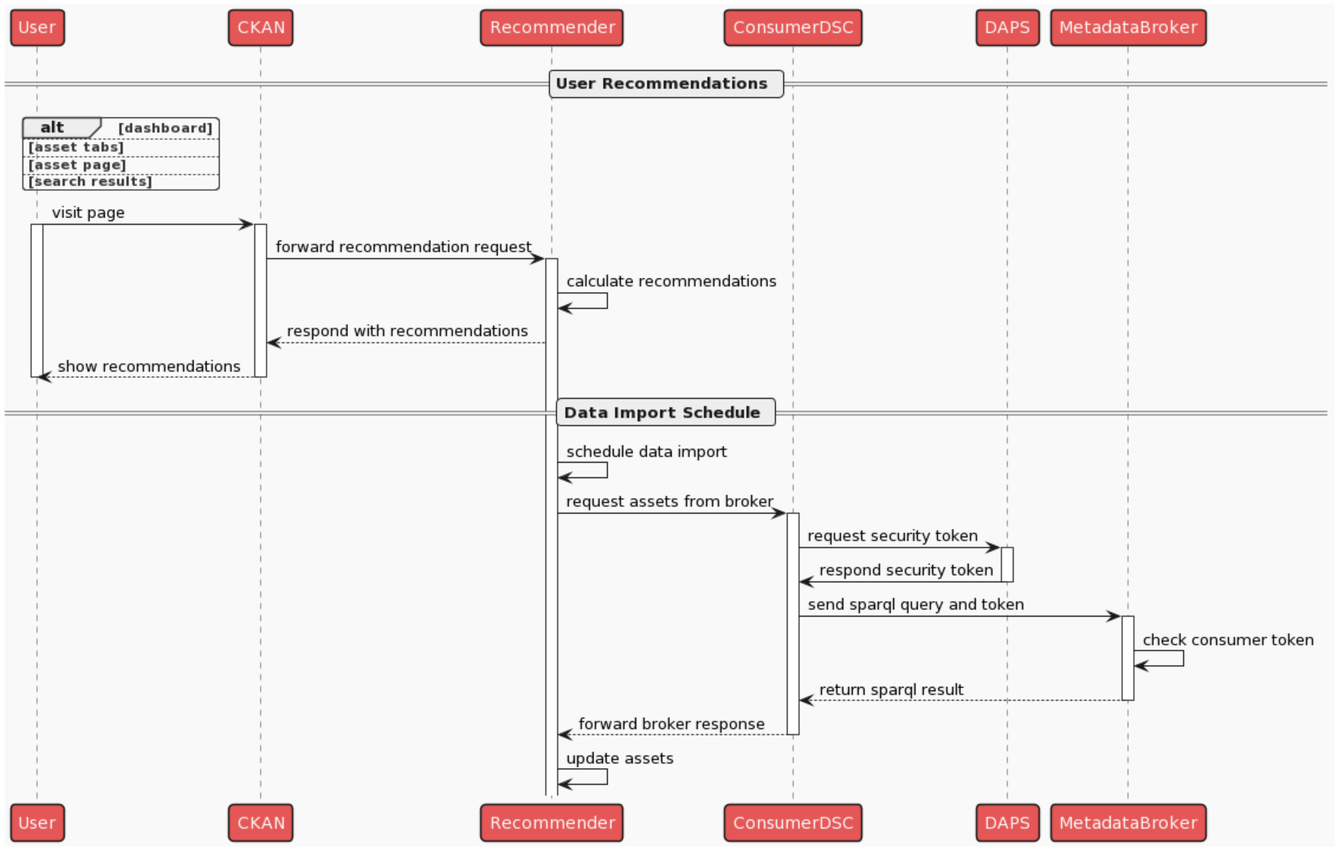


Figure 16: Recommender Workflow

User Recommendations

1. The user visits either the dashboard (landing page), one of the assets tabs, a particular asset's page or gets shown the search result page in the CKAN frontend.
2. CKAN backend subsequently creates respective recommendation request for one or more of the 6 use-cases (i.e., 1: Datasets to Users, 2: Services to Users, 3: Datasets to Services, 4: Services to Datasets, 5: Datasets to Datasets, 6: Services to Services) depending on the context of the asset and user and forwards the request to the Recommender.
3. The Recommender calculates the recommendations based on the actual endpoint instantiated depending on the use-case to be applied for the user and asset in context.
4. The recommendations are received by the CKAN backend.
5. The recommendations are shown to the user in dedicated locations within the actual page in the CKAN frontend.

