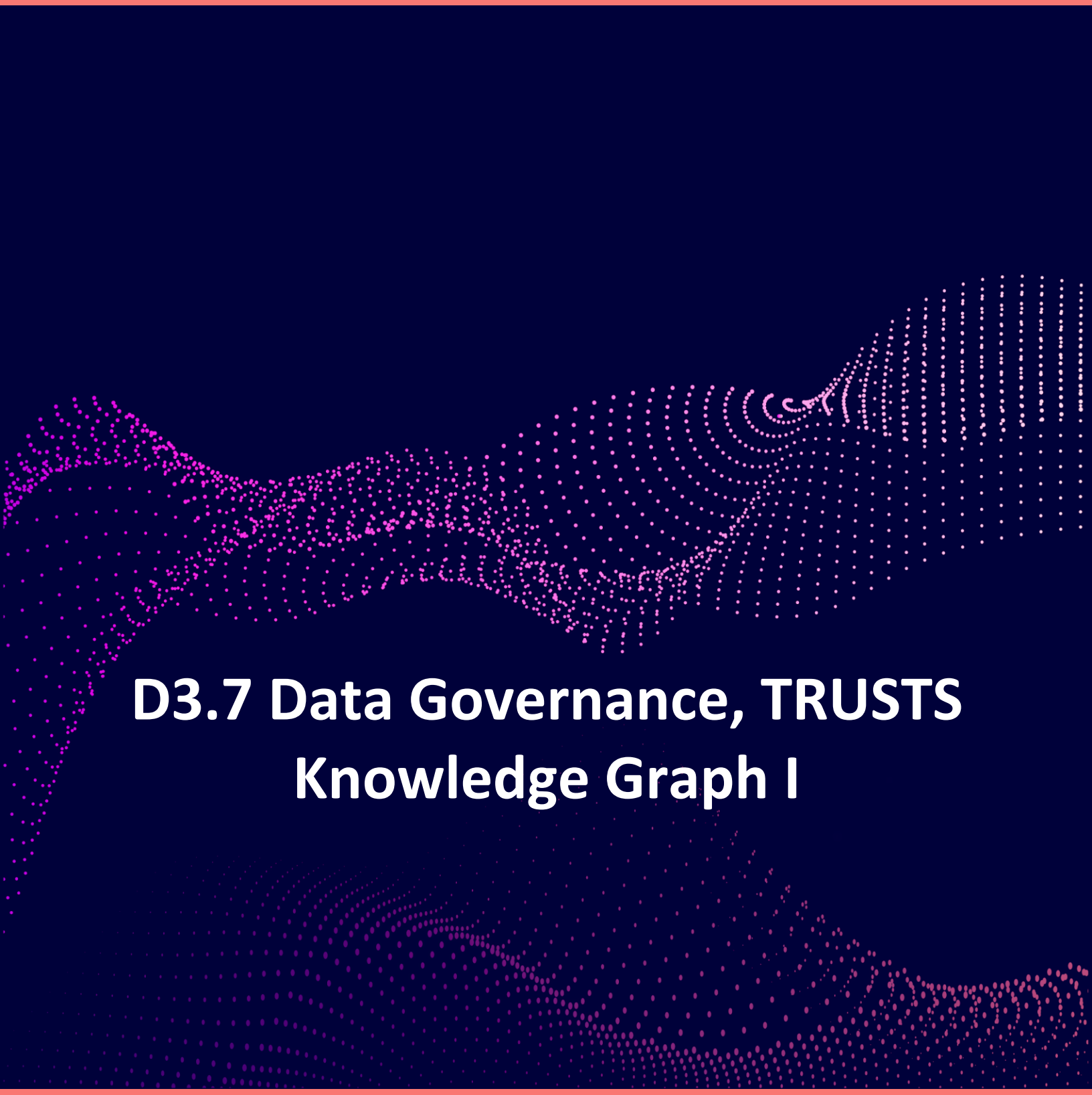

D3.7 Data Governance, TRUSTS Knowledge Graph I

Authors: **Victor Mireles, Stefan Gindl, Sotiris Karampatakis, Michael Boch**

Additional Information:

June 2021

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

D3.7 Data Governance, TRUSTS Knowledge Graph 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	D3.7 Data Governance, TRUSTS Knowledge Graph I		
Work Package	3		
Contractual due date	30/06/2021	Actual submission date	30/06/2021
Nature	Report	Dissemination Level	Public
Lead Beneficiary	Semantic Web Company		
Responsible Author	Victor Mireles, Semantic Web Company		
Contributions from	Stefan Gindl, Michael Boch, Research Studios Austria Sotiris Karampatakis, Artem Revenko, Martin Kaltenböck, Semantic Web Company Nikos Fourlataras, Relational George Margetis, FORTH Ahmad Hemid, Fraunhofer IAIS, Diether Thelier, KNOW Center		

Revision history (including peer reviewing & quality control)

Version	Issue Date	% Complete[1]	Changes	Contributor(s)
v1.0	21.04.2021	1%	Initial Deliverable Structure	Victor Mireles
v2.0	18.06.2021	70%	Internal Version Review	Victor Mireles Stefan Gindl Sotiris Karampatakis Nikos Fourlataras George Margetis Diether Thelier
v3.0	29.06.21	95%	Revisions	Victor Mireles Stefan Gindl Ahmad Hemid Sotiris Karampatakis Artem Revenko
v4.0	30.06.21	100%	Final Version	Martin Kaltenböck

Disclaimer

[1] According to TRUSTS Quality Assurance Process:

1. to be declared

Copyright message

Table of Contents

1 Executive Summary 9

2 Introduction 11

2.1 Definitions 12

2.2 Mapping Projects' Outputs 13

2.3 Deliverable Overview and Report Structure 15

3 Use of a Knowledge Graph in TRUSTS 16

3.1 Discovering assets 16

3.2 Node operation 19

3.3 Automatic creation of routing rules 20

3.4 Installation of applications 22

3.5 Consuming of services 23

3.6 Contracting 23

3.7 Usage Control 27

3.8 Interoperability with third party data initiatives 28

4 The IDS Information Model 30

4.1 DCAT 33

4.2 ODRL 36

4.3 DCMI 37

5 The TRUSTS Information Model 39

5.1 Asset 41

5.2 Configuration 45

5.3 Connector 47

5.4 Node 48

5.5 Contracts 49

5.6 External Sources 51

6 Conclusions and Next Actions 53

7 References 54

List of Figures

Figure 1: Routing 22
Figure 2: Taxonomy of datamarkets 31
Figure 3: IDS1, the three representations of the IDS-IM 32
Figure 4: DCAT 36
Figure 5: ODRL 37
Figure 6: DCMI 38
Figure 7: Dublin Core (DC) 40
Figure 8: Schema 41

List of Tables

Table 1: Adherence to TRUSTS GA Deliverable & Tasks Descriptions 14
Table 2: Metadata entities and Statements in the KG for use by the contracting system 25
Table 3: Statements included in contracts that must be acted upon by other components 26
Table 4: Metadata entities and Statements 28
Table 5: Ontos 33
Table 6: Asset, Classes of entities 42
Table 7: Asset, Relationship between entities 43
Table 8: Asset, Properties of entities 43
Table 9: Configuration: Classes of entities 46
Table 10: Configuration: Relationships between of entities 47
Table 11: Configuration: Properties of entities 47
Table 12: Connector: Classes of entities 48
Table 13: Connector: Relationship between entities 48
Table 14: Connector: Properties of entities 49
Table 15: Node: Classes of entities 49
Table 16: Node: Relationship between entities 49
Table 17: Contracts: Classes of entities 50
Table 18: Contracts: Relationships between entities 51
Table 19: Contracts: Properties of entities 51
Table 20: External Sources: Classes of entities 52
Table 21: External Sources: Relationships between entities 52
Table 22: External Sources: Properties of entities 53
Table 23: Use of namespaces 39

Glossary of terms and abbreviations used

Abbreviation / Term	Description
EOSC	European Open Science Cloud
OWL	Web Ontology Language
RoD	Registry of Datamarkets
RDF	Resource Description Framework
KG	Knowledge Graph
IDS	International Data Spaces
IDS-IM	International Data Spaces Information Model
DMA	Data Market Austria

1 Executive Summary

The TRUSTS project aims to develop a platform for trading data and data services in a trustworthy and reliable manner, which will enable a data economy in which privacy and security are at the forefront. This platform will consist of a set of nodes, each operated by a different organization which will remain in full control of their data assets. The nodes, in turn, will be executing a set of common components which will allow for transactions that will strictly adhere to any contractual agreements between trading parties, will be fully auditable, and will enable a set of innovative privacy-preserving and data-ownership respecting business models. It will result in enhancements to the European data economy, by enabling new types of transactions to take place, and open the door for restricted and private data to be monetized with strict adherence to GDPR and other relevant regulations.

