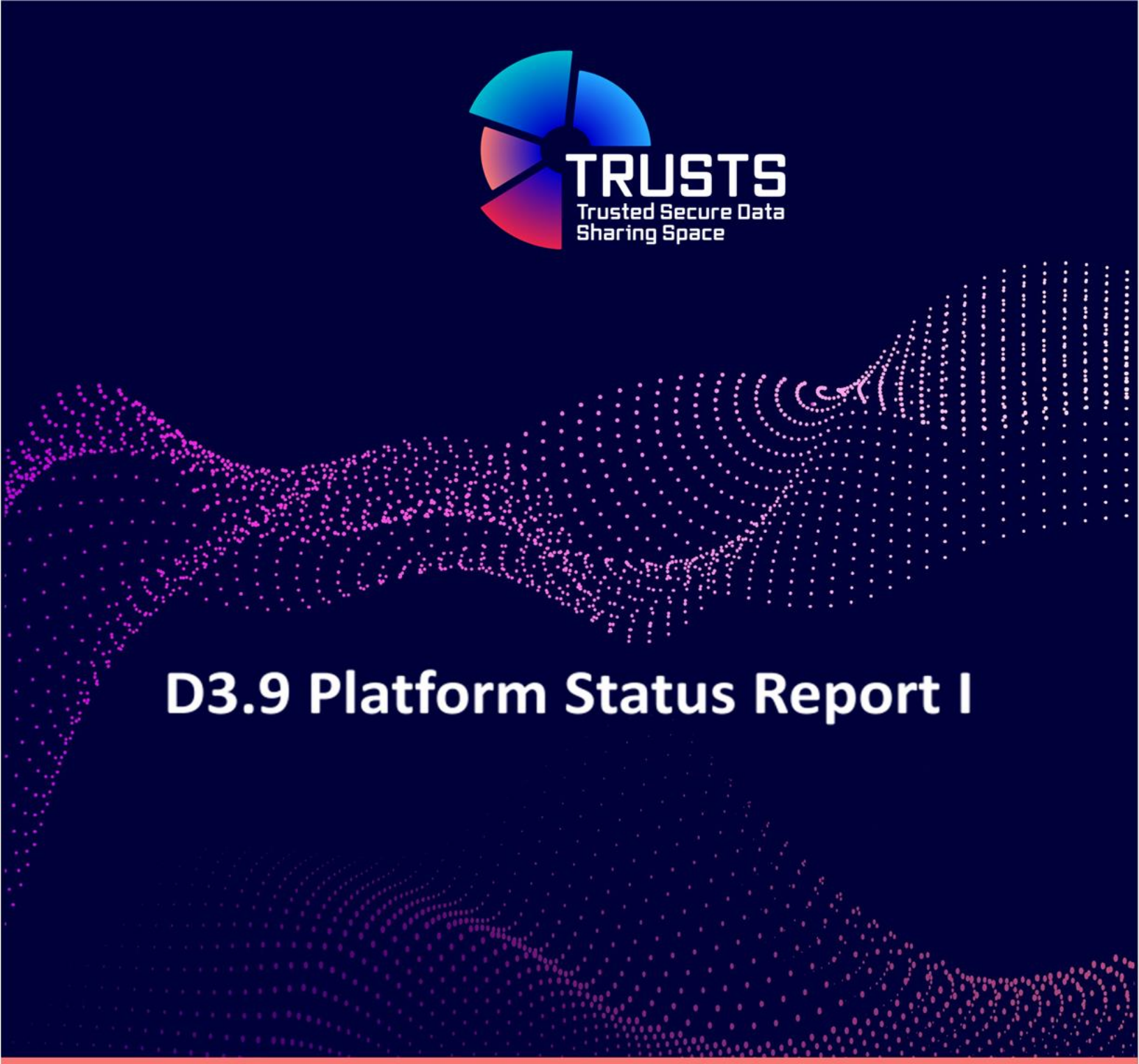

D3.9 Platform Status Report I

Authors: **Kim Fidomski, Benjamin Heitmann, Ahmad Hemid**
December 2020, resubmission in November 2021

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

TRUSTS Trusted Secure Data Sharing Space

D3.9 Platform Status Report I

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	D3.9 Platform Status Report I		
Work Package	WP3 - TRUSTS Platform implementation		
Contractual due date	31/12/2020	Actual submission date	15/11/2021 (resubmission)
Nature	Report	Dissemination Level	Public
Lead Beneficiary	Fraunhofer (FhG)		
Responsible Author	Benjamin Heitmann		
Contributions from	FhG, SWC, EMC, G1, NOVA, EBOS, LST, REL, FORTH, KNOW, RSA		

© TRUSTS, 2021

Revision history (including peer reviewing & quality control)

Version	Issue Date	% Complete	Changes	Contributor(s)
v0.1	01/10/2020	5	Initial Deliverable Structure	Ana-Maria Florea (REL)
v1.0	10/05/2021	95	Deliverable ready for peer review	Ana-Maria Florea (REL)
v1.2	27/05/2021	100	Revisions according to peer review	Benjamin Heitmann (FhG)
v1.3	02/06/2021	100	Final version for first submission	Benjamin Heitmann (FhG)
v2.0	30/09/2021	50	Restructuring of deliverable based on EC reviewer feedback	Kim Fidomski (FhG), Ahmad Hemid (FhG)
v2.1	27/10/2021	80	Rewriting of text based on EC review letter	Kim Fidomski (FhG), Ahmad Hemid (FhG)
v2.2	29/10/2021	85	Revisions according to peer review at Fraunhofer	Benjamin Heitmann (FhG), Kim Fidomski (FhG), Ahmad Hemid (FhG)
v2.3	05/11/2021	90	Revisions according to peer review from work package WP3	Victor Mireles-Chavez (SWC), Kim Fidomski (FhG), Ahmad Hemid (FhG)
v2.4	11/11/2021	98	Revisions according to peer review from the consortium	Ioannis Markopoulos (FNET), Gianna Avgousti (EBOS), Kim Fidomski (FhG), Ahmad Hemid (FhG)
v2.5	12/11/2021	99	Layout changes	Ahmad Hemid (FhG)
v2.6	15/11/2021	100	Final version for second submission	Kim Fidomski (FhG), Ahmad Hemid (FhG)

© TRUSTS, 2021

Disclaimer

The content of the publication herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services.

While the information contained in the documents is believed to be accurate, the authors(s) or any other participant in the TRUSTS consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Neither the TRUSTS Consortium nor any of its members, their officers, employees, or agents shall be responsible or liable in negligence or otherwise however in respect of any inaccuracy or omission herein.

Without derogating from the generality of the foregoing neither the TRUSTS Consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

Copyright message

© TRUSTS, 2020-2022. This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.

Table of Contents

© TRUSTS, 2021

List of Figures

© TRUSTS, 2021

List of Tables

© TRUSTS, 2021

Glossary of terms and abbreviations used

Abbreviation / Term	Description
ACME	Automated Certificate Management Environment
API	Application Programming Interface
AR	Architectural Requirement
CKAN	Comprehensive Knowledge Archive Network
DAPS	Dynamic Attribute Provisioning System
DevOps	Development and Operations
DMA	Data Market Austria
FR	Functional Requirement
GA	Grant Agreement
ID	Identifier
IDS	International Data Spaces
MVP	Minimum Viable Product
UC	Use Case
YAML	Yet Another Markup Language

© TRUSTS, 2021

Executive Summary

The purpose of this deliverable is to document the work performed in the context of Task 3.5 ‘Platform Development & Integration’ of WP3 and describe the current status report of the TRUSTS platform. This report acts solely as a snapshot documentation of the platform implementation status as it is continuously developed according to the DevOps principles.

This is the first of three versions of this deliverable focusing on the initial implementation steps that resulted in the first Minimum Viable Product (MVP) of the TRUSTS platform.

Among others, this deliverable is based on D2.6 “Architecture design and technical specifications document I” where the technical architecture of the TRUSTS platform is described. . D3.9 extends it by describing the implementation start of the platform as well as the evaluated artifacts and the architecture implemented during the reporting period.