Data Import Schedule

1. The Recommender instantiates the data import schedule, which is responsible for adding new assets available from the Metadata Broker, updating them accordingly and removing deleted assets.
2. The Recommender calls the ConsumerDSC with a created SPARQL query for fetching assets of certain types.
3. The ConsumerDSC requests a security token from DAPS.
4. DAPS sends back a token or error if it doesn't authenticate the ConsumerDSC.

5. The ConsumerDSC creates the request with the SPARQL query, adds its token and sends the request to the MetadataBroker.
6. The MetadataBroker checks the token and executes the SPARQL query if correct.
7. The result of the SPARQL query executed by the MetadataBroker is returned to the ConsumerDSC.
8. The ConsumerDSC forwards the response to the Recommender.
9. The Recommender updates its assets with the new information from the MetadataBroker, adds new assets if not existing yet and removes assets not being available from the MetadataBroker anymore.

4.2.3.5 Vocabulary Updates

The Vocabulary Extension developed for the CKAN platform allows for a centralised management of vocabularies through the Vocabulary Service, and the corresponding update to all the nodes in the platform. Prerequisites for this update system to work are detailed in Deliverable 3.8 and, briefly, consist of creating an asset in the Central Node's DSC for each vocabulary, creating an IDS subscription to this asset in the DSC of each node, and then a non-IDS subscription between each Platform Interface and its corresponding DSC. Once these requirements are met, the following sequence of actions are triggered when an update to the vocabularies occurs, which results in platform-wide propagation of this update. Afterwards, users in any of the platform nodes can use the new versions of vocabularies for faceted search of assets, as well as for onboarding new assets.

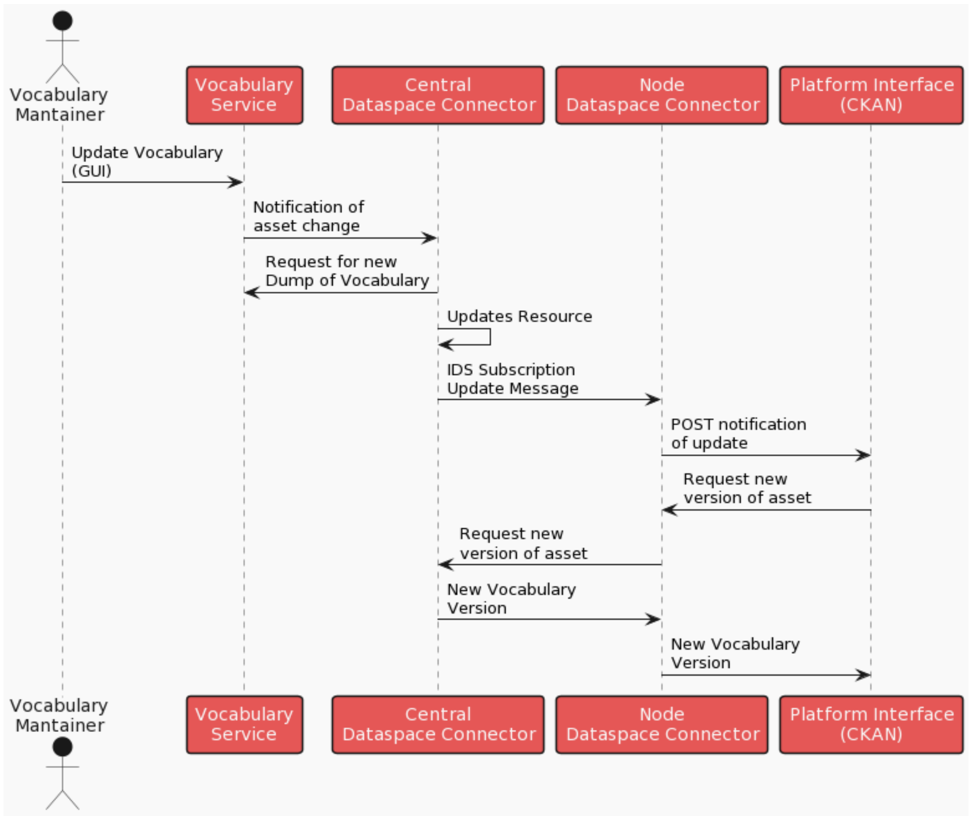

Vocabulary Mantainer
Vocabulary Service
Central Dataspace Connector
Node Dataspace Connector
Platform Interface (CKAN)
Update Vocabulary (GUI)
Notification of asset change
Request for new Dump of Vocabulary
Updates Resource
IDS Subscription Update Message
POST notification of update
Request new version of asset
Request new version of asset
New Vocabulary Version
New Vocabulary Version

Figure 17: Platform-wide propagation of vocabulary updates.

4.3 TRUSTS Remarkable Features as a Trusted Data Marketplace

Between the existing data marketplaces, the TRUSTS platform, it can be trusted. In the following, the add-value features and characteristics are presented.