The architectural design of the TRUSTS platform requires the orchestration of different components, which in turns necessitates the exchange of information in an unambiguous and consistent manner. This is especially important since the TRUSTS platform has as its objectives the interoperability of different existing data infrastructures, some of which are operated by the project partners or their customers, and some of which are operated by third parties. In particular, metadata about assets, computing resources, participants and policies has to be exchanged for the platform to satisfy its functional requirements. The collection of this metadata will be termed the TRUSTS Knowledge Graph, and the specific organization of said graph is termed the TRUSTS Information Model.

During the first 18 months of this project, the experience of the different partners was put together to refine functional and architectural requirements and to put together a first prototypical implementation of the TRUSTS platform. From these experiences, the needs for metadata exchange were further specified and put to test, in particular those which are specific of the technologies that different partners bring into the platform. Furthermore, the continued maturation of metadata standards within the IDS and the global metadata management and semantic web communities, has provided a solid foundation for the definition of the TRUSTS Information Model. This combination of practical hands-on experience in connecting the different existing components, and the contributions from academic and standardization work is what serves as background for this deliverable.

This document reports on the different uses that will be done of the TRUSTS Knowledge Graph, the specific metadata requirements of each of these as well as related metadata schemata. Finally, it provides the first version of the TRUSTS Information Model, which is to guide the implementation of mechanisms of interaction between components, and of interoperability with external data providers.

In brief, the TRUSTS Information Model is a formalization of the data-trading domain in which TRUSTS is expected to play central role. It provides concise and actionable definitions of basic notions such as Dataset, Application, Service, Contract, Node, Participant which constitute the everyday vocabulary of the project, in particular of the implementation teams. The exact properties that said entities can have, as well as the relationships that can occur between them are also described in detail. These definitions have been informed by the specific requirements of the platform, and this document provides a guide on what the effect of these definitions can be expected to be in the different components.

It is envisioned that throughout the remainder of the project, this document will act as a reference during implementation of the software artefacts necessary for the interconnection of the different TRUSTS platform components. With this guide, the distribution of work across the different teams of the project will be facilitated, and future developments (including that done by third parties wishing to interact with the Platform) will greatly benefit from clear and well-defined semantics. Finally, it is expected that the

public nature of this deliverable, along with the proper dissemination activities, will help communities and projects facing similar challenges to reuse the solutions proposed so far.

Here we report on the first version the TRUSTS Knowledge Graph and its corresponding Information Model. This first version will be subject to test in the upcoming months and will also be discussed in the wider metadata and semantic web communities. The result of these processes, as well as concrete metrics on the performance of the Knowledge Graph will be presented in the second version of this deliverable towards the end of the project lifetime.

2 Introduction

Metadata is understood as a set of statements about entities in a system. In the TRUSTS platform, the relevant entities are, for example, assets, participants, nodes, contracts or topics. Statements about these are included in the metadata in order to enable several functionalities, and to make sure that the components responsible for them have a common understanding of the state of the entities in the platform. In any given time, the collection of all metadata in the TRUSTS platform constitutes a Knowledge Graph[1], since it satisfies the following conditions:

i) Entities of the TRUSTS platform correspond to nodes in the graph, each of which has a unique and fixed identifier

ii) Relationships between entities, each of which has agreed-upon semantics that are understood by all components producing and consuming the metadata, correspond to the edges of the graph.

iii) Statements about entities are represented by sets of edges, which can be stored according to a well-defined schema whose organization allows properly equipped systems to make queries in an efficient manner to satisfy their respective functionalities.

iv) It is possible to link some of the entities with those of external sources of knowledge in order to enrich metadata and enable further operations on it.

The choices of what constitutes metadata, how it is to be represented and transmitted, how identifiers are going to be assigned, and what are the specific semantics of the different relations, are all informed by the functional requirements of the TRUSTS platform. These definitions, and subsequent operations on metadata are not made for the purpose of organization in itself, but rather, to enable the TRUSTS platform to carry out the tasks that it is required to, and to reduce ambiguity or redundancy of the information exchanged for these purposes. The operations of the different components that constitute the TRUSTS platform are to be parametrized using the TRUSTS Knowledge Graph, and any change of state of the platform which is to have effects on other components is to be reflected in the graph as well.

Since the topic of metadata management is one that is encountered in many situations, it is no surprise that a large amount of work has been done in the past towards definitions of metadata schemata, their precise semantics, and the technical procedures surrounding them. In particular, as part of the International Data Spaces initiative, a comprehensive metadata model was developed, called the IDS Information Model (IDS-IM). This model, in turn, builds upon the experiences of several decades of the metadata management [4,5], policy representation[3] and archival communities[6]. In the TRUSTS project, additions and modifications to this model are proposed.

This document describes the first version of the metadata model to be used in the TRUSTS platform. Since the choice of the model is driven by the functional requirements and the architectural features of the platform, these are analyzed in terms of their metadata requirements. During this analysis, the suitability of the different components of the IDS-IM is also considered, and points for improvement or extension are identified. With these, an overall picture is presented of how the Knowledge Graph containing the metadata of the TRUSTS platform is to be organized, updated, and exploited by different components. This final, summarizing exposition builds heavily upon many aspects of the IDS-IM and other initiatives, of which a summary is also included in order to make this a self-contained reference document for the development of the first versions of the TRUSTS platform.

2.1 Definitions

TRUSTS platform

The set of interconnected nodes, and the components running within them, that support the functional requirements of the TRUSTS project.

TRUSTS resources

The set of entities whose description is relevant for the operation of the platform. In particular, this includes assets (datasets, applications, services), nodes, deployed components, organizations. The phrase "TRUSTS resource" refers to the actual, concrete entity.

Metadata

Any description about resources in the TRUSTS platform. This document is devoted to describing which of these descriptions are relevant and how they are organized and transmitted.

Metadata schema

A specification of how metadata for one or more classes of resources is to be recorded. It enumerates the list of metadata for a given resource and the type that said metadata should have (e.g. string, integer, controlled vocabulary).

Controlled vocabulary

An organized set of concepts with fixed identifiers, each of which can have one or more labels for human consumption. In this document, a controlled vocabulary is assumed to be conformant to the SKOS[2] specification.

Information Model

A specification of the different classes of TRUSTS resources that are to be considered, the metadata schemata that are to be adopted for each of them, and the relations that can hold among them. An information model specifies a set of valid resources, statements about said resources and an interpretation of said statements that can be operationalized. In this document, we consider an information model to be described using the Ontology Modeling Language OWL[3] alongside a natural language description that is sufficient for interpreting, constructing and processing statements that conform to this specification.

Knowledge Graph

A graph that contains nodes corresponding to TRUSTS resources and that i) conforms to a given Information Model, ii) represents the state of a set of resources, iii) can be stored and queried according to well-defined methods, and iv) can be linked with other such graphs in order to enrich the meaning of the statements encoded in its edges.

[2] https://www.w3.org/TR/skos-reference/ Last accessed June 22, 2021
[3] https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/ Last accessed June 22, 2021

2.2 Mapping Projects' Outputs

Table 1: Adherence to TRUSTS GA Deliverable & Tasks Descriptions

TRUSTS Task		Respective Document Chapter(s)	Justification
T3.4 *Data Governance : Metadata, Lineage and Semantic Layer*	*This task provides one of the backbones of TRUSTS to ensure a clear data governance model in the form of a TRUSTS Knowledge Graph that includes models (taxonomies, ontologies), metadata of all TRUSTS objects (data, services, tools, users, etc.), and lineage information (the information about provenance as well as the lifecycle of a dataset, service or software tool et al.) that can be used for interoperability (T3.3), Smart Contracts (T3.2), Search and Brokerage (T3.5 and 3.6) and above. This Knowledge Graph will be realised in the form of a semantic layer for TRUSTS that connects all objects in the system, and provides context and meaning for TRUSTS mechanisms and features.*	*Whole Deliverable*	*This document is one of the main outputs of T3.4, as it is here where the metadata layer is formally specified.*
T3.3 *Data marketplaces interoperability solutions*	*Based on the findings of D2.1: Definition and analysis of the EU and worldwide data market trends and industrial needs for growth, and by analysing existing interfaces and standards, and even developing new relevant standards (see T7.4 Standardisation), the*	*Chapter 3, Section 8. Chapter 5, Sections: 1 and 6*	*Semantic Interoperability is an important aspect of interoperability in general. This document contains the solutions, from the metadata point of view, that TRUSTS proposes for interoperability with external sources. The research into the requirements of interoperability undertaken*

	interoperability solution for TRUSTS will be designed in this task. This means the definition of interfaces to ensure interoperability with other industrial data marketplaces. In addition interoperability solutions with the European Open Science Cloud (EOSC) will be evaluated and implemented where possible.		*in T3.3 have greatly informed this document.*
***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 release of D2.4A, 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 en*	*Chapter 3, Sections 1-7*	*Metadata exchange is necessary for the integration of the different components that constitute the platform. Furthermore, the TRUSTS-IM proposed in this document is greatly informed by the implementation activities taken so far in the project.*
TRUSTS Deliverable			