Task T3.5 ‘Platform Development & Integration’ is strongly related to the WP2 and T2.4 ‘Architecture design and technical specifications’ addressing deliverable D2.6 ‘Architecture design and technical specifications document I’. The work performed and the contribution between T2.4 and T3.5 established a strong collaboration with all technical partners involved in designing the TRUSTS architecture. A strong collaboration was also established with WP2 ‘Requirements Elicitation & Specification’ and T2.2 ‘Industry-specific functional requirements elicitation and analysis’ and T2.3 ‘Testing framework and benchmarking’, for a good and solid understanding of the functional requirements and trials planned for all three use cases.

A very strong teamwork and collaboration between T3.5 and the rest of WP3 tasks, as well as T3.4 and T3.6 is observed during the initial full platform components analysis process and during the component’s assessment process, as well as during development and testing phases or iterations performed for the first version of the platform.

A collaboration with work package WP4 "Privacy and Preserving Technologies", was established for the understanding of security components that are crucial for the TRUSTS project. In addition, a close collaboration with work package WP5 "Demonstration of the three business-oriented UCs", was established for a better understanding of the TRUSTS trials and the three UCs.

The TRUSTS platform builds on the experiences and best practices from previous initiatives for supporting data markets. These are the Data Market Austria (DMA) [1] and the International Data Spaces (IDS) [2]. Both initiatives have strongly influenced the TRUSTS platform. On a technical level, the platform reuses infrastructure from both DMA and IDS towards a true hybridized implementation. This will result in a platform implementation which aims to combine the best of both previous initiatives while mitigating strategically important weaknesses.

In addition, the platform is aiming for a conceptual alignment with the GAIA-X initiative, as future compatibility with Gaia-X is of strategic importance to the TRUSTS platform. GAIA-X is a project for the development of an efficient and competitive, secure and trustworthy federation of data infrastructure and service providers for Europe. We expect GAIA-X to set important impulses for the data economy in Europa by e.g., communicating with important groups of stakeholders and by setting standards for technical and organizational aspects.

© TRUSTS, 2021

1 Introduction

Following the DevOps principles, in the TRUSTS project platform, releases are happening continuously. This deliverable (D3.9) acts like a snapshot documentation of the TRUSTS Platform. It is the first status report in a series of three reports within the lifetime of the project. D3.9 'Platform Status Report I' due for submission December 2020 (M12), D3.10 'Platform Status Report II' due December 2021 (M24) and D3.11 'Platform Status Report III' due for month 36, December 2022. All three versions of these reports document the current state of the platform regarding its functionality and operational parameters.

D3.9 is related to Task 3.5. To reach the objectives of the task the platform is developed in an iterative way and is based on reusing open-source software. The concept of a Minimum Viable Product (MVP) is used. Using the MVP concept, relevant stakeholders are continuously involved in the development of the platform implementation. At the same time, this approach ensures that all artifacts of the platform are mature and of a high-quality level.

The development started in M6. As a starting point and to ensure a smooth start, T3.5 worked intensively with T2.1, T2.2 and T2.3. The collaboration with T2.2 and T2.4 serves to define functional requirements and to create a list of components for the design of the platform architecture. T3.5 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 gathered in T2.4. Reusing assets from existing platforms and well-known initiatives gives the platform development a head start, as it builds on established and proven technologies. Accordingly, the DMA and IDS components are analyzed and prioritized according to requirements, and maturity.

Throughout the development process, the progress of the various tasks and work packages is monitored. This applies in particular to the tasks mentioned above.

Weekly sprint meetings were held in parallel with this deliverable. Technical experts and programmers intensively discussed the progress of the platform implementation, component dependencies, and obstacles. In this way, the implementation of the platform could be done in an agile way.

The approach mentioned above resulted in the first two Minimum Viable Products of the TRUSTS platform during the reported period. In addition to the assessed and prioritised components and the covered functional requirements (FRs), both the MVP.v0 and MVP.v1 architecture are presented and explained in this deliverable. D3.10, the second version of this report, will cover MVP.v2.

© TRUSTS, 2021

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 Chapters(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. This task is expected to collaborate with T2.1, 2.2 and 2.3 in order to prepare a smooth start of development in M6. The task makes use of 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 a head start by building on established and proven technologies. While, from an implementation point of view, this task covers general functionality (e.g., dataset 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.	Section 2 Section 3 Section 4 Section 5 Section 6	Summary of the functional requirements. Introduction of existing artifacts from well-known initiatives and projects. Description of MVP.v0 and MVP.v1. Details about the current implementation. Conclusion of the deliverable, description of next actions towards the second version of this deliverable (D3.10)
TRUSTS Deliverable			
D3.9 Platform Status Report I 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 Deliverable overview and report structure

The purpose of this deliverable is to document the work performed in the context of WP3 and Task 3.5 'Platform Development & Integration' and describe the status of the TRUSTS platform during the report period. It is a snapshot documentation of the platform.

The deliverable is structured into the following chapters:

- **Industry-specific functional requirements:** The initial functional requirements of the TRUSTS platform are summarized here.
- **Existing artifacts from well-known projects and initiatives:** Existing artifacts from well-known initiatives and projects that are considered for the TRUSTS platform are described in this chapter. These initiatives and projects are the Data Market Austria (DMA), the International Data Spaces (IDS), and the Comprehensive Knowledge Archive Network (CKAN) project.
- **Minimum viable products of the TRUSTS platform:** Both, the Minimum Viable Product Version Zero (MVP.v0) and the Minimum Viable Product Version One (MVP.v1) are described in Chapter 4. The focus is on MVP.v1. The architectures of the MVPs as well as the components included are described in more detail.
- **Implementation:** The components of the TRUSTS platform and their implementation status are listed in Chapter 5. The components are prescribed by the technical architecture of the platform, as described in D2.6 "Architecture design and technical specifications document I". In addition, the interactive mockup is presented to give a first impression of how the web application of the TRUSTS platform can look like in the future from a users' perspective.
- **Conclusions and next actions:** The deliverable is concluded by listing the project results which are described in this deliverable and giving an overview of the next actions towards the second version of this deliverable.

© TRUSTS, 2021

2 Industry-specific functional requirements

After the analysis of the industry-specific functional requirements of the TRUSTS platform in D2.2, a comprehensive set of important functional requirements have been specified. Table 1 summarizes these requirements, providing:

- A unique identifier number for each requirement
- The description of the corresponding requirement
- The tasks involved in the implementation of the requirement

Table 2: TRUSTS functional requirement specifications

ID	Description	Tasks
Datasets and services onboarding functionality and processes		
FR1	The system should provide standardized API descriptions for enabling providers to onboard their datasets and services.	T3.3
FR2	The system should provide APIs that enable its interoperability/federation with other industrial marketplaces and external sources.	T3.3
FR3	The system should be able to provide datasets and service descriptions.	T3.4
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.	T3.4
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.	T3.3, T3.4
FR6	The system should be able to provide datasets and services recommendations to its' users pertaining to their profile and needs.	T3.6
FR7	The system should employ matchmaking mechanisms through which hosted datasets are matched with hosted services (e.g., suitable for their analysis) and vice versa.	T3.6
FR8	The system should identify and match related datasets so as to provide combined and enriched data.	T3.6
FR9	The system should be able to improve datasets and services profiles based on extracted information originating from the available data.	T3.6

© TRUSTS, 2021

Purchasing and billing		
FR10	The system should provide smart contract mechanisms as a validation means of sellers/buyers' agreements.	T3.2
FR11	The system should ensure the integrity and authenticity of the smart contracts signed by its users.	T3.2
FR12	The system should provide a human friendly representation of smart contracts (e.g., natural language).	T3.2
FR13	Signed smart contracts should be legally valid, enforceable, and interpretable.	T3.2
FR14	The system should encompass mechanisms for keeping transactions performed ensuring that they cannot be infringed.	T3.2
FR15	The system should provide billing mechanisms for enabling consumers to pay providers according to the agreed smart contract.	T3.2
FR16	The system must provide alternative and flexible pricing models taking into consideration the diversity of the available datasets and services.	T3.2
FR17	The system should provide brokerage mechanisms for addressing the offerings and demands of the hosted datasets and services.	T3.6
(Meta-)Data Governance		
FR18	The system should provide explicit metadata information for each dataset or service that is accommodated in the platform.	T3.4
FR19	The system should incorporate models, ontologies and taxonomies for the classification and semantic representation of the accommodated datasets and services in the platform.	T3.4
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.	T3.4
FR21	The system should provide mechanisms capable of identifying the provenance of the hosted datasets.	T3.4
FR22	The system should provide mechanisms capable of identifying the lifecycle of the hosted datasets.	T3.4
FR23	The system should harvest metadata extraction from external datasets.	T3.4
FR24	The system should be able to provide semantic information even for unstructured datasets.	T3.4
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.	T3.6