- **Privacy-aware and sovereign platform for data sharing and exchanging.**
  TRUSTS should be trusted among the available data markets since it was built based on the "Data Protection by Design" method and offers powerful encryption methods, and anonymization and de-anonymization tools, which will be made available to TRUSTS users as a downloadable data service. Additionally, it enables control features with usage policies and records them in the smart contracts.
- **Compliance with data protection law.**
  The TRUSTS architecture was developed considering EU data protection principles (data minimisation, purpose limitation, and data security) and rules. Its most up-to-date combination of federated machine learning and advanced cryptography techniques creates a suitable environment for exchanging data in compliance with data protection law.
- **Semantic layer usage for its metadata.**
  TRUSTS's metadata, or information model, was built using semantic web technology based on IDS information model. Its content is stored in knowledge graphs. Also, it gives the power of using one standard query language to search in all the federated data market nodes.
- **Federated data markets.**
  TRUSTS makes use of DSC integration as the core component in each TRUSTS central and corporate node. At the same time, each node can be considered a separate functional node and independent regarding its internal work, data storage, and usage rules and policies. Therefore, data stays on the owner's premises while allowing advanced analytics over several parties' data.
- **Interoperability with the third party and EOSC.**
  Using again the semantic layer, it provides an add-value to the TRUSTS to allow the digital asset interoperability with third party and EOSC. This was achieved by data harvesting from the EOSC (e.g., OpenAIRE and Europeana).
- **Provision of various digital assets for sharing.**
  TRUSTS is not restricted to share or exchange only datasets, but it is extended to provide further accessibility to the data by both offered services and data applications where the latter allow to process or access the data using their corresponding applications, meanwhile the former enables RESTful web services for data access.
- **Smart contracting.**
  Blockchain technology in the Smart Contract component empowers the TRUSTS platform with the well-known benefits of blockchain regarding the recording of contracts between the stakeholder and business TRUSTS partners.
- **User-friendly UIs.**
  TRUSTS's UIs were improved while taking in consideration to have the systems as business-wise rational.
- **Tested with three different real-life use-cases.**
  To have a proof of concept for the developed TRUSTS platform and its services and components, the platform was introduced and tested in three real-life use-cases, provided, and executed by WP5.

5 Implementation

Chapter 5 discusses the implementation process and details of the TRUSTS platform. Section 5.1 is about the evolution of the platform using agile methods. Then, the recent changes to the components of the TRUSTS platform and how they are going to address the latest version of the functional requirements are demonstrated. In addition, the implementation status of those components is given. Sources for reference are provided for some of these components. This is followed by an experiment demonstrating the interaction of the TRUSTS platform with external operators.

5.1 Evolution of the platform with agile methods

As reported in D3.9 and D3.10, the use of agile methods led to continuous improvement of the platform. The concept of a “Minimum Viable Product (MVP)” was used to involve relevant stakeholders in the evolution of the platform, at different points in time during the project. An MVP [3, 4] can be thought of as a version of a product with just enough features to be usable by early customers, who can then provide feedback for future product development.

In TRUSTS, each MVP version was a clearly specified hand-off point, at which a specific combination of platform components in specific versions was released to all partners who were working on the implementation of use cases and on the implementation of privacy enhancing technologies. These partners were the internal customers of the MVPs within the TRUSTS project. Every MVP version was released internally within the project, together with a description of how to perform technical tests. The results of these technical tests were used as input for the next version of MVP. The goals for the MVPs were characterised by three dimensions: integration of APIs, functional integration, and operational integration. This has been detailed in D3.9.

5.2 Implementation status of components addressing the FRs

This section gives an overview of all components and shows which functional requirements are or can be met by which components. FRs are not necessarily met by only one component. In addition, the change in the component list is highlighted. As an overview, the following components are included in the end-of-project version of the TRUSTS platform:

Table 15: Component sources and references

Name of the Component	Source	References for (technical) details
IDSA Dataspace Connector	Reused from IDS	https://github.com/International-Data-Spaces-Association/DataspaceConnector
Reverse Proxy	Reused from open-source software	Any reverse proxy can be used.
Recommender System	Developed by KNOW for the TRUSTS	D3.12 [7] and D3.13 for details.