D3.7 Data Governance, TRUSTS Knowledge Graph I

The deliverable contains the definition and specification of the Semantic layer, its utilized taxonomies, ontologies and metadata schemata. In addition, it elaborates on how the semantic layer supports the functionality of the TRUSTS platform. D3.4B is a revised and updated version of D3.4A, covering the final state of semantic technologies in TRUSTS (including all related software components).

2.3 Deliverable Overview and Report Structure

The contents of this document are informed by the developments that have taken place throughout the project. In particular, Work Package 2 has gathered functional and architecture requirements which, in turn, influence the different types of metadata that need be exchanged within the platform. Task 3.3 has been the venue for ample discussion and experimentation with respect to interoperability, the results of which have been included into this deliverable. In particular, the contents of Chapter 5 section 6; and of Chapter 3 section 8, are the result of work from T3.3. Task 3.5 which is in charge of the integration of the platform has been the venue for the technical exploration which has informed Chapter 3 sections 1 to 5. Finally, the contents of Chapter 3 sections 7 and 8 is informed by the work of task 3.2.

Chapter 3 presents the different uses that will be done of the TRUSTS knowledge graph. For this, different functionalities are explained in detail and in accordance with the architecture of the TRUSTS platform, and their respective metadata requirements are identified.

Chapter 4 presents a summary of the metadata models on which the TRUSTS metadata model is built upon. Special focus is given to the IDS-IM which serves as the basis of this work. After this presentation, limitations that have been identified are discussed.

Chapter 5 presents the enhancements and modifications that have been made to the IDS-IM. This chapter and the previous should constitute a reference for the development of the first versions of the TRUSTS platform.

Chapter 6 discusses the different technical developments required for creating, updating and exploiting the TRUSTS Knowledge Graph.

3 Use of a Knowledge Graph in TRUSTS

In the frame of the TRUSTS project, there are several tasks that will be accomplished using the metadata contained in the Knowledge Graph (KG). In general, whenever a component requires information generated by another component it will be able to query the KG for this information. In order to decide what the schema (Information Model) and contents of this KG should be, we first analyze what uses of it should be made, in a sense, to gather the functional requirements. With this in hand and based on the existing IDS-IM described in the next chapter, we introduce in chapter 5 the Information Model that is to sit behind the KG.

3.1 Discovering assets

Asset discovery is the process by which potential users can navigate a catalog and find assets of their interest. This is accomplished, according to the TRUSTS architecture described in D2.6, by a combination of search and recommendation systems. These systems, in turn, require a source of metadata about the available assets, a catalog. Catalogs can be interpreted as lists of assets which contain sufficient information for them to be discovered through search and recommendation, and for interested parties to acquire and subsequently access them by means of other components. It is no surprise that the catalog of assets plays a central role in the TRUSTS architecture.

Catalogs have long been the subject of metadata schemata definitions, with several of them being widely deployed. Among them one can find DCAT (part of the IDS-IM and described in detail below), DataCite[4] (mostly geared towards research data), Marc21[5] (specifically designed for libraries) and ISAD-G[6] (for cataloguing archives). There are several common aspects of all such cataloguing schemata which are also relevant for the TRUSTS platform.

Multi-dimensionality.
Catalogs are best understood as means for searching among collections of assets. However, the search itself is not the ultimate goal but, rather, the access and (in the case of commercial settings) purchase of assets. For this reason, cataloguing standards usually include references to out-of-catalogue entities (e.g., shelves in a library, providers in a supermarket, etc.) of a variety of natures. The inclusion of such holistic views in catalogs is operationally exploited by many components, and it is thus no surprise that the asset catalog plays a central role in the TRUST KG. Additionally, this multi-dimensional description of assets is essential to realize the FAIR principles.

[4] https://schema.datacite.org/, accessed June 2021
[5] https://www.loc.gov/marc/bibliographic/, accessed June 2021
[6] https://www.ica.org/en/isadg-general-international-standard-archival-description-second-edition, accessed June 2021

Use of controlled vocabularies for properties.

All schemata are collections of predefined properties. In particular, catalog schemata provide a set of properties for catalogs, catalog entries, groups of entries, etc. These properties come from a controlled vocabulary, in the sense that a fixed, well-known, immutable set of properties can be assigned to the different types of entities. Since there are properties which are present in several cataloguing schemata (such as the title of an item, or the name of the author), it is not uncommon for a standard set of properties to be used by more than one schema. For example, the metadata properties of the Dublin Core Metadata Initiative (DCMI, described in detail below) are used by several different schemata. The IDS-IM is one such schema, and thus the TRUSTS-IM will also make use of these.

Hierarchical organization.

The different entities which make up a catalog are often organized into hierarchies. For example, a catalog can contain other sub-catalogs, each of which contains a set of records, and each of which contains a set of concrete elements (e.g., physical books in a library). While this hierarchy can be flexible, in the sense of an entity having more than one parent, it is very often assumed that it is, at least locally, a strict hierarchy. This assumption is often exploited in user interfaces, to help querying large collections by diving into more and more specific sets of entities.

Use of controlled vocabularies as values for some properties.

The set of all the entries of a catalog can be structured in many different ways, one of the hierarchical one mentioned above. As the breadth of the catalog grows, the structure can be enriched so that, for example, it is possible to distinguish the entries belonging to a certain category. Likewise, structures distinguishing entities by their access method, or by their pricing model, can also be exploited by search interfaces to reduce human effort in search. These structures are best exploited when they are defined by the use of controlled vocabularies in the values of some of the entry's properties. Examples of this are subject headings (such as the Library of Congress Subject Headings[7] that one can use to find books in a library system), or the standardized country names as defined in the ISO-3166 standard.

Multilingualism.

Catalogs of assets contain relatively small amounts of information, as compared to the assets they themselves catalog. This makes it relatively easy to translate catalog entries, which is also particularly useful in the case where assets are not necessarily tied to a specific natural language. While multilingualism is inherited from any controlled vocabularies used, some free-text fields in catalogs must also support values in different languages. In the case of the TRUSTS platform, which will have nodes distributed across Europe and belonging to organizations from different economic activities, there should be no technical impediments to multilingual description of assets.

[7] https://www.loc.gov/aba/publications/FreeLCSH/freelcsh.html, accessed June 2021

The properties mentioned above are desirable in all catalogues for supporting asset discovery. Catalogues possessing said properties support at least the following three important properties:

1. Assigning unique and immutable identifiers to assets.
2. Categorization of assets
3. A distinction of different concreteness levels:
    a. The asset which is catalogued, categorized and described, which is the subject of discovery.
    b. Specific presentations of an asset, for example, different formats or different pricing models, which are the subjects of configuration for, respectively, the access and contracting mechanisms. These specific presentations can be related to inherent properties, such as hashes, which help distinguish each of them and assert the authenticity of an instance in contractual procedures.
    c. Specific instances of an asset, those which are subjects of contracts. Given the nature of digital assets, many copies of an asset can be made, and each purchased individually, but adequate usage control requires that each be assigned an individual identifier, related to an individual contract instance.

In the case of TRUSTS, there are particular relations of discoverability of assets with other functional requirements, as well as specific discoverability needs of the platform. The functional requirements described in D2.2 require different actions to be taken upon assets: they must be searchable, accessible and recommendable, they must be subjects of contracts and reviews, access to them must be metered, and they must be matchable with other assets. Each of these dimensions to one asset impose different properties on the cataloguing schema, especially since many of them have effects on others (e.g., a user might want to search only for assets within a certain price range or deployed in one particular geography). In TRUSTS, the hierarchical organization is also relevant because it shall reflect the federated nature of the platform. Namely, there will be one central catalog of all assets traded which will be the concatenation of the catalogs belonging to each organization. This means that all catalogs must include information about the provider of an asset, as well as metadata about them. In particular, it is necessary to hold information regarding the regulatory framework in which a given asset provided is embedded (which might or may not be linked to geographical location), as such knowledge will empower consumers to make more informed decisions.