© TRUSTS, 2021

FR26	Dataset discovery should be based on the FAIR[1] principle.	T3.4
Data as a Service and Subscribers management		
FR27	TRUSTS datasets and services should be provided to the users on demand, regardless of geographic or organizational separation between provider and consumer considering all potential territorial legislation/regulatory restrictions.	T3.5
FR28	TRUSTS should be able to be deployed as a federation[2] of distributed, interconnected, and interoperable nodes.	T3.1, T3.3, T3.5
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. Users should be rated for their quality of transactions.	T3.5
FR30	The system should support mechanisms for users' (producers/consumers) subscription opting for different schemes (e.g., annual, monthly, etc.) and authentication.	T3.5
FR31	The system should support corporate accounts that fall under one subscription/enrollment per organization.	T3.5
FR32	The system should enable users to create, read, update, and delete (withdraw or make unavailable) datasets, services, and user profile records.	T3.5
FR33	The system should provide validation criteria for the new enrolled users, as well as reputation schemes with regard to available datasets and services.	T3.5
FR34	The system should allow consumers to announce their need for specific datasets / services if there are not any available, already.	T3.5
FR35	The system should provide notifications regarding datasets / services updates to users that have granted access to them.	T3.5
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.	T3.5, T5.2, T5.3
FR37	TRUSTS UIs and workflows must follow a business-wise rational (e.g., one stop shop), for coherently mapping the market's needs.	T3.5
Data protection		

[1] https://www.go-fair.org/fair-principles/

[2] Federated architecture (FA) is a pattern in enterprise architecture that allows interoperability and information sharing between semi-autonomous de-centrally organized lines of business (LOBs), information technology systems and applications.

© TRUSTS, 2021

FR38	The system must provide cryptographic and secure protocols for the analysis of sensitive data as required by the respective stakeholders.	T4.1
FR39	The system should provide de-anonymization attack assessment and risk analysis for the private / sensitive datasets to be onboard.	T4.3
FR40	The system should employ anonymization tools and guidelines for data anonymization.	T4.3
FR41	The system should provide means for converting algorithms that might compromise the data privacy into safe privacy preserving ones without harming their functionality[3].	T4.5
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.	T4.2
FR43	The system must incorporate a secure infrastructure for the distributed analysis of data based on ML approaches	T4.4
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 ensure that local laws apply to each federated node. Predefined contracts should exist.	T6.1, T6.2, T6.3, T6.4

The functional requirements form an essential basis for the implementation of a first functional version of the platform, since the functional requirements to be realized have a significant influence on the decision about the components used for the implementation. As a result, some of the requirements identified and listed in Table 2 are considered and covered in the initial version of the platform.

[3] Such requirements are set by the relevant stakeholders. Please note that the functional requirement does not impose any specific solution. Respective solutions will be evaluated by the technical work packages of the project.

© TRUSTS, 2021

3 Existing artifacts from well-known projects and initiatives

Extensive research is being conducted in the area of data market platforms, leading to a variety of suitable technologies and components on the market. The TRUSTS platform will reuse several of them. Existing components from two European data market initiatives, DMA, and IDS, are evaluated and prioritized according to the predefined requirements of the TRUSTS platform, maturity, and price. Based on the results, it will be decided which components are suitable for reuse in the TRUSTS platform.

The following seven components developed in the DMA and IDS projects (already referenced in TRUSTS Grant Agreement) as well as in CKAN were reviewed to determine if they were suitable for reuse in TRUSTS.

IDS Trusted Connector	
The connector is part of the IDS essential services. The Trusted Connector is open-source and built on open standards. It is an implementation of the Trusted Connector in the Industrial Data Space Reference Architecture. The Trusted Connector will work as an interface to nodes in the TRUSTS platform. It provides certified connections between pairs of nodes, serving as a proxy between the different services running on a node, and other nodes. The message routing is based on Apache Camel. The Trusted Connector has functionalities for transaction management and usage control.	
Comments	• Can be built and deployed as a docker container • Several protocol adapters are contained: IDSCPv2 (gives additional flexibility in creating routes) • DAPS compatible • Any request to a service is returned as payload • Compatible with services and applications • XML files are used to define possible routes. The provided REST API for creating routes has not been tested yet. Since Apache Camel is the routing framework and has its purpose as an enterprise integration platform, the number of routes is not limited.
Official name	IoT edge platform “Trusted Connector” of the International Data Space
Authors	Fraunhofer AISEC, Julia Schütte
Download / Source repository link	https://github.com/industrial-data-space/trusted-connector
Software licence	Apache License 2.0
Documentation	https://industrial-data-space.github.io/trusted-connector-documentation/docs/overview/

© TRUSTS, 2021

Development example	https://github.com/industrial-data-space/ids-comm-example shows how to connect to the trusted connector within the same organisation / trust boundary.
FRs addressed	FR 27, FR 28, FR 31, FR 38, FR 39, FR 40, FR 41, FR 42, FR 43

IDS Broker	
The IDS Metadata Broker supports the implementation of customized broker solutions. The broker, and its accompanying metadata store, serves as a central repository of all metadata regarding assets, participants and resources, as specified in the IDS-IM. It is a web application that serves as a wrapper of a knowledge graph that is compliant with the IDS-IM ontology. It collects metadata from the different corporate nodes and provides it to the smart contract executor and the platform interface in the User Portal Node (regarding assets), as well as to the Business Support Services (regarding organizations).	
Comments	Collects information from different nodes, their content, addresses, etc.
Official name	IDS Metadata Broker
Authors	Fraunhofer IAIS
Download / Source repository link	https://github.com/International-Data-Spaces-Association/metadata-broker-open-core
Software licence	Apache License 2.0
Documentation	https://github.com/International-Data-Spaces-Association/metadata-broker-open-core/blob/master/README.md
Development example	Was provided in direct consultation to the Fraunhofer IAIS team responsible for development, in a pair of workshops.
FRs addressed	FR 1, FR 3, FR 5, FR 18, FR 21, FR 22, FR 23, FR 25, FR 26

IDS Connector Framework
The IDS Connector Framework is developed to simplify the development of an IDS Connector. The connector is a part of the IDS essential services. It can serve as the basis for an alternative implementation of the connector, which TRUSTS could use together with the Trusted Connector. The framework provides basic functionality for processing IDS messages and communicating with the DAPS and brokers. Since different connectors may have different requirements in terms of protocols and data

© TRUSTS, 2021

endpoint types, the final implementation is application dependent. The TRUSTS platform implementation must take this into account when connecting between the IDS Connector Framework and the Trusted Connector.	
Official name	IDS Connector Framework
Authors	Fraunhofer ISST, Steffen Biehs, Dominik Schütgens, Sebastian Opriel
Download / Source repository link	https://gitlab.cc-asp.fraunhofer.de/fhg-isst-ids
Software licence	Apache License 2.0
Documentation	https://gitlab.cc-asp.fraunhofer.de/fhg-isst-ids/ids-framework/-/blob/dev/README.md
FRs addressed	FR 27, FR 28, FR 31

DMA Metadata Mapper	
Performs mapping of metadata files, converting them from one schema to another. Any asset provider can make use of this. This component transforms existing metadata that organizations might have in their pre-existing data stores, into the IDS-IM specification. Once this metadata is converted, it can be ingested into the local Metadata Registry and then broadcast into the Central Catalogue for advertising to other participants.	
Official name	DMA Metadata Mapper
Authors	-
Download / Source repository link	https://gitlab.com/datamarket-austria
Software licence	MIT License
Documentation	https://datamarket.gitlab.io/Documentation/
Development example	-
FRs addressed	FR 23