	project	
Platform Interface (CKAN)	Reused from open-source software, CKAN	https://gitlab.com/trusts-platform/ckan/-/tree/trusts
Landing page	Developed by FORTH for the TRUSTS project	TRUSTS Platform / ckan-trusts-theme · GitLab
Notification Service	Developed by REL for the TRUSTS project	https://gitlab.com/trusts-platform/note-service
Metadata Mapping	Developed by RSA for the TRUSTS project	D3.6 for details.
Corporate Interface (CKAN)	Reused from open-source software, CKAN	https://gitlab.com/trusts-platform/ckan/-/tree/trusts
Interoperability Component	Developed by RSA for the TRUSTS project	D3.6 for details.
Business Support Services	Developed by LST for the TRUSTS project	D1.4 for details.
Broker and Metadata Storage	Reused from IDS	https://github.com/International-Data-Spaces-Association/metadata-broker-open-core https://jena.apache.org/documentation/fuseki2/
Vocabulary Service	Developed by SWC, deployed in the TRUSTS project	https://www.poolparty.biz/
Dynamic Attribute Provisioning Service (DAPS)	Reused from IDS	https://github.com/International-Data-Spaces-Association/omejdn-daps
Smart Contract Executor	Developed by EMC for the TRUSTS project	D3.3 and D3.6 for details.

The list of components has changed slightly compared to the component list given in D3.10 and D2.7 [8] due to limited resources and the nature of agile development. The following four components are no longer part of the platform architecture:

1. Dataflow Router: The functionality is covered by the DSC.
2. App Store: There is a workaround for the component's functionality by having a docker registry to store the image and an application server to offer backend services for managing the data applications.

3. Automated Certificate Environment (ACME): The tasks of the ACME are currently performed by the DAPS component. Accordingly, the non-existence of this component has not been a stopper for project development.
4. Mapping Builder: For current usages, the TRUSTS platform is also scalable without a Metadata Builder.

In addition, modifications to the interoperability component were required for the following reasons: The initial architecture of the interoperability solution was adapted to address new requirements and newly available technologies in accordance with the spirit of agile development. For instance, the early architecture, as presented in Figure 12 of D3.4 [9], foresaw the use of the IDS Trusted Connector. However, the Trusted Connector evolved into the DSC, which is also used in the latest version of the TRUSTS platform. Consequently, metadata exchange for interoperability is also routed through this connector. Additionally, the early overall architecture foresaw a “Registry of Data Markets," a “Data Exchange TRUSTS component," and a “Data Exchange Client” component (see Figure 7 of D2.6 [10]). The registry of data markets still exists [11] and remains a repository for information about data markets. However, it is not used as an “address book,” providing URLs for interoperating with external data markets, as it proved too difficult to get them on board. Instead, the concept was shifted to the “TRUSTS platform client” and the EOSC connectors, which both replace the before-mentioned “Data Exchange TRUSTS component” and “Data Exchange client component." The newly designed architecture provides more flexibility in dealing with yet unknown data markets willing to interoperate with TRUSTS.

The FRs can be addressed by individual components or by a combination of components, as shown in the table below. All high priority FRs have been implemented. Likewise, most of the lower-priority ones—seven FRs—could not be implemented due to resource limitations.

Table 16: FRs addressed components and implementation status

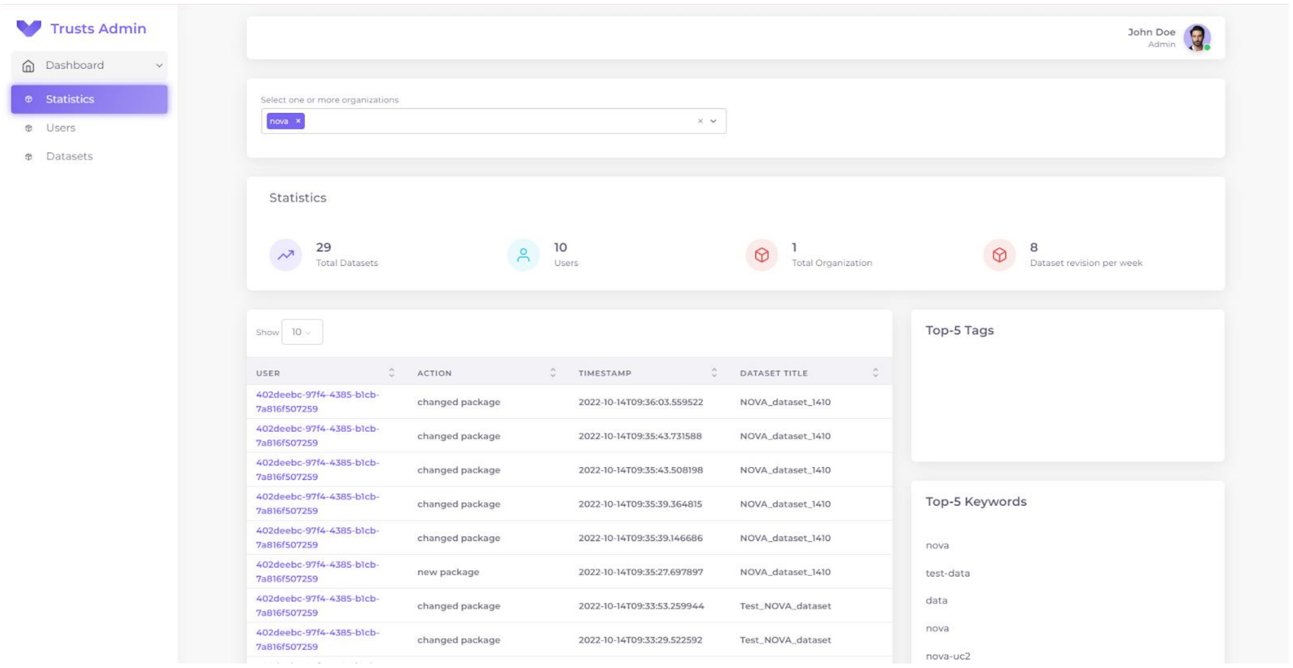
FRs	Components	Implementation status
FR 1	IDS Dataspace Connector, Platform Interface, Corporate Interface, Broker and Metadata Storage	full
FR 2	Interoperability Components	full
FR 3	Platform Interface, Corporate Interface, Broker, and Metadata Storage	full
FR 4	Platform Interface, Corporate Interface, Interoperability Components	full
FR 5	Platform Interface, Corporate Interface, Broker, and Metadata Storage	full
FR 6	Recommender System	full
FR 7	Recommender System	full
FR 8	Recommender System	full
FR 9	Recommender System	full