Furthermore, the assets traded on the TRUSTS platform present properties different from those in other catalogs, which might be relevant for discovery. In particular, applications require specific infrastructure to be executed on, which might be a factor for deciding on purchase. Likewise, services require specific access mechanisms to be present on the consumer, which must be informed up-front when searching. All of these technical details must thus be accommodated into the catalogue in order for them to play a role in search and recommendation applications. Additionally, assets in trusts might be catalogued along with a “sample”, to increase trust by the consumer and ensure correct technical alignment. Finally, detailed dataset description is necessary in order to enable dataset-to-service recommendations. All of these

metadata are to be stored in the TRUSTS-KG in order to ensure discoverability and adherence to the FAIR (findable, accessible, interoperable, reusable) principles, which require a more precise description of assets, beyond the merely catalographic[10].

3.2 Node operation

The TRUSTS platform is a distributed set of nodes, each operated by a different organization, which communicate among them to offer and consume assets. Among these nodes there is the Central Node which contains a copy of all the metadata about assets offered by the other nodes.

Setting up a node requires an initial exchange of information between the new node and the central node. While not all of it needs to be included into the TRUSTS KG, that which is exploited during the operation of the node, particularly by different components or by different nodes, will be included as long as it must not be kept private. Among this is the list of image registries that are to be accessible from the new node to install TRUSTS components as well as applications acquired later. Likewise, the names of the images and code repositories that the node operator must pull to deploy their node must be kept in a centralized location where it can be kept up to date. Likewise, the location of the central node (IP address) must be kept up to date in the KG, as it will be leveraged by several components, chief among them, the node's Trusted Connector.

The Trusted Connector [7] is a software component which acts as gateway to the different components inside a node. It provides a series of endpoints that different components can connect to in order to send information to, or receive information from, other endpoints. These communications are all encrypted and signed, and the Trusted Connector is in charge of verifying the signatures, and their validity in a dynamic manner. Additionally, the Trusted Connector provides an enterprise bus (based on Apache Camel) that is used to distribute messages, enabling complex workflows involving several components. Finally, the Trusted Connector provides functionalities for managing the different containers running on the node.

The operation of each node requires a set of metadata to be recorded and accessed by different components. A given node must publicize, through the central node, the details regarding access to several of the services it provides. The details of these can be consulted in D2.6, but it suffices to say here that the endpoints (ports) which the node's Trusted Connector exposes must be known to all other participants. Likewise, in order to enable adequate routing of data within the node, the internal names of the different components of the node shall be known. In particular, if a node is offering a service, it must be broadcasted that this service is deployed, within the node, with a particular name. This information, along with the domain name of the node, will be further propagated to all other nodes.

With this metadata, a node A wishing to access an endpoint E from service X's port P at node B will do so by doing HTTP requests to the following address.

http://{connector_A_localname}:{connector_port}/B/X/P/E

These variables (B, X, P, E) are to be retrieved by node A from the centralized metadata repository in order to construct the above-mentioned URL. The local connector's name and its port need, in contrast, not be broadcasted, as they are relevant only to the users of node A from within the same organization.

3.3 Automatic creation of routing rules

When an application, service, or component (hereafter resource) is deployed on a TRUSTS node, routing in the Trusted Connector should be configured to make it accessible. In more detail, as depicted in Figure **Routing**, any communication that has to occur between a TRUSTS component and a third party resource which resides outside of TRUSTS, or another TRUSTS resource which is hosted in another TRUSTS node, should be realized through the Trusted Connector. To this end, for each such communication, a specific route should be defined, according to the Apache Camel[8] Consumer – Provider model followed by the IDS Trusted Connector[9]. This route requires two rules to be specified, one in the node initiating the communication (Consumer) and one in the node receiving and responding to said communication (Provider). For these to be established, it is necessary to know the following details:

(1) The hostname of the Trusted Connector that acts as provider

(2) The port through which this Trusted Connector is accessible from the internet

(3) The name inside the Provider node by which the resource is accessible

(4) The name by which the resource is known to the rest of the TRUSTS platform.

(5) The port that the resource exposes to the Trusted Connector in its node (the provider node)

(6) The REST API endpoint (route) that the consumer might want to access in the resource.

Additionally, components in the consumer's organization consuming the resource (e.g., an application running in a browser, or an existing component acting as an HTTP client) must have access to **(7)** the consumer node's Trusted Connector name in the organization's network, and **(8)** the port number for receiving local requests.

With this in hand, the consuming component will make a request to the local Trusted Connector, which forwards it to the provider's Trusted Connector which, in turn, will forward it to the resource. This is depicted in more detail in Figure **Routing**, where the origin of each of the 8 metadata mentioned above is also specified. Since these metadata are to be used by both provider and receiver nodes, and by several components within each, they must be accessible through the TRUSTS KG. Some of these metadata can

[8] https://camel.apache.org/

[9] https://github.com/International-Data-Spaces-Association/DataspaceConnector/wiki/Using-Camel

be used to configure the routes as soon as the providing component is installed, while others are to be queried dynamically when the request is about to take place. In particular:

1. When a new resource is deployed in the provider node, it needs to be available to other nodes. Its internal hostname and port must be available to the node's Trusted Connector. This Trusted Connector will then create a rule that guarantees that any requests it receives are forwarded to this specific component.
2. When access to a resource is purchased by the consumer. The consumer's Trusted Connector must create a rule that forwards all requests it receives in its local access port to the remote node serving the resource.

The origin of this metadata can be varied. Some can be input directly by the provider when onboarding a Service or Application, and some can be extracted automatically from the programmatic descriptions of said assets, respectively a docker-compose[10] file, or an OpenAPI3.0[11] specification, as detailed below.

.

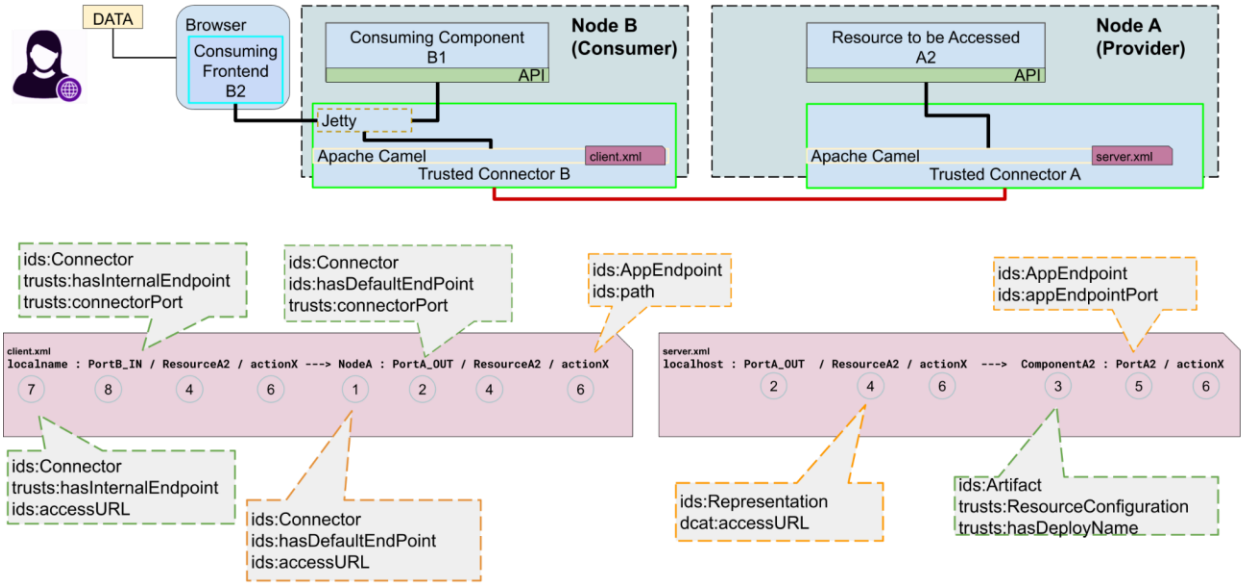

Figure 1: Routing

Sending messages between Trusted Connectors requires configuration in both the receiving and emitting end. This configuration comes in the form of Apache Camel routes, and the creation of these rules necessitates access to metadata. Pictured are the different metadata properties used for configuration. Boxes with green borders represent properties not currently part of IDS-IM. Circled numbers correspond to the metadata description in the text.