© TRUSTS, 2021

CKAN Framework	
CKAN is an open-source data management system. CKAN provides a powerful system for publishing, sharing, and working with data. It has an application programming interface (API) that allows the development of extensions. Accordingly, CKAN supports the development of a web application for data exchange.	
Official name	Comprehensive Knowledge Archive Network Framework
Authors	
Download / Source repository link	https://github.com/ckan/ckan
Software licence	GNU Affero General Public License (AGPL) v3.0
Documentation	http://docs.ckan.org
FRs addressed	FR 36, FR 37

IDS Information Model (as a concept)	
The IDS information model is a RDFS/OWL ontology. It facilitates compatibility and interoperability. Data products can be described, published, and localized using the information model. The described functionality enables a consistent interpretation and use of data.	
Official name	International Data Spaces Information Model
Authors	List of authors
Download / Source repository link	https://github.com/International-Data-Spaces-Association/InformationModel
Software licence	Apache License 2.0
Documentation	https://international-data-spaces-association.github.io/InformationModel/docs/index.html#
Development example	https://jira.iais.fraunhofer.de/stash/projects/ICTSL/repos/ids-infomodel-demo/browse
FRs addressed	FR 18, FR 19, FR 20, FR 21, FR 22

© TRUSTS, 2021

IDSA DAPS	
This is a component to support secure communication, authentication, and authorization in TRUSTS infrastructure. The DAPS is a part of the IDS essential services.	
Official name	Fraunhofer AISEC Dynamic Attribute Provisioning Service (DAPS)
Authors	Fraunhofer AISEC, Gerd Brost
Download / Source repository link	Closed source, but an instance of this is running at all times and is meant to be used for development and testing purposes.
Software licence	No reuse of software planned. TRUSTS will need to develop a long-term strategy regarding this component, if it is required in the TRUSTS platform.
Documentation	https://github.com/International-Data-Spaces-Association/IDS-G/tree/draft/core/DAPS
Development example	https://app.swaggerhub.com/apis/IDS_Association/DynamicAttributeProvisioningService/0.1.3 and https://daps.aisec.fraunhofer.de/endpoints
FRs addressed	FR 38, FR 43, FR 44

© TRUSTS, 2021

4 Minimum viable products of the TRUSTS platform

In TRUSTS, the Minimum Viable Product (MVP) versions can be seen as a clearly specified handover point where WP3 hands over a specific version of the platform to WP4 and WP5. Part of the MVPs are previously defined components. Components may appear in multiple MVP versions. Part of the review of the MVPs by WP4 and WP5 are the technical test scenarios defined by WP3 for each MVP. WP4 and WP5 must return the results of the technical test scenarios to WP3 as input for the next iteration or version of the MVP.

A Minimum Viable Product is a functional iteration of a product with just enough features to be usable to meet a minimum number of requirements. This makes it possible to quickly obtain user feedback and learn from it. This way, undesirable development can be detected and prevented as early as possible.

Until the writing of this text, two MVPs have been started, the first one is MVP.v0, then followed by MVP.v1. The following is a discussion of both, including their goals, components, and architecture.

4.1 MVP versions

The architectures and the included components of MVP.v0 and MVP.v1 are described in the following sections, providing an overview of the features of the first two versions of the platform.

4.1.1 MVP.v0

As an initial product, MVP.v0 is limited to the essential product components and covers only a minimum number of requirements. As a quick start implementation, the initial MVP version had seven goals:

1. **Proof of concept**: Make sure that the concept works for the TRUSTS project to minimize shortcoming for example.
2. **Initial experience**: First experiences are gained. A better understanding of some components is created.
3. **Select core components**: For MVP.v0 most essential components are selected for a first iteration of the product to reach these seven goals.
4. **Define roadmap for MVPs**: For further development, a MVP roadmap is defined.
5. **Study Apache Camel features**: Apache Camel is an open-source integration framework. It can be used to integrate a variety of systems that either consume or produce data.
6. **Reduce project risks**: Side effect of the Minimum Viable Product concept, as users are involved in the development process at an early stage.
7. **UCs partners prepare to use the platform**: Strong link between T3.5 and the use case trials. An initial MVP as a prerequisite for the use case trials.

Based on these predefined goals and on the evaluation of existing components (see Chapter 3), the Trusted Connector, IDSA DAPS, and the CKAN Framework were selected as most suitable for reuse in an initial version of the TRUSTS platform and to ensure a smooth start of the implementation.

© TRUSTS, 2021

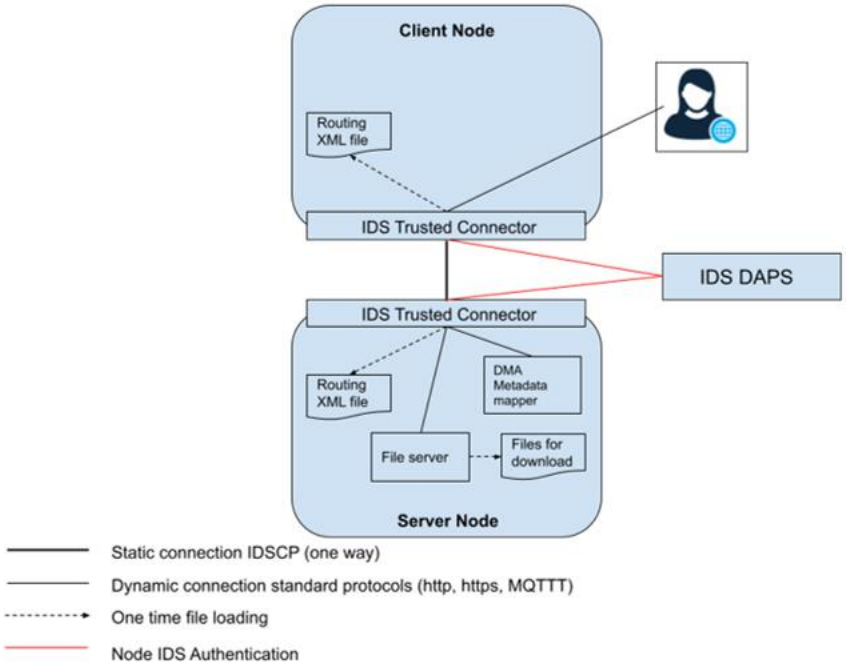

Figure 1: MVP.v0 architecture

Figure 1 shows the architecture of MVP.v0. This initial MVP version has only two nodes: server node and client node. The communication between the nodes is possible via the IDS Trusted Connector.

The IDS Trusted Connector is a core component of the TRUSTS infrastructure, providing secure communication inside of the TRUSTS ecosystem between participants and inside any participant between applications. For dynamic routing, the Trusted Connector supports XML.

The IDS DAPS component issues access tokens that are required to access the services and the data of other connectors. The protocol implemented in DAPS allows certified connectors to authenticate to the DAPS. In return, the connectors retrieve an access token that they can use to access other connectors. The decision about permissible access is not made by the DAPS, but by the connector that the requesting connector wants to access.

The Metadata Mapper component performs mapping of metadata files, converting them from one schema to another. Any asset provider can make use of this. It takes as input a sample metadata file and creates a mapping file, which is used to configure the DMA metadata mapper, which will in turn convert other metadata files (like the sample) into the desired metadata format.

4.1.2 MVP.v1

As all main goals of MVP.v0 were achieved, it serves as a starting point for MVP.v1. Based on the experience gained from the development of MVP.v0, selected components were confirmed. The next iteration of the platform development was the creation of MVP.v1.

© TRUSTS, 2021

In the following, the augmented goals for MVP.v1 are listed:

1. **Create infrastructure for the use case trials**: MVP.v1 as starting point for WP5's use case trials.
2. **Create a platform for main features of TRUSTS**: Offer datasets, apps, and services as main features of the TRUSTS platform.
3. **Dynamic communication between nodes**.
4. **Simplify deployment process to nodes**.
5. **Proof that the platform can support any dockerized applications, services and non-dockerized services** (with some restriction to protocol types at first stage).
6. **Technical foundation for UI components to platform**.
7. **Training use case partners on platform usage**: Necessary for use case trials to be conducted.
8. **Get feedback from use case partners**: Feedback will be collected based on the use case trials and will be incorporated into MVP.v2.