FR 10	Smart Contract Executor	baseline
FR 11	Smart Contract Executor	baseline
FR 12	Smart Contract Executor	not
FR 13	Smart Contract Executor	not
FR 14	Smart Contract Executor	full
FR 15	Payment Compatibility Demonstrator	baseline
FR 16	Smart Contract Executor	not
FR 17	Smart Contract Executor	full
FR 18	Broker and Metadata Storage	full
FR 19	Vocabulary Service	full
FR 20	Vocabulary Service	full
FR 21	Broker and Metadata Storage, Vocabulary Service	full
FR 22	Broker and Metadata Storage, Vocabulary Service	baseline
FR 23	Metadata Mapper, Interoperability Component, Broker and Metadata Storage	full
FR 24	Vocabulary Service	full
FR 25	Recommender System, Platform Interface, Broker and Metadata Storage, Vocabulary Service	full
FR 26	Broker and Metadata Storage	baseline
FR 27	IDS Dataspace Connector, Platform Interface, Corporate Interface	baseline
FR 28	IDS Dataspace Connector	baseline
FR 29	Platform Interface, Corporate Interface	full
FR 30	Platform Interface, Corporate Interface, Smart Contracts Executor	baseline
FR 31	IDS Dataspace Connector, Platform Interface, Corporate Interface	not
FR 32	Platform Interface, Corporate Interface	full
FR 33A	–	not
FR 33B	–	not
FR 34	Platform Interface, Corporate Interface	baseline

FR 35	Notification Service	full
FR 36	Platform Interface, Corporate Interface	full
FR 37	Platform Interface, Corporate Interface	full
NFR 1	Platform Interface, Corporate Interface	not
NFR 2	Platform Interface, Corporate Interface	baseline
NFR 3	IDS Dataspace Connector, Platform Interface, Corporate Interface	baseline
NFR 4	Platform Interface, Corporate Interface	not
NFR 5	Landing Page, Platform Interface, Corporate Interface	baseline
NFR 6	IDS Dataspace Connector, Reverse Proxy, Notification Service	full

5.2.1 Business Support Service

Under the scope of T5.2 and in collaboration with NOVA, LSTech developed an admin-dashboard for the TRUSTS platform, called Business Support Services, which is a basic but essential tool for monitoring the usage of the platform. Through an intuitive dashboard, information related to registered organisations, users, and data sets is available.



Statistics

29 Total Datasets
10 Users
1 Total Organization
8 Dataset revision per week

USER	ACTION	TIMESTAMP	DATASET TITLE
402deebc-97f4-4385-b1cb-7a816f507259	changed package	2022-10-14T09:36:03.559522	NOVA_dataset_1410
402deebc-97f4-4385-b1cb-7a816f507259	changed package	2022-10-14T09:35:43.731588	NOVA_dataset_1410
402deebc-97f4-4385-b1cb-7a816f507259	changed package	2022-10-14T09:35:43.508198	NOVA_dataset_1410
402deebc-97f4-4385-b1cb-7a816f507259	changed package	2022-10-14T09:35:39.364815	NOVA_dataset_1410
402deebc-97f4-4385-b1cb-7a816f507259	changed package	2022-10-14T09:35:39.146686	NOVA_dataset_1410
402deebc-97f4-4385-b1cb-7a816f507259	new package	2022-10-14T09:35:27.697897	NOVA_dataset_1410
402deebc-97f4-4385-b1cb-7a816f507259	changed package	2022-10-14T09:33:53.259944	Test_NOVA_dataset
402deebc-97f4-4385-b1cb-7a816f507259	changed package	2022-10-14T09:33:29.522592	Test_NOVA_dataset

Top-5 Keywords

nova
test-data
data
nova
nova-uc2

Figure 18: Screenshot of the TRUSTS admin dashboard

The main information that is provided is the following: datasets per organisation and their status, users per organisation; most popular datasets, and top dataset tags and keywords.