[10] https://docs.docker.com/compose/compose-file/, last accessed June 25, 2021
[11] https://swagger.io/specification/, last accessed June 25, 2021

3.4 Installation of applications

One of the type of assets available for purchase through the TRUSTS platform are applications: software artefacts that are transferred to and executed in the computing infrastructure of the consumer. Installation of applications means downloading a container image by the consumer and setting up the corresponding routing and configuration parameters. After installation, the application can be accessed from within the consumer's internal network, and every access to the application is done through the Trusted Connector, allowing for detailed metering of access and enforcing of contract provisions.

When a participant (either a corporate entity or a user) wants to work with a bought application or service (i.e., App) the necessary execution environment has to be setup up by a system administrator (SA) so that the App can be installed and run. Therefore, the system administrator needs to provide either a physical or virtual machine with an IP address assigned and accessible from the Trusted Connector of the organization's TRUSTS node. They have to configure the Trusted Connector outbound ports so that the App will be accessible by end-users through the combination of IP address and outbound port (or a given domain and path). These ports and ip addresses are to be available to the application users.

As the App will be hidden from outside with the help of the Trusted Connector, the routing mechanism described above has to configure an outbound port in the Trusted Connector for the App to be installed. Thus, the metadata necessary for app installation is a superset of that required for the automatic creation of routes.

When the TRUSTS environment is set up, the App can be installed by downloading it from the TRUSTS Docker Registry. This implies that the consumer's node must have access to docker registry, and thus its metadata must be included in the TRUSTS KG. Subsequently, the docker container holding the actual instance of the App can either be instantiated directly with Docker or wrapped (and configured more easily) from within in a docker-compose V3[12] environment after the default configuration for the App is pulled from the TRUSTS KG and provisioned at App startup. This configuration can be extracted automatically from the docker-compose file at app on-boarding times, and includes volume names, port numbers and network names.

After being configured correctly, requests to the App get routed from the outbound port of the underlying machine via the Trusted Connector's outbound port to the App's outbound port. If the App itself needs any other Service/App (i.e., Component) to execute its tasks, respective Routes have to be configured so that the App is able to access other Components either behind the same or remote Trusted Connectors. Additionally, if the App has certain requirements regarding certain protocols or headers to be sent along requests, Routes have to take this information into account as well, and thus it has to be available in the TRUSTS KG.

Importantly, when an app is downloaded, the Usage Control mechanism in the consumer's node must also be ready to process access requests to it from within the consumer's network. Thus, metadata regarding contracting must be made available during installation. This will enable the Usage Control component to react to an access request to the application by sending a query to the node's Smart Contract Execution

[12] https://docs.docker.com/compose/compose-file/compose-file-v3/, accessed June 2021

with the ID of the requesting user, and the ID of the contract with which this app was installed and receive authorization information.

Other configurations might be necessary inside the consumer's node, but even if automatable, have no effect on, or require information from, other TRUSTS nodes. Namely, as the Trusted Connector will wrap Apps and other Components potentially sharing same names, it is necessary to distinguish Components by assigning them to separate (Docker-) networks and update the Routes, respectively. Likewise, if the App itself needs file system access via so called Docker volumes, the system administrator needs to take care of implied access restrictions or to make sure that equivalent docker options/configurations are used (e.g., Docker Named Volumes instead of Bind Volumes).

3.5 Consuming of services

Services are defined in the TRUSTS platform as software artefacts which are executed on the provider's computing infrastructure, but which are accessed by the consumer. In order to make a service available it is necessary to configure routes for this in the provider's Node, as specified in the sections above. Additionally, since services are to be subjected to contractual agreements and usage control, the consumer must also access them through their respective Trusted Connector[13], and thus routing must also be set up in their connector. This is described in detail in the section "Automatic creation of routing rules" above.

In order for services to be usable to consumers, an adequate description of their interfaces must be provided. For this, the TRUSTS KG must contain provisions for a full OpenAPI3.0[14] description to be made of every service. From said descriptions, configuration for routing can be generated, and also further catalographic metadata to aid in the discovery of the service. For example, by providing a properly annotated OpenAPI 3.0 description, a potential customer can quickly realize if a service consumes the type of data they have at hand. Finally, by providing such a description, customers can auto-generate clients for services for integrations into their applications.

3.6 Contracting

One of the key features envisioned in the TRUSTS platform is the possibility to offer assets in accordance with pre-specified contracts, and to have the compliance with them automatically and reliably verified. For this purpose, a series of contracting mechanisms will be deployed as part of the TRUSTS platform. In order to support contracting, and contract compliance verification, it is necessary to establish a common vocabulary between said mechanisms, the assets catalogue and the usage control mechanisms. In brief, the entities (assets, participants, etc) that are referred to in contracts, and the relationships between them, must be the same as in the catalogue and elsewhere in the platform. Thus, the design of the metadata schema must take into account the specific requirements of the contracting mechanisms.

[13]The provider can opt-out of this provision desired, for example, casual users that don't have a fully configured TRUSTS node at their disposal, to access their service.
[14] https://swagger.io/specification/#version-3.0.3

There are several ways to see assets in TRUSTS: e.g., a digital asset which interacts with other digital artifacts (e.g., a file to be loaded by a piece of software, or an endpoint to be accessed by a client), a catalographic entity (that is to be findable using search and recommendation), or a subject of a contract. It is the latter which occupies us in this section.
A contract is a series of statements involving an asset (or a set thereof) and two or more parties, in which a series of conditions (monetary or not) are stipulated. Importantly, these statements are to be used by a variety of actors for different tasks, which are detailed below.

We decide to encode these statements as part of the metadata, and thus include its specification in the information model, because **i)** many of these are about entities which are already contained in the metadata, and **ii)** these statements must be interpreted by different components in a consistent manner. For example, the Asset which is subject of a contract must be equated with an Asset to which access restrictions apply, or to an Asset which must be findable on the Catalog. One effective way to equate these is to use a single identifier for all of them. The same is true of the different predicates involved, for example, a purchase model as stipulated in a contract must be accompanied by corresponding behaviour in the usage control system. Since the purpose of metadata, as stated above, is for harmonized and semantically web-defined description of entities that enables consistent and reliable actions, we make the metadata as the source of truth regarding contracting purposes within the TRUSTS platform.

In the following, we describe the different uses of metadata for contracting, and its relation to other components. We incorporate a subset of the concepts important in contracting as detailed in deliverable D3.3. First, we describe the use of metadata generated for other purposes in contracts. Then we describe the use of statements originating from contracts by other components.

Table 2: Metadata entities and Statements in the KG for use by the contracting system.

Class of Entity or Type of Statement	Components using it	Purpose in Contracting
Identifier for Assets	Platform Interface, Usage Control, Recommender, Metadata Mapper, Notification Service	Identify the assets which are subject of the contract
Identifier for Agents (Users, Corporate entities)	Platform Interface, Usage Control, Recommender	Identify the Providers and Consumers in the Contract
Identifiers for Nodes	Platform Interface, Usage Control, Asset Consumer	Specifying the means of access to an asset that are covered by the contract

Type of Asset	Platform Interface, Usage Control, Recommender, Metadata Mapper, Dataflow Router	Determine what different kinds of purchase models are available
Asset is Provided by Agent	Platform Interface, Usage Control, Metadata Mapper	Specify provider in the Contract

Table 3: Statements included in contracts that must be acted upon by other components

Statement in a Contract	Consuming Component	Purpose
An Agent is providing an Asset	Usage Control	The Usage Control in the provider's node must be aware that requests for the specific asset will be incoming.
An Agent is purchasing an Asset	Usage Control	The Usage Control in the provider's node must give access to this specific Agent to this specific Asset
	Recommendation	The recommendation system can take into account this action to recommend further assets to this and other users.
A contract is valid from a given starting date	Usage Control	The Usage Control in the provider's node must take this rule into account when it receives a request for this Asset
A contract is valid until a given starting date	Usage Control	The Usage Control in the provider's node must take this rule into account when it receives a request for this Asset
A contract allows for a fixed number of access operations to an Asset	Usage Control	The Usage Control in the provider's node must take this rule into account when it receives a request for this

		Asset
A contract specifies that access to a Dataset can only be done through a specific Service	Recommendation	The recommendation system can take into account this action to recommend the service to other users acquiring this Asset.