Figure 2 demonstrates the MVP.v1 architecture. As it has been shown, it has three nodes: central node and two corporate nodes (but even more than two nodes, also can be added). The communication between the nodes is possible via the IDS Trusted Connector.

The IDS Trusted Connector is a core component of either central or corporate nodes, providing secure communication inside of the TRUSTS ecosystem between participants and inside any participant between applications. Each node is owned and operated by different organizations, on which the trusted exchange of digital assets is possible. Moreover, a node is a computing environment (e.g., a virtual machine) which runs a set of standardized components to support the different functionalities of the platform. Along these components, common to all nodes, organizations can execute their in-house developed applications and services. These applications and services can then be traded through the TRUSTS platform, and can also, in turn, consume and produce other assets through the platform.

The two types of nodes included in MVP.v1 have infrastructure built around the Trusted Connector, as follows:

- **Central Node:** exists to support the operation of the whole TRUSTS platform, playing the role of authorization, monitoring, smart contract executor, catalog, application repository, among others.
- **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. Among the components being set up, is the Corporate Interface, a web application through which the organization's users can interact with the TRUSTS platform.

The IDS DAPS component has the same task explained in the Subsection 4.1.1. In the meanwhile, TRUSTS repository component saves the apps (docker images that participant organizations can, after signing a contract with the provider, pull and execute in their premises) and loads deployment scripts and configuration files.

© TRUSTS, 2021

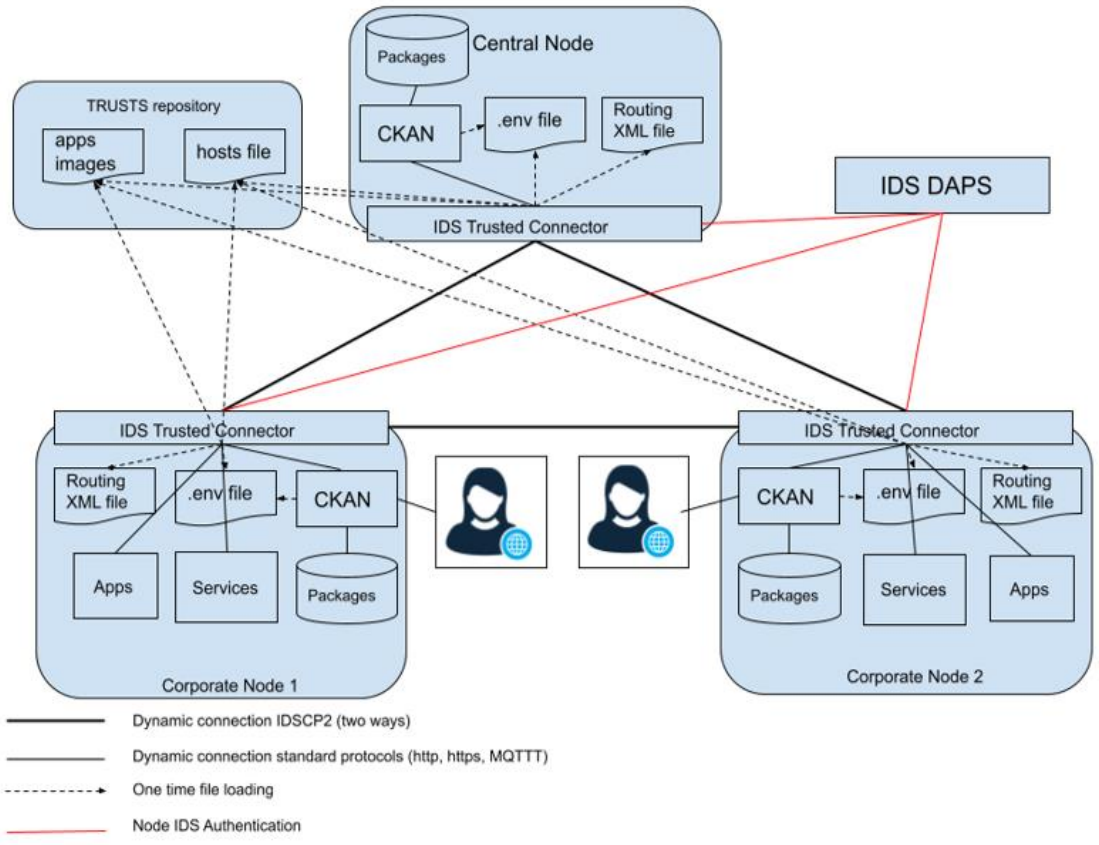

Figure 2: MVP.v1 architecture

Components deployed within a node are connected among themselves through a network environment. The connections between different nodes are all done using the IDSCPv2 communication protocol and are handled by the set of standardized components. In particular, connections coming from outside the node are first received by the Trusted Connector component and then distributed to other components. The IDSCPv2 protocol, apart from the standard asymmetric-key encryption of modern HTTPS connections, allows for conveying information specific for the functionalities of the platform. This includes user identification tokens, names and ports of the different nodes involved, as well as IDs of the components they are destined to.

© TRUSTS, 2021

5 Implementation

In this chapter, the implementation process of the TRUSTS platform is presented. First, the planning for the evolution of the platform by employing an idea from agile development is offered. Then, the selected components of the TRUSTS platform and how they are going to address the functional requirements are demonstrated. Followed by the implementation status of those components. Finally, it is concluded by offering the TRUSTS platform mockups

5.1 Planning for the evolution of the platform with agile methods

In the TRUSTS project, we will use agile methods in achieving continuous improvement. In particular, we will use the concept of a "Minimum Viable Product (MVP)" to involve relevant stakeholders in the evolution of the platform, at different points in time during the project.

A Minimum Viable Product [4, 5] 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 should be seen as a clearly specified hand-off point, at which a specific combination of platform components in specific versions is released to all partners who are working on the implementation of use cases and on the implementation of privacy enhancing technologies. These partners are the internal customers of the MVP within the TRUSTS project.

Every MVP version will be released internally within the project, together with a description of how to perform technical tests. The results of these technical tests will then be used as input for the next version of the MVP.

The goals for the MVPs are characterized by three dimensions: integration of APIs, functional integration and operational integration.

Integration of Application Programming Interfaces (APIs):
So-called "software mock objects" can be used to implement and test APIs in isolation, independent of the progress of the implementation of the API. This can be used to distribute the work on components which will be connected to the APIs, and for integrating the APIs of existing software with newly developed software. This dimension is also important for validating the interplay between components which are run on different hosts.

The testing of APIs will result in a better understanding of the component view on all software artifacts in the project and will enable better communication between the technical experts in the project. In addition, it allows for testing of essential and shared infrastructure software components which are reused from either the IDS or DMA. Finally, it will allow for testing of deployment mechanisms.

Functional integration:
The testing of functional integration is concerned with the details of the interplay between components on the level of data and functions. Instead of testing APIs, functional testing is concerned with testing if components

© TRUSTS, 2021

can correctly handle incoming data and return the correct kind of output data. In addition, any side effects or error conditions which can appear, have to be accounted for.

The testing of functional integration benefits from a complete documentation of the data models used by all data stores in the project. It is also desirable to connect all implemented functions to realistic data sources. In addition, functional integration is very important to confirm that components developed by different project partners can interact in a correct way, and that the handling of errors in the input data allows resilient operation of the platform components.

Operational integration:
The testing of operational integration is concerned with providing functionality in a way which simulates the envisaged customers of the platform. A prerequisite of this is that the platform components can communicate with each other via the correct APIs, and that functions of components are able to handle input and output data correctly and to be resilient towards errors. Operational integration is also concerned with the user experience and the way in which the overall functionality of the platform is communicated and packaged to the user. As a result of operational integration, all required aspects will be tested.

5.2 Components of the TRUSTS platform addressing the functional requirements

The following lists the planned components of the TRUSTS platform. The components are prescribed by the technical architecture of the platform, as described in D2.6 “Architecture design and technical specifications document I”.