The tool is based on the CKAN Stats extension and is developed in angular/typescript, with the backend service in Python 3. The backend service has access to every corporate node using the necessary access tokens and provides a REST API.

5.3 Simulation of an external data market using TRUSTS Clone

An experiment was conducted simulating an exchange of metadata between an external operator and TRUSTS. The goal was to verify the operational functionality of the interoperability component. Therefore, we deployed another instance of the TRUSTS platform that was completely separate from the main TRUSTS platform. This additional, separate deployment served as the simulated external marketplace. This external marketplace was pre-loaded with several datasets. During the experiment, we selected one of the datasets and used the interoperability component to transfer its metadata into the TRUSTS format and subsequently load it into TRUSTS. Figure 19 shows an overview of this process. The simulation showed that operators of external data marketplaces can use the interoperability component to map their metadata catalogues into TRUSTS.

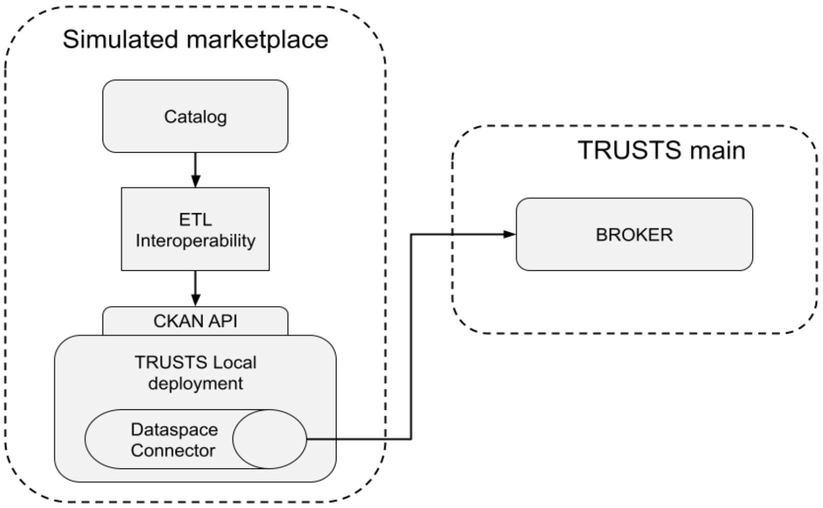


Figure 19: Overview of the Interoperability Experiment

5.4 Assessment of operator’s effort for featured components

The TRUSTS platform offers the functionalities that a data marketplace cannot do without selling, buying, secured communication, immutable transaction logging, a friendly user interface, and a recommendation system. Nevertheless, effort remains for a potential TRUSTS platform operator. In the following, the featured componenets are assessed regarding their limitations, Licencing, Updates, Security, and Market readiness.

Table 17: IDS Dataspace Connector Assessment

IDS Dataspace Connector	
Limitations & Improvements:	The DSC, in our current implementation, supports only http, https protocols and only the get method. It doesn't support put, post, delete, and other REST methods, which means that applications and services can only have only one way data flow from a provider to a consumer. It is possible to overcome this limitation through additional research on the DSC, especially on its features like camel-base communication via IDSCPv2. Additionally, it, like any open-source product, can be improved through the creation of additional functionality.
Licencing	This component is open source and has Apache-2.0 licence.
Updates & support	There is an active site supported by the product creators where periodically is available for downloading new versions of it. Also, some sort of support is available from this group of developers through the same site (see the reference given in Table 15).
Security	From a security point of view, two aspects should be considered: (1) security for the whole TRUSTS platform (communication between nodes) and (2) security inside any node. The communication security between nodes is at a good level because of the use of mutual TLS authentication based on centrally issued X.509v3 certificates. Communication is encrypted and goes through only one port. The security can be improved using reverse proxy. The security inside the node uses token authorization and can also be encrypted using the TLS security channel; it could be improved additionally through the deployment of a reverse proxy. Our recommendation is for any node deployment after building of TRUSTS infrastructure to carry out Penetration Test of the whole infrastructure of the node.
Market readiness	Mostly, the DSC has all its main features as a marketplace connector, and from that point of view, based on the description above and the history of developing it (the current version is 8.0.2), we can recommend it as ready for market.