We have shown above several pieces of information that are used by both the Contracting system and other components in the TRUSTS platform. To enable this sharing of information, and to guarantee it is consistent and well-defined interpretation by the different components, each of this metadata must be part of the TRUSTS Knowledge Graph.

The statements included in a contract, however, are much more convoluted than simply “Agent X has access to Asset Y”. In general, a contract specifies a series of Duties, Permissions and Prohibitions (collectively known as Rules), each of which acts upon an asset and a set of Parties. These Rules dictate the behaviour of the Usage Control system, and so a common and well-defined meaning of each must exist between the component for composing, presenting and signing contracts (the Contracting Mechanism) and the component in charge of enforcing them (the Usage Control mechanism). Furthermore, these rules should be represented verbatim in the human readable version of the contract, and therefore mechanisms must exist for verbal, multilingual representations of contracts. Finally, when auditing the execution of a contract, both for legal and business purposes (e.g., by the Business, Administration and Monitoring tools), this meaning must be preserved. This implies that the TRUSTS knowledge graph must include a formalized representation of the rules contained within a contract, which, in turns, necessitates the use of an expressive Information Model. It is for this reason that ODRL is subsumed into the TRUSTS information model and described in detail in chapter 4.

Importantly, contracts for specific transactions are generated from templates in which the different variables are substituted for concrete asset and agent identifiers, as well as literals such as dates for validity or integers for number of allowed accesses or currency. This templating mechanism is relevant for the metadata management because an actual contract instance inherits the rules of the template, which in turn imposes new restrictions on the descriptions of such templates, namely, that they must contain a number of unbound variables which are to be evaluated upon instantiation. The facilities for template to instance transition including this evaluation of variables is not by default supported by ODRL, so workarounds are suggested in chapter 5 section 5 to deal with this.

3.7 Usage Control

The purpose of the TRUSTS platform is the exchange of data assets between different parties, which must be carried out with a certain level of reliability and trustworthiness. This necessitates, in turn, the use of stringent Usage Control mechanisms. These will go beyond the authorization systems normally available in computing environments, to accommodate for a variety of commercial models. For example, they will allow for assets to be accessible only for a limited time period, or for a limited number of access operations. Furthermore, functional requirements (e.g. IR3 in deliverable 2.2), of the TRUSTS platform include the possibility of detailed access logs being auditable at all times, in order to provide guarantees to all parties involved in a contract. Finally, the complexity of the routing of information within a TRUSTS node is also relevant for usage control as, for example, several instances of the same service might be running on a node, and a request must be forwarded to a particular one in accordance with contractual policies.

Usage control, as envisioned in the TRUSTS platform, is carried out by a separate component that resides in every node and which must clear all access operations that the node's connector requests. For this clearance to be executed, information from the request must be compared with that contained in a ledger that enjoys the trust of all parties. This ledger is the Contracting Mechanism of the TRUSTS platform. For this comparison to be made, a consistent and well-defined model should be adapted by all involved components, so that the meaning of Agent, Asset, Access Operation, etc. is consistently interpreted by all. This is the purpose of the TRUSTS KG, and the information model on which it is based must, therefore, be prepared to handle the functional requirements of usage control.

It is envisioned that the following types of entities and statements about them dictate the behavior of the Usage Control mechanism.

Table 4: Metadata entities and Statements which are included in the Knowledge Graph for purposes other than Usage Control, but which can be leveraged by Usage Control

Class of Entity or Type of Statement	Components using it	Purpose in Usage Control
Identifier for Assets	Platform Interface, Contracting, Recommender, Metadata Mapper, Notification Service, Routing	Identify the assets that are the subject of a request
Identifier for Agents (Users, Corporate entities)	Platform Interface, Contracting, Recommender	Identify the Agents which are requesting access.
Identifier of	Contracting	Check with the Smart Contract Execution if the access

Contract		should be granted
Access details of an Asset	Platform Interface, Metadata Mapper	Determine which routes of the Connector should be triggered

3.8 Interoperability with third party data initiatives

Interoperability with third-party data initiatives is the focus of T3.3 within TRUSTS. Third-party initiatives include (i) existing external data markets, and (ii) the European Open Science Cloud (EOSC[15]). The challenges for designing an appropriate metadata schema arise from the variations in the characteristics of both different data markets and different EOSC initiatives. Neither data markets nor EOSC initiatives expose a unique and homogeneous interface to be used for data exchange. Instead, they strongly differ in both their conceptual and technological orientation. We aim to build a set of components to effectively establish interoperation with third-party data initiatives and TRUSTS:

- a “data exchange client component”: an interface for external initiatives to align their metadata with TRUSTS.
- a “data exchange TRUSTS components”: a component to route data from the aforementioned client component to the data storages of TRUSTS.
- a “registry of data markets” (RoD): a catalog listing existing data initiatives and their connection endpoints.

A potential use case is the interest of a TRUSTS customer in data assets for a given domain, where the data assets listed in TRUSTS do not fully match the requirements. In TRUSTS corporate nodes, the Data Exchange Client Component is connected to the recommender component of TRUSTS. The recommender component incorporates the metadata that has been registered by an external data initiative via the Data Exchange Client Component and makes it searchable.

In the following, we describe the metadata schemata foreseen for the establishment of these components. This encompasses (i) the metadata schema built into the RoD, (ii) EDMI (EOSC Dataset Minimum Information)[16], and (iii) the prototypical design of the metadata schema for interoperability with external datamarkets.

EOSC-related Metadata

The EOSC pilot study[17] identified in its deliverable “D6.9: Final report on Data Interoperability”[18] a common set of minimum information for future EOSC initiatives, summarized in EDMI[19]. We consider the usage of EDMI as crucial for the TRUSTS metadata schema to facilitate compliance with EOSC. EOSC has by nature a strong focus on science and research, which is reflected in EDMI. EDMI partially overlaps with the IDS-IM. It shares multiple properties such as “edmi:name” ↔ ”ids-im:title”, “edmi:description” ↔ “ids-

[15] https://eosc-portal.eu/, last accessed June 17, 2021
[16] https://eosc-edmi.github.io/, last accessed June 17, 2021
[17] https://eoscpilot.eu/, last accessed June 17, 2021
[18] https://www.eoscpilot.eu/content/d69-final-report-data-interoperability, last accessed June 17, 2021
[19] https://eosc-edmi.github.io/, last accessed June 17, 2021

im:description", or "edmi: dateCreated" ↔ "ids-im: date created". However, there are multiple properties unique to EDMI, which are required for EOSC interoperability and thus for TRUSTS. For example, EDMI has properties strongly tied to scientific referencing such as "scientificType", "citation", or "referenceCitation", but also for documenting scientific experiments, such as "variablesMeasured" or "measurementTechnique" (see Table 8: EDMI metadata for a full list of properties unique to EDMI).

Metadata required for interoperability with third-party datamarkets

The ecosystem of datamarkets consists of a plethora of platforms with significantly differing content, offerings, and technical requirements. The metadata schema for datamarket interoperability aims at identifying a set of properties shared by many or most datamarkets. The Data Exchange Client Component developed as part of T3.3 exposes this schema both in a GUI and via an API. Datamarkets interested in exchanging data assets with TRUSTS can adopt this schema. By adherence to the exposed schema, their data assets will get listed within TRUSTS appropriately. At this stage, we analyzed a set of datamarkets and selected Namara[20] and Dawex[21] to extract a set of properties relevant for them. The currently envisaged schema includes properties such as "trusts:contact" (the name of data market contact), "trusts:collectiontype" (the type of data collection, e.g. survey, questionnaire, log, etc.), or "trusts:instrument" (the method used to collect the data).

At the current stage, the metadata schema for datamarket interoperability is still in an initial stage. Interrogations with datamarket operators are in the planning, with the aim of getting further insights into the technical specifications of datamarkets as well as their willingness to adopt proposed approaches.