Table 3: List of components

ID	Component
C1	Trusted Connector
C2	Dataflow Router
C3	Reverse Proxy
C4	Recommender
C5	Platform Interface
C6	Landing Page
C7	Asset Consumer

© TRUSTS, 2021

C8	Notification Service
C9	Metadata Mapper
C10	Usage Control
C11	Mapping Builder
C12	Corporate Interface
C13	Service Consumer Adapter
C14	Data Exchange TRUSTS Component
C15	Data Exchange Client Component
C16	Registry of Data Markets
C17	Business Support Services
C18	Broker + Metadata Storage
C19	App Store
C20	Identity Provider + Key Distribution System
C21	Vocabulary Services
C22	Dynamic Attribute Provisioning System (DAPS)
C23	Automated Certificate Management Environment (ACME)
C24	Smart Contract Executor

In the following we provide an overview showing which components need to address which functional requirements:

© TRUSTS, 2021

Table 4: Summary of connections between functional requirements and components

ID of functional requirement (FR)	IDs of components which address the requirements
FR 1	C5, C12, C18
FR 2	C14, C15, C16
FR 3	C5, C12, C18
FR 4	C5, C12, C14, C15, C16
FR 5	C5, C12, C18
FR 6	C4
FR 7	C4
FR 8	C4
FR 9	C19
FR 10	C24
FR 11	C24
FR 12	C24
FR 13	C24
FR 14	C24
FR 15	C24
FR 18	C18
FR 19	C21
FR 20	C21
FR 21	C18, C21

© TRUSTS, 2021

FR 22	C18, C21
FR 23	C9, C11, C14, C15, C18
FR 24	C19, C21
FR 25	C5, C18, C21
FR 26	C18
FR 27	C1, C2, C5, C7, C12
FR 28	C1
FR 29	C5, C7, C10, C12, C20
FR 30	C5, C12
FR 31	C1, C5, C12
FR 32	C5, C12
FR 33	C5, C11, C17
FR 34	C5, C12
FR 35	C8
FR 36	C5, C12, C20
FR 37	C5, C12, C20
FR 38	C1, C10, C19, C20, C22, C23
FR 39	C1, C19
FR 40	C1, C19
FR 41	C1, C19

© TRUSTS, 2021

FR 42	C1, C19
FR 43	C1, C19, C20, C22, C23
FR 44	C5, C10, C12, C20, C22, C23

5.3 Summary of the implementation status of the planned components

The development of MVP.v1 was done thanks to close collaboration with tasks T3.1, T3.3, T3.4. and WP5. CKAN currently covers the needs of metadata (T3.4) partially and interoperability (outcome of T3.3) issues that's why it was selected as one of the base components.

The WP5 team actively participated in the design of the MVP.v1 that allowed them to start partial trials for the three use cases. On behalf of Use Case 1, T3.5 also developed the backend of its three applications on the following programming languages, Python, R, C#. In addition, the implementation of the ML and AI algorithms intended for the solution of the AML Risk Assessment and Transaction Monitoring has been achieved. Following the development, and the outputs of T3.1, they were also dockerised as it was required, in order to be tested and evaluated into the TRUSTS platform in regard to their effectiveness and efficiency. MVP.v1 implements the main functional requirements like offering datasets, applications and services through high secured and controlled channels using peer to peer communication.

During MVP.v1 implementation, new functionality of CKAN and new configuration of IDS Trusted connector were developed and tested to achieve all goals of MVP.v1. Additionally, to meet MVP's goal were added new and changed existing UI in CKAN.

On the CKAN side (which was used as corporate- and central-node interface), a pair of plugins were developed and bundled into a new extension. One of these plugins allows for the inclusion of new types of assets into CKAN, namely services and applications, with their own set of metadata fields and file requirements. The second plugin allows for exchanging metadata between different nodes, by making use of CKAN's REST API through the Trusted Connector. This plugin, referred to as "Push mechanism", allows for active metadata exchange between CKAN instances, as opposed to the passive, harvested based, approach commonly in use.

In this first iteration, it was required that both services and applications be described using an OpenAPI3.0 compliant YAML file describing the web interface, and a docker-compose file describing the deployment (in the case of apps) and the connection details (in both cases). These files were not programmatically exploited in this first iteration but, rather, the operator of a corporate node that is acquiring an asset has to download the corresponding files and act upon them. Namely, the docker-compose file must be executed, which in turn pulls the image of the application from the registry, or the service client (or browser) has to be directed to the correct endpoints based on the OpenAPI description. This manual manipulation of files at the time of asset exchange is only a temporary solution, as the requirements specify for these operations to be as seamless to the user as possible. Next iterations will make this process successively more user-friendly and less dependent on technical expertise. Some limitations in the form of the number of resources (ports, memory, etc.) have

© TRUSTS, 2021

been already identified for the execution of applications, which will limit the user-friendly-ness of deployments acquiring large amounts of applications and processes. We remain confident, however, that organizations in such cases will have at their disposal the technical expertise that, coupled with the TRUSTS provided documentation, can deal with these issues.

On the Trusted Connector side, XML files were created to configure the embedded Apache Camel component, allowing for requests to be routed through to the CKAN component, any service being provided by a node, or an application being executed in one of the nodes. For this first iteration, these files were deployed by hand according to the use-cases. In brief, this allows for consumers (human or machine) of services and applications to send all their requests to the trusted connector which will then, in turn, forward them either to a container running in the same node (in the case of application), or to another connector (in the case of services). In the case of applications, this allows for the inclusion, in a later stage of implementation, for usage controls or usage metering methods so that, even when the application is executed in the premises of the consumer, its use can be recorded and accounted for. In the case of services, this allows for verification (through the exchange of DAPS provided tokens) of the identity of the consuming node.

From the deployment point of view, a docker compose file was provided for each of the corporate nodes to launch the components executed therein, after pulling them from the TRUSTS-operated docker registry. The configuration of these nodes is done by adjusting a single file (whose contents are fed into the corresponding containers in the form of environment variables), allowing for a certain amount of customization in the deployment. Apart from downloading the docker-compose file and creating the configuration file, a node operator had to announce to the central node administrator some parameters needed for connecting to it, which would then in turn be broadcast to the other participants. It is important to note that the node deployment operation will always be very technical in nature, requiring operators to engage their technical personnel for this event. Further simplification of the deployment process will be undertaken in the next iterations of the platform.

In the following table, the implementation status of all components is listed.

Table 5: Summary of the implementation status of the platform components

Component name	**Status**	**Details**
C1 - Trusted Connector	Reused from IDS	IDS Trusted Connector Version 4.0.0
C2 - Dataflow Router	Planning and design phase	-
C3 - Reverse Proxy	Reused from Open-Source Software	NGINX, Version 1.12.0
C4 - Recommender	Developed by KNOW for TRUSTS	Details are described in D3.12 (M18)

© TRUSTS, 2021

C5 - Platform Interface	Reused from Open-Source Software	CKAN (Comprehensive Knowledge Archive Network), Version 2.9.3 Source Code in TRUSTS repository: https://gitlab.com/trusts-platform/ckan/-/tree/trusts
C6 - Landing Page	Planning and design phase	-
C7 - Asset Consumer	Planning and design phase	CKAN, as listed for C5 – Platform Interface might also be used to implement this component.
C8 - Notification Service	Planning and design phase	-
C9 - Metadata Mapper	Reused from DMA	Source Code in TRUSTS repository: https://gitlab.com/trusts-platform/dma-simple-RMLmapper More details in D3.7 (M18)
C10 - Usage Control	Planning and design phase	More details in D3.7 (M18).
C11 - Mapping Builder	Reused from DMA	Source Code in TRUSTS repository: https://gitlab.com/trusts-platform/dma-metadata-mapping-builder

C12 - Corporate Interface	Reused from Open-Source Software	CKAN (Comprehensive Knowledge Archive Network), Version 2.9.3 Source Code in TRUSTS repository: https://gitlab.com/trusts-platform/ckan/-/tree/trust s
C13 - Services Consumer Adapter	Planning and design phase	-
C14 - Data Exchange TRUSTS Component	Planning and design phase	This component is deployed in TRUSTS nodes and ingests assets from third-party data-markets and EOSC and its related initiatives into the TRUSTS catalog. More details in D3.4 (M12).