Table 18: TRUSTS Notification service Assessment

TRUSTS Notification service	
Limitations & Improvements:	TRUSTS Notification Service is used only together with an SMTP email server and, in its current version, has no authentication mechanism. Like an open-source product, it can be modified and improved.
Licencing	This component is open source and has Apache-2.0 licence.
Updates & support	Any possible new version of the component is available for download from TRUSTS' official sites.
Security	Communication with the TRUSTS Notification Service goes through a secured and encrypted TLS channel, but in the current version, because it was created to support the DSC, there isn't any authentication mechanism. To overcome this

	restriction, additional efforts are required, and first the DSC needs changes to support an authentication process for the notification service.
Market readiness	This component is a supporting component by the TRUSTS platform and supposed to be used in only local secured infrastructure of the TRUSTS node and with very restricted functionality (sending a mail to inform users that the data asset of interest was changed). No access from the Notification Service to any other components of the platform exists.

Table 19: IDS DAPS Assessment

IDSA DAPS (omejdn-daps)	
Limitations & Improvements:	IDSA DAPS as a Certificate Authority has some limitations in automating the issuing and revoking of certificates. Some steps to implement this functionality have already been taken by the creators, but additional research and work are needed. As an open-source product, it can be done by any interested party.
Licencing	The component is open source and has Apache-2.0 licence.
Updates & support	Any possible new version of the component is available for download from the IDSA site.
Security	Communication with IDSA DAPS could be done through a TLS encrypted channel. IDSA supports the OAuth model and can restrict access to only registered parties that have valid certificates issued and recognized by corresponding IDSA DAPS servers.
Market readiness	This component is a rather like the one for use in data marketplaces.

Table 20: Platform Interface Assessment

Platform Interface (CKAN with custom extensions)	
Limitations & Improvements:	While the platform interface supports all of the flows specified in this project, there are missing flows to, for example, i) provide offers and counteroffers for an asset (contract negotiation is currently limited to either accepting or rejecting the provider's offerings), ii) automatically provision of infrastructure for launching TRUSTS-acquired applications, or iii) create different users with different access levels to the TRUSTS platform.
Licencing	The component is open source and has an AGLP licence.
Updates & support	The CKAN upstream package maintainers release a new version of the platform approximately every year. It is possible that major releases could break the functionality of some of the project-developed extensions.
Security	CKAN itself does not support TLS encryption, but it can be put behind a TLS-terminating reverse proxy. User management and multitenancy in CKAN are rather limited. For this reason, it is advised that the operator keep the corporate interfaces always confined to a corporate network (e.g., a VPC).

Market readiness	This component has thousands of installations worldwide, including the world's leading public data portals.

Table 21: Vocabulary Provider Assessment

Vocabulary Provider (PoolParty Semantic Suite)	
Limitations & Improvements:	It is used as a vocabulary manager in the TRUSTS platform, allowing for centralized control of vocabularies and their distribution to all nodes. This component is rather mature. The remaining improvements relevant for the TRUSTS platform's operations are related to its deployment on container management systems (e.g., Kubernetes). This is expected to be ready by mid-2023.
Licencing	commercial product. distributed by Semantic Web Company GmbH.
Updates & support	Roughly every nine months, a new release.
Security	The provider is an ISO 27001 certified software vendor. The component uses TLS encryption for communication, and allows for OAuth 2.0 based user authentication, including 2FA.
Market readiness	The component has been commercially available for several years now, with hundreds of installations worldwide.

Table 22: Recommender System Assessment

Recommender System	
Limitations & Improvements:	The TRUSTS recommender system implements three classic recommendation algorithms, collaborative filtering, and content-based filtering. One limitation of these algorithms is that they typically suffer from data sparsity issues.
Licencing	After the project ends, a licence agreement needs to be set up to use it. However, the TRUSTS algorithms and data ingestion components are, of course, open-source software.
Updates & support	For future improvements, recommendation algorithms based on matrix factorization or deep learning that address data sparsity issues could be employed. Additionally, we plan to adjust the hyperparameters of the recommendation algorithms based on the user feedback (i.e., clicks) on the recommendations.
Security	The recommender system itself does not provide TLS encryption, but it can be put behind a reverse-proxy providing secured access.
Market readiness	The TRUSTS recommender system is based on Know-Center’s ScaR recommender framework [12]. ScaR has a TRL of 6–7 and is proprietary software.