Registry of Data Markets

Datamarkets, but also EOSC initiatives, willing to exchange data assets with TRUSTS, communicate with the TRUSTS storages using the Data Exchange Client Component. They register the metadata of their data assets in this component. Further on, the Data Exchange TRUSTS Component harvests the so registered data in pre-defined time intervals. An additional component, the RoD (Registry of Datamarkets), serves as an address book for the Data Exchange TRUSTS Component to locate the datamarket interfaces online. It functions as an address book routing the communication between the Data Exchange TRUSTS Component and the Data Exchange Client. Furthermore, the RoD works as a central point for information related to Datamarkets and is planned to exist beyond project lifetime. It features a search engine to find datamarkets based on their attributes. The attributes represent characteristics of the respective datamarkets and are based on the taxonomy of datamarkets based on their business models compiled in the work by van de Ven[22]. This taxonomy segregates datamarkets along four domains (service, technology, organization, finance), which are further split into dimensions, e.g., "revenue model" and "pricing model" for the finance domain. Each dimension has its own set of characteristics, e.g., "freemium", "pay-per-use", "flat fee tariff", … for the dimension "pricing model" in the domain "finance". The RoD allows faceted

[20] https://marketplace.namara.io/, last accessed June 17, 2021
[21] https://www.dawex.com/en/, last accessed June 17, 2021
[22] van de Ven, M.R. (2020). Creating a Taxonomy of Business Models for Data Marketplaces. Master Thesis, TU Delft Technology, Policy and Management.

search along those domains, dimensions, and characteristics. For each datamarket, it shows which characteristics are fulfilled (see Figure 2: Taxonomy_of_datamarkets).

The TRUSTS metadata schema needs to reflect the requirements of the RoD. Therefore, we envision the properties listed in Table 8: RoD properties for inclusion of the TRUSTS metadata schema. It mirrors the attributes of the taxonomy of datamarkets.

	Dimension	Characteristics					
Service domain	Value proposition	Easy data access and/or tooling	Secure data sharing	High quality and unique data	All services in a single platform		
	Enterprise data marketplace	Yes		No			
	Data processing and/or analytics tools	Yes		No			
	Marketplace participants	B2B		C2B		Any	
	Industry domain	Any data	Geo data	Financial & Alternative data	Health & Personal data	Audience data	Sensor & Mobility data
	Geographic scope	Global		Regional		Local	
	Time frame	Static	Up-to-date	(Near) real-time	Multiple		
Technology domain	Platform architecture	Centralized		Decentralized			
	Data access	API	Download	Specialized software	Multiple options		
	Data source	Self-generated	Customer provided data	Acquired data	Multiple sources		
Organization domain	Matching mechanism	One-to-one	One-to-many	Many-to-one	Many-to-Many		
	Platform sponsor	Private		Consortium		Independent	
Finance domain	Revenue model	Commissions	Subscriptions	Usage fees	Asset sales		
	Pricing model	Freemium	Pay-per-use	Flat fee tariff	Package based pricing	Multiple	
	Price discovery	Set by buyers	Negotiation	Set by marketplace provider	Set by external sellers		
	Smart contract	Yes		No			
	Payment currency	Fiat money		Cryptocurrency			

Figure 2: Taxonomy of datamarkets

The taxonomy of datamarkets based on their business models characterizes datamarkets along a set of domains, dimensions, and characteristics.

4 The IDS Information Model

The IDS Information Model[23] (IDS-IM) is the result of extensive work in the IDSA towards a unified ontology for representing exchange of digital content [2]. It is a formalization of the domain of digital content exchange that aims to have clear and unambiguous semantics, so that it can be used by a variety of

[23] https://w3id.org/idsa/core, accessed June 2021

software components developed by different organizations and distributed across several computing infrastructures. It deals mainly with data assets and data processing software, as well as associated entities of the Industrial Data Spaces (IDS) such as participants, infrastructure, components and processes. The IDS-IM details out and formally defines said entities in order to enable components to share, search for, and reason upon their structured meta-data descriptions

This formalization is represented in three separate but interrelated ways (See **Figure IDS1**). The first, called Conceptual Representations is aimed at human consumers and provides a detailed explanation of the formalization, an analysis of its origins and examples of its uses. The second, called Declarative Representation is a large set of RDF statements in OWL that constitute the authoritative formalization of the domain. It is accompanied by further RDF statements that comply with the SHACL and SKOS ontologies (among others) which enables its consumptions by both humans and machines for a variety of tasks. Finally, the Programmatic Representation is a translation of the classes, as defined in OWL, into classes, as defined in an object-oriented programming language. In particular, a Java implementation[24] exists that is automatically updated whenever the RDF statements are amended. The Declarative Representation is considered authoritative because of its clear semantics and its adherence to standards which, together, allow for the IDS-IM to be linked and combined with other ontologies. Since the TRUSTS-IM described in this document is, in effect, an extension of the IDS-IM, we consider only this Declarative Representation and refer to it simply as IDS-IM.

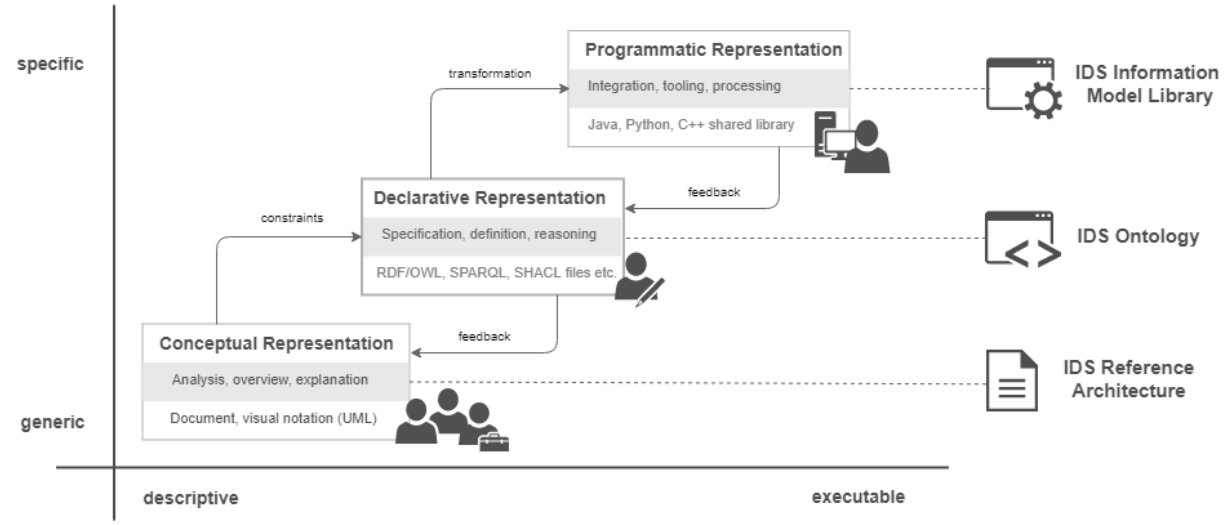

Figure 3: IDS1, the three representations of the IDS-IM. Retrieved from https://international-data-spaces-association.github.io/InformationModel/docs/index.html on 2021.06.14

The ontology which constitutes the core of the IDS-IM consists of 220 OWL classes, 84 data properties and 199 object properties. These are defined and maintained directly by the Information Model sub-group (SWG4) of the IDSA. However, the versatility and expressiveness of the IDS-IM lies in the fact that it is

[24] https://github.com/International-Data-Spaces-Association/InformationModel

linked with several other ontologies. These are listed in **Table 5: Ontos**, and the most important ones are described in detail in the sections below. Additionally, a series of well-known controlled vocabularies is also part of the IDS-IM, specifically for the specification of data properties (in the RDF sense) of the different entities described.

Table 5: Ontos: Other ontologies are linked to the IDS-IM

Ontology	Number of Linked Classes
DCAT	11
ODRL	26
Data Cube	15
ProvO	31
vcard	63
OWL-Time	20
Organization Ontology	10
Foaf	13

The different classes of the IDS-IM and linked ontologies are divided into 7 facets, which are rough groups of classes that deal with related types of entities: Resources (subsuming Data and Applications), Infrastructure, Participants, Regulations, Interactions and Maintenance. Among these, the Resource facet is central to the IDS-IM, as it is used to describe the different assets that are to be exchanged. Each resource is described according to three different views: Commoditization (relating to its quality of being tradable), Communication (relating to its quality of being transmissible) and Content (relating to its quality of being catalogued, discovered and consumed).

It is important to note that the IDS is not intended to be a data market. It is therefore of no surprise that the IDS-IM is not completely suitable for the datamarket case. However, it does accommodate for expressing information required for cataloguing assets, as well as access policies to them. These functionalities of the IDS-IM come from the inclusion of the DCAT and ODRL ontologies respectively. Likewise, there are a series or classes and properties which are meant for the operation of Data Spaces, namely those which refer to Components, Connectors etc. which are also leveraged by TRUSTS.