© TRUSTS, 2021

C15 - Data Exchange Client Component	Planning and design phase	This component provides an interface to specify and select data assets and to map their metadata schema into a format understood by TRUSTS. The component communicates with the Data Exchange TRUSTS Component. More details in D3.4 (M12).
C16 - Registry of Data markets	Planning and design phase	This component lists existing third party data markets and relevant initiatives of EOSC. It serves as an address book routing the communication between the Data Exchange TRUSTS Component and the Data Exchange Client Components installed on the premises of third party data markets and EOSC initiatives. More details in D3.4 (M12).
C17 - Business Support Services	Planning and design phase	-
C18 - Broker + Metadata Storage	Reused from IDS	IDS Metadata Broker + Apache Jena Fuseki Details in D3.7 (M18)
C19 - App Store	Planning and design phase	-
C20 - Identity Provider + Key Distribution System	Planning and design phase	-
C21 - Vocabulary Services	Developed by SWC	PoolParty https://www.poolparty.biz/ Details in D3.7 (M18)
C22 - Dynamic Attribute Provisioning System (DAPS)	Reused from IDS	Instance of service provided for testing by Fraunhofer
C23 - Automated Certificate Management Environment (ACME)	Reused from IDS	Instance of service provided for testing by Fraunhofer

© TRUSTS, 2021

C24 - Smart Contract Execution	Under evaluation regarding the possibility of development by FhG and EMC	More details will be in D3.3 (M36)

5.4 TRUSTS platform mockups

For the development of the interactive mockups for the TRUSTS platform, an iterative design process was used. The process involved incremental extensions and refinements of the developed mockups as well as reviews from the mockup committee. The mockup committee consists of cross-work package partners.

The mockups are used to develop an individual and suitable design of the TRUSTS platform. The extensive mockup serves as the basis for the web application development. It behaves almost identically to the planned web application.

So far, a total of 53 designs have been developed and used to create an interactive mockup demo of the TRUSTS platform. Indicative screens of the designed mockups are presented in figures 3, 4 and 5. The interactive mockups were used to demonstrate various functionalities, pages, and designs:

- **Landing page**: The landing page of the TRUSTS platform adopts a long-scrolling page model with parallax themes.
- **Registration pages**: The registration pages include both institutional and individual user account registration pages.
- **Login page**: The login page allows a user to gain access to an application. In TRUSTS the user can input the username and password to log in.
- **Dashboard for content contributors**: The content contributor dashboard for example shows the individual's tags, purchases, and uploads by category.
- **Data assets page for content contributors**: On the data assets page the user can filter assets by specific categories, assets are recommended, and best-selling and top-rating assets are displayed at a glance.
- Upload data asset pages, adopting a two-step process
- Upload application pages, adopting a two-step process
- Upload service pages, adopting a two-step process
- Dashboard for users who have not contributed any assets to the TRUSTS platform
- Dashboard for tagging of assets from federated marketplaces
- Exploration of available data assets
- Exploration of available applications
- Exploration of available services
- Data asset preview page
- Application preview page
- Service preview page
- Search results page: On this page all the search results are shown. Search results can be sorted