6 Conclusion

The TRUSTS platform status is described in this deliverable. All versions of the platform released in the final project year are described in more detail. After reading this report, the reader should be familiar with the step-by-step development of the TRUSTS platform over the past year. The components that form the building blocks of the platform architecture were introduced. The FRs addressed by the TRUSTS platform are listed, and implementation details are given. Furthermore, the component's effort for a potential TRUSTS operator is described. Therefore, the components were analysed based on the following key points: limitations, licence, updates, security, and market readiness.

The TRUSTS project demonstrated the complexity of building a federated and secure data marketplace. The components and functionalities supporting such a platform were deeply discussed. Open-source components were carefully evaluated. The benefits of reusing open-source software are also accompanied by disadvantages as soon as problems occur, and the software experts are not available at short notice. Nevertheless, several open-source components have been successfully integrated into the TRUSTS platform.

IDS components, namely the IDS Dataspace Connector, Metadata Broker, and IDSA DAPS, are essential parts of the TRUSTS platform. The TRUSTS-IM is based on the IDS information model. It is the main component of the data exchange in the communication infrastructure established in TRUSTS. But also new components were developed as part of the TRUSTS project. The WP3 development team gained a lot of experience in building, supporting, deploying, and improving data marketplaces.

All high-priority requirements are implemented in the end-of-project version of the TRUSTS platform, namely TRUSTS v.1.0. Providers can offer datasets, applications, and services. Consumers can search for data assets. Data assets can be acquired. Secure communication is provided. Interoperability with external data markets has been demonstrated. Users can rely on recommendations. Nevertheless, there is room for improvement. The ability of the TRUSTS platform to interoperate with external data marketplaces has been demonstrated but needs to be improved in terms of automation.

Even if it is sufficient for the project's use cases, the reliability and stability of the platform can be further improved. Further analysis of the platform's security and GDPR compliance is also required and using the platform for the first time requires technical knowledge. Accordingly, the user-friendliness needs improvement.

But it remains to emphasise that the platform was continuously improved in various aspects during the reporting period, so that the use case trials could be executed satisfactorily. The platform versions released in July 2022 and September 2022 have been tested intensively and successfully.

7 References

[1] M. Traub, et al., "Broker and Assessment Technology Specification and Development Road", Data Market Austria, May 2017.

[2] B. Otto, et al., "Reference Architecture Model Version 3.0", International Data Spaces Association, April 2019.

[3] V. Lenarduzzi, D. Taibi, "MVP Explained: A Systematic Mapping Study on the Definitions of Minimal Viable Product", Euromicro SEAA, 2016.

[4] J. Münch, et al. "Creating minimum viable products in industry-academia collaborations." International Conference on Lean Enterprise Software and Systems. Springer, Berlin, Heidelberg

[5] D2.3: Industry specific requirements analysis, definition of the vertical E2E data marketplace functionality and use cases definition II, https://www.trusts-data.eu/wp-content/uploads/2022/01/D2.3_Industry-specific-requirements-analysis-definition-of-the-vertical-E2E-data-marketplace-functionality-and-use-cases-definition-II_Dec2021.pdf, accessed Dec 14, 2022.

[6] CKAN Harvesting Extension, https://github.com/ckan/ckanext-harvest, accessed Dec 14, 2022.

[7] D2.7: Architecture design and technical specifications document II, https://www.trusts-data.eu/wp-content/uploads/2021/07/D3.12-Profiles-and-Brokerage.pdf, accessed Dec 14, 2022.

[8] D3.7: Profiles and Brokerage, https://www.trusts-data.eu/wp-content/uploads/2022/01/D2.7-Architecture-design-and-technical-specifications-document-II_Dec2021.pdf, accessed Dec 14, 2022.

[9] D3.4: Data Marketplaces with Interoperability Solutions I, https://www.trusts-data.eu/wp-content/uploads/2021/01/D3.4-Data-Marketplaces-with-Interoperability-Solutions-I.pdf, accessed Dec 20, 2022.

[10] D2.6: Architecture design and technical specifications document I, https://www.trusts-data.eu/wp-content/uploads/2021/09/D2.6_Architecture-design-and-technical-specifications-document-I.pdf, accessed Dec 20, 2022.

[11] Registry of Datamarkets, https://datamarkets.info/, accessed Dec 20, 2022.

[12] ScaR : Scalable Recommendation-as-a-service, http://scar.know-center.tugraz.at/, accessed Dec 20, 2022.

[13] Resubmitted D3.9: Platform Status Report I, https://www.trusts-data.eu/wp-content/uploads/2022/01/D3.9-Platform-Status-Report-I_Resubmission_Nov2021.pdf , accessed Dec 20, 2022.

[14] D3.10: Platform Status Report II, https://www.trusts-data.eu/wp-content/uploads/2022/01/D3.10-Platform-Status-Report-II_Dec2021.pdf, accessed Dec 20, 2022.