© TRUSTS, 2021

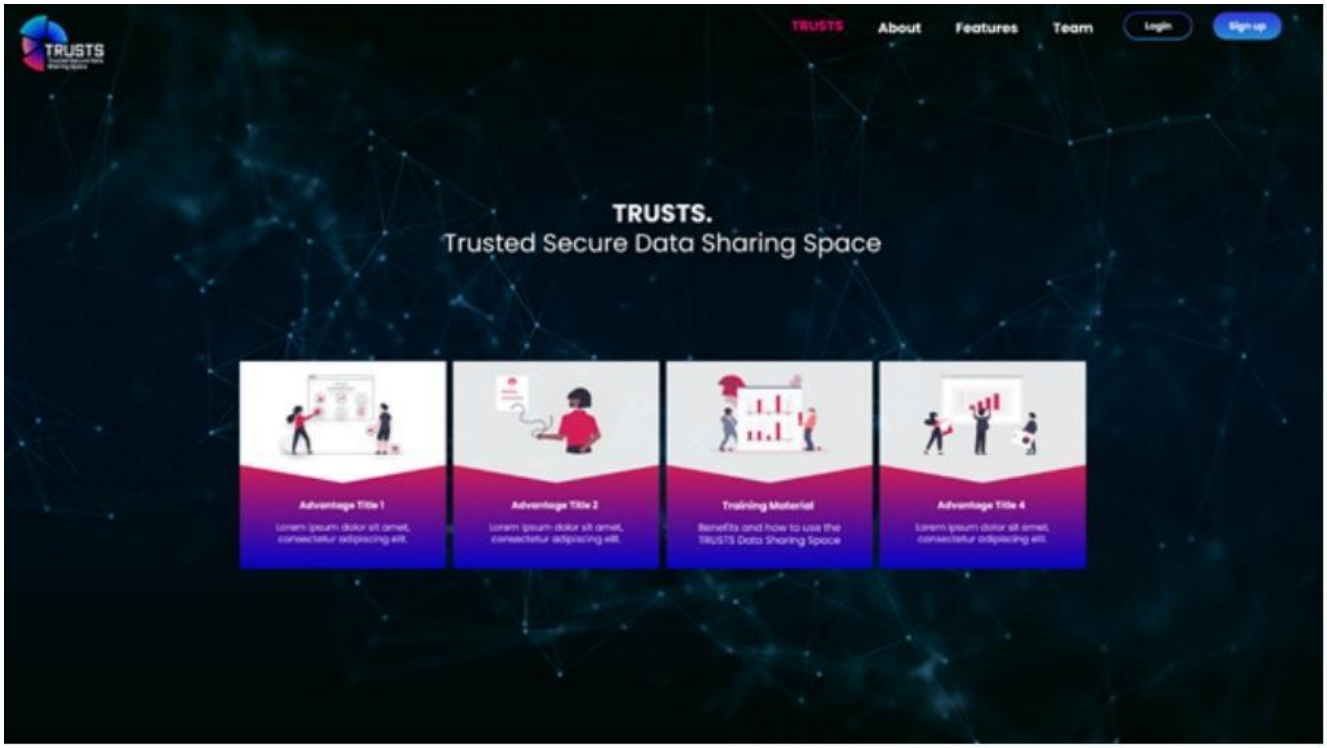

Figure 3: TRUSTS Landing page mockup

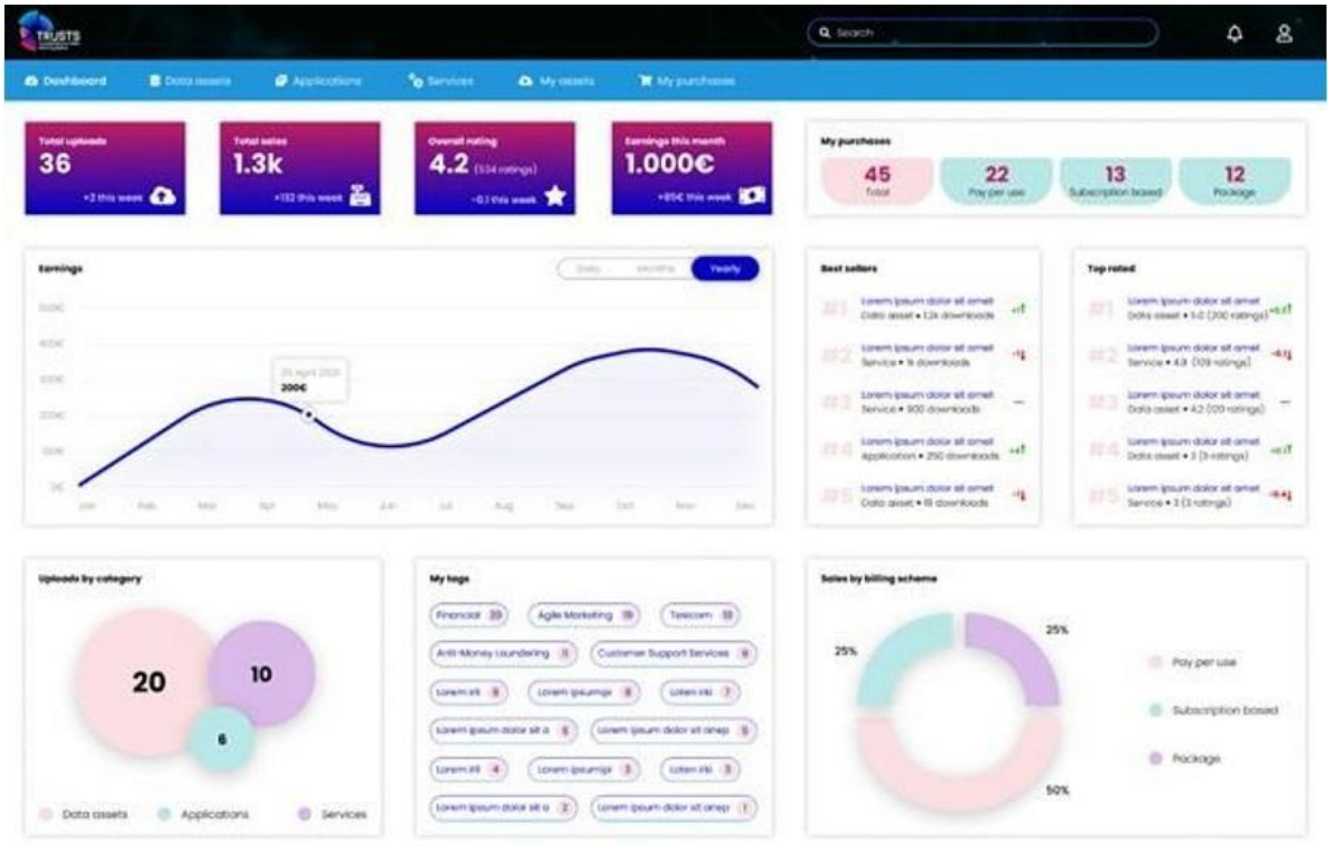

Figure 4: Mockup of TRUSTS Dashboard for content contributors

© TRUSTS, 2021

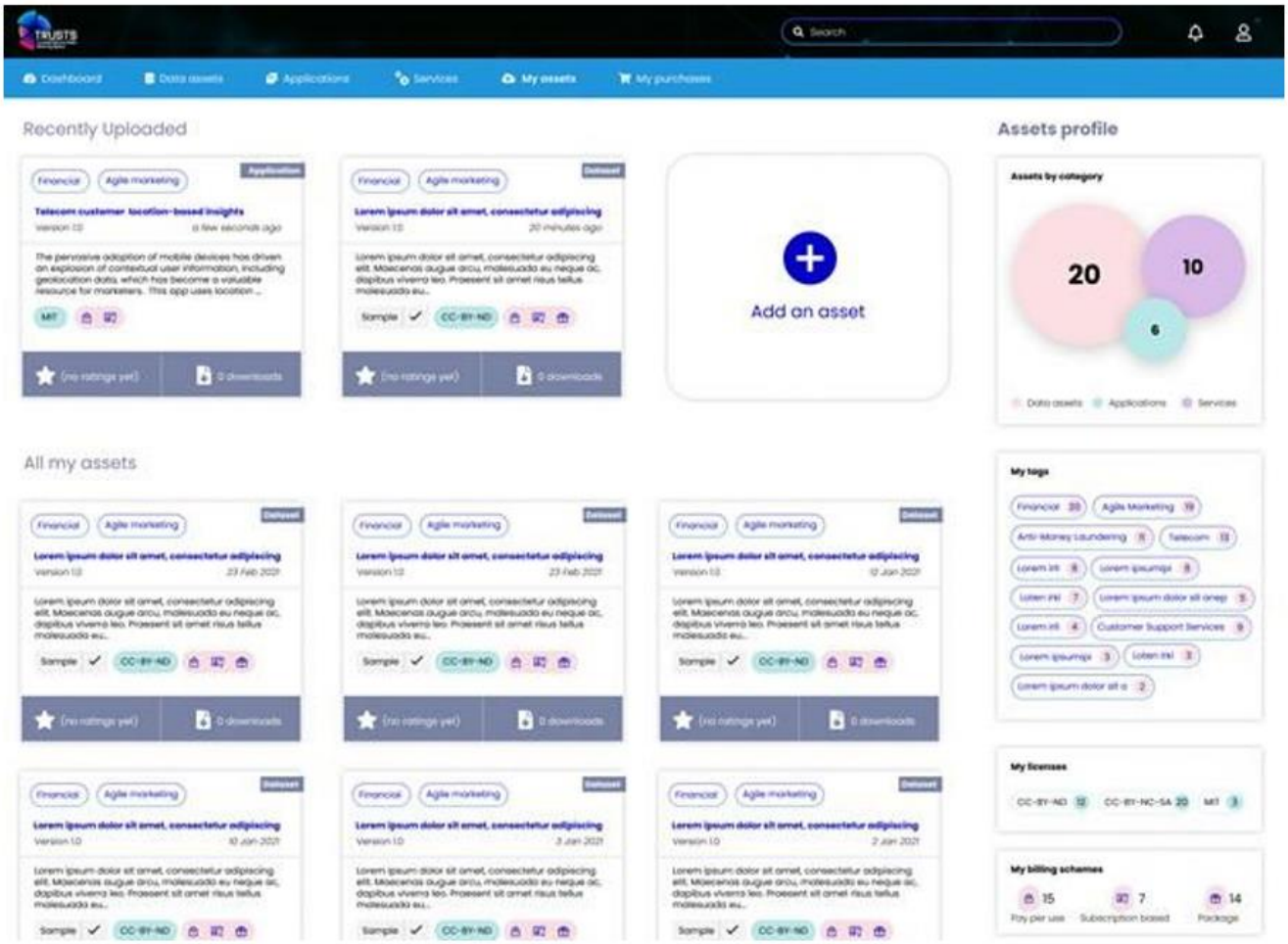

Figure 5: Mockup of TRUSTS 'My Assets' page for content contributors

6 Conclusion and next actions

The purpose of this deliverable is to document the work performed in the context of WP3 and Task 3.5 'Platform Development & Integration'. This deliverable describes the current status report of the TRUSTS platform. It documents a snapshot of the platform. This is the first of three versions of this deliverable. This version focuses on an initial functional product of the platform that covers only a minimum number of the functional requirements identified in WP2 (see Table 1). The architecture of the initial version of the platform is explained. It is based on the technical architecture of the TRUSTS platform described in deliverable D2.6 "Architecture design and technical specifications document I". Thus, this deliverable is in strong collaboration with other deliverables.

In Chapter 3, several components are documented and introduced. The components were described in terms of their suitability for an initial functional version of the platform to ensure a head start of the implementation. Next, the initial TRUSTS platform is described from an architectural perspective by giving a general overview of the technical architecture of the first two MVPs and describing the different roles of nodes in the platform. It is also described how the architecture supports the trading of different kinds of assets on the initial version of the TRUSTS platform.

This is followed by a list of the components of the TRUSTS platform and their implementation status. The planning for the evolution of the platform with agile methods is described. The development process used in the TRUSTS project allows multiple iterations of the platform architecture. In particular, the concept of a Minimum Viable Product is used to involve relevant stakeholders in the evolution of the platform, at different points in time during the project. Finally, the mock-up implementation was shown to provide a high-level overview of the TRUSTS platform in future from the user's perspective.

For the second version of this deliverable (D3.10), we will describe the next iteration of the TRUSTS platform. It means that for example the next version of the platform, MVP.v2, will be introduced and explained in more detail in D3.10. MVP.v2 will contain additional component implementations. Available experience and feedback will be incorporated into the development of MVP.v2. Furthermore, D3.10 will be based on interactions and feedback with the other tasks in WP3, and on interactions and feedback with the use cases in WP5. The technical focus will be on integrating the constituent components into a homogeneous platform. As part of this, it is planned to extend the knowledge about connectors. Another aspect involves setting up a testing environment with instances of all components required for providing a secure data marketplace and secure services. All future iterations of the platform implementations after MVP.v2 will be described in deliverable D3.11, which is the third and final version of the platform status report. D3.11 is scheduled to be released at the end of the project duration.

© TRUSTS, 2021

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] G. Eggers, et al. "GAIA-X Technical Architecture", Federal Ministry for Economic Affairs and Energy (BMWi), June 2020. Accessed via: https://www.data-infrastructure.eu/GAIAX/Redaktion/EN/Publications/gaia-x-technical-architecture.pdf?__blob=publicationFile&v=5

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

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

© TRUSTS, 2021