One outstanding feature of the IDS-IM is the notion of messages among connectors, which defines a data format by which changes to the metadata can be transmitted from one node to another. For example, when new assets become available in one node, it can send a message notifying this fact to a centralized

metadata store. Likewise, when a node becomes unavailable for some reason, this change of state can be notified to other nodes via a standardized messaging format. This messaging format, combined with the explicit semantics of the IDS-IM ontology definition, allows for consistent communication between different components. In TRUSTS, this messaging format will be exploited, and components adapted to send and receive such messages when suitable (e.g. catalog interfaces), which will allow for seamless interoperation with other IDS deployments.

4.1 DCAT

Data Catalogue Vocabulary (DCAT) is a specification[25] developed by W3C, in the context of government data catalogs such as data.gov and data.gov.uk but is also applicable and has been used in other contexts. It provides an RDF vocabulary to facilitate interoperability between data catalogs published on the Web. It enables a publisher to describe datasets and data services in a catalog, using a standardized model and vocabulary that facilitates the consumption and aggregation of metadata from multiple catalogs. Thus, discoverability of datasets and data services can be increased. Additionally, it allows a decentralized approach to publishing data catalogs. Federated search for datasets across catalogs in multiple sites is also possible using the same query mechanism and structure. Aggregated DCAT metadata can serve as a manifest file as part of the digital preservation process.

Complementary vocabularies can be used together with DCAT to provide more detailed format-specific information, for instance properties from the VOID vocabulary can be used within DCAT to express various statistics about a dataset if that dataset is in RDF format.

A data catalog conforms to DCAT if:

- Access to data is organized into datasets, distributions and data-services.
- An RDF description of the catalog itself, the corresponding catalogued resources and distributions is available
- The contents of all metadata fields that are held in the catalog and that contain data about the catalog itself, the corresponding catalogued resources and distributions are included in this RDF description and are expressed using the appropriate classes and properties from DCAT, except where nos such class or property exists
- All classes and properties defined in DCAT are used in a way consistent with the semantics declared on the DCAT specification.

A DCAT profile is a specification for a data catalog that adds additional constraints to DCAT. A data catalog that conforms to the profile also conforms to DCAT. Additional constraints in a profile may include

- Cardinality constraints
- Sub-classes and sub-properties of the standard DCAT classes and properties
- Classes and properties for additional metadata fields not covered in DCAT vocabulary specification
- Controlled vocabularies or URI sets as acceptable values for properties
- Requirements for specific access mechanisms (RDF syntaxes, protocols) to the catalog's RDF description

[25] https://www.w3.org/TR/vocab-dcat/, accessed June 2021

Some of the DCAT profiles are:

- DCAT-AP[26]: The DCAT application profile for data portals in Europe. There exist also a number of regional specialized profiles like:
  - DCAT-AP_IT[27] (Italian)
  - DCAT-AP.de[28] (German)
  - DCAT-AP-SE[29] (Swedish)
- GeoDCAT-AP[30]: Geospatial profile
- StatDCAT-AP[31]: Statistical profile

[26] https://joinup.ec.europa.eu/solution/dcat-application-profile-data-portals-europe, accessed June 2021
[27] https://docs.italia.it/italia/daf/linee-guida-cataloghi-dati-dcat-ap-it/it/stabile/dcat-ap_it.html, accessed June 2021
[28] https://dcat-ap.de/def/, accessed June 2021
[29] https://lankadedata.se/spec/DCAT-AP-SE, accessed June 2021
[30] https://joinup.ec.europa.eu/solution/geodcat-application-profile-data-portals-europe, accessed June 2021
[31] https://joinup.ec.europa.eu/solution/statdcat-application-profile-data-portals-europe, accessed June 2021

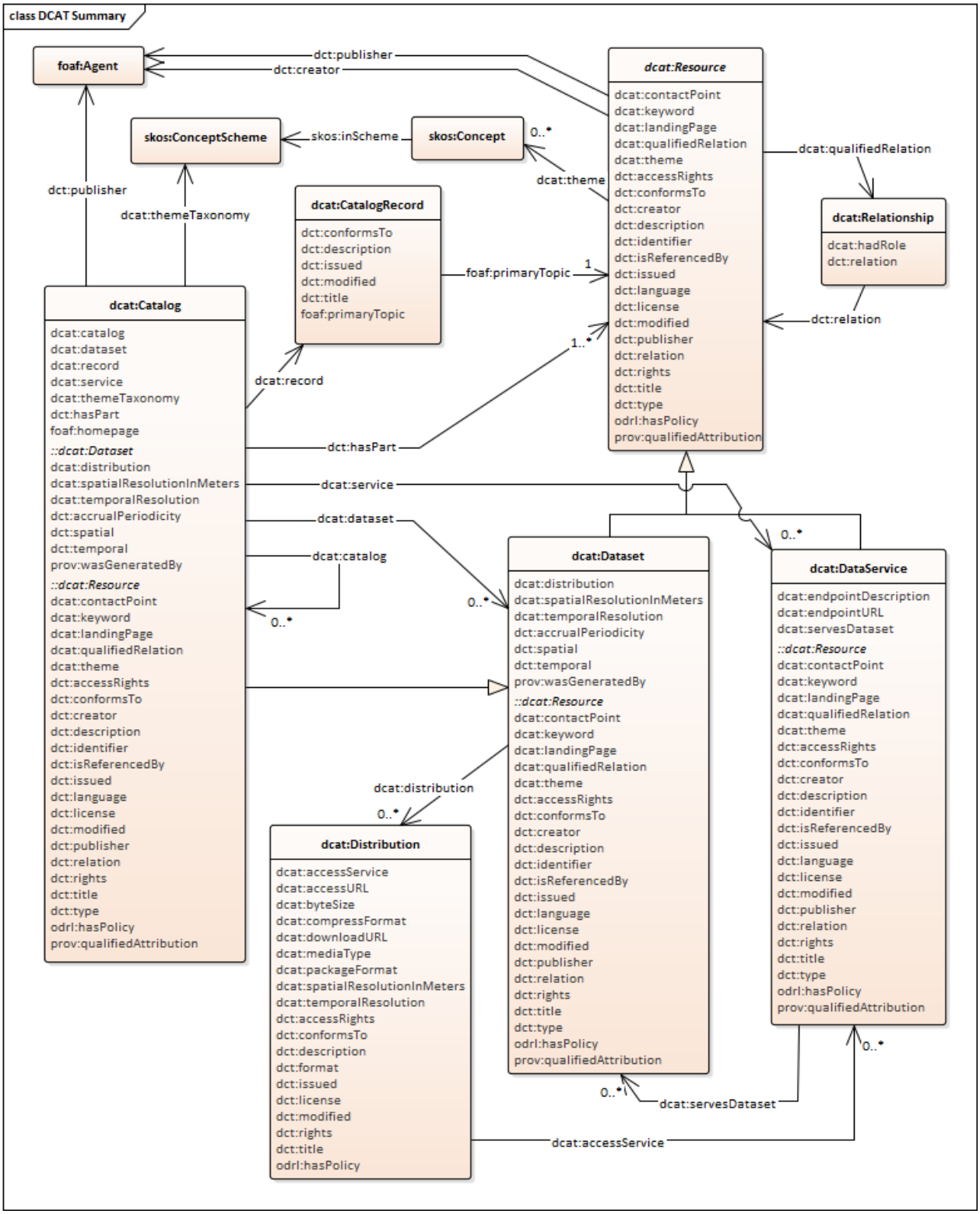

Figure 4: DCAT. An overview of the DCAT model, showing the classes of resources that can be members of a catalog, and the relationships between them. Image from https://www.w3.org/TR/vocab-dcat-2/ accessed on 2021-06-20

4.2 ODRL

The Open Digital Rights Language (ODRL[32]) is a policy express language that provides a flexible and interoperable information model, vocabulary and encoding mechanisms for representing statements about the usage of content and services. The ODRL Information Model describes the underlying concepts, entities and relationships that form the foundational basis for the semantics of the ODRL policies.

Policies are used to represent permitted and prohibited actions over a certain asset, as well as the obligations required to be met by stakeholders. In addition, policies may be limited by constraints (e.g. temporal or spatial) and duties (e.g. payments) may be imposed on permissions.

The ODRL Information Model represents Policies that express Permissions, Prohibitions and Duties related to the usage of Asset resources. The Information Model explicitly expresses what is allowed and what is not allowed by the Policy, as well as other terms, requirements, and parties involved. The aim of the ODRL Information Model is to enable flexible Policy expressions by allowing the policy author to include as much, or as little, detail in the Policies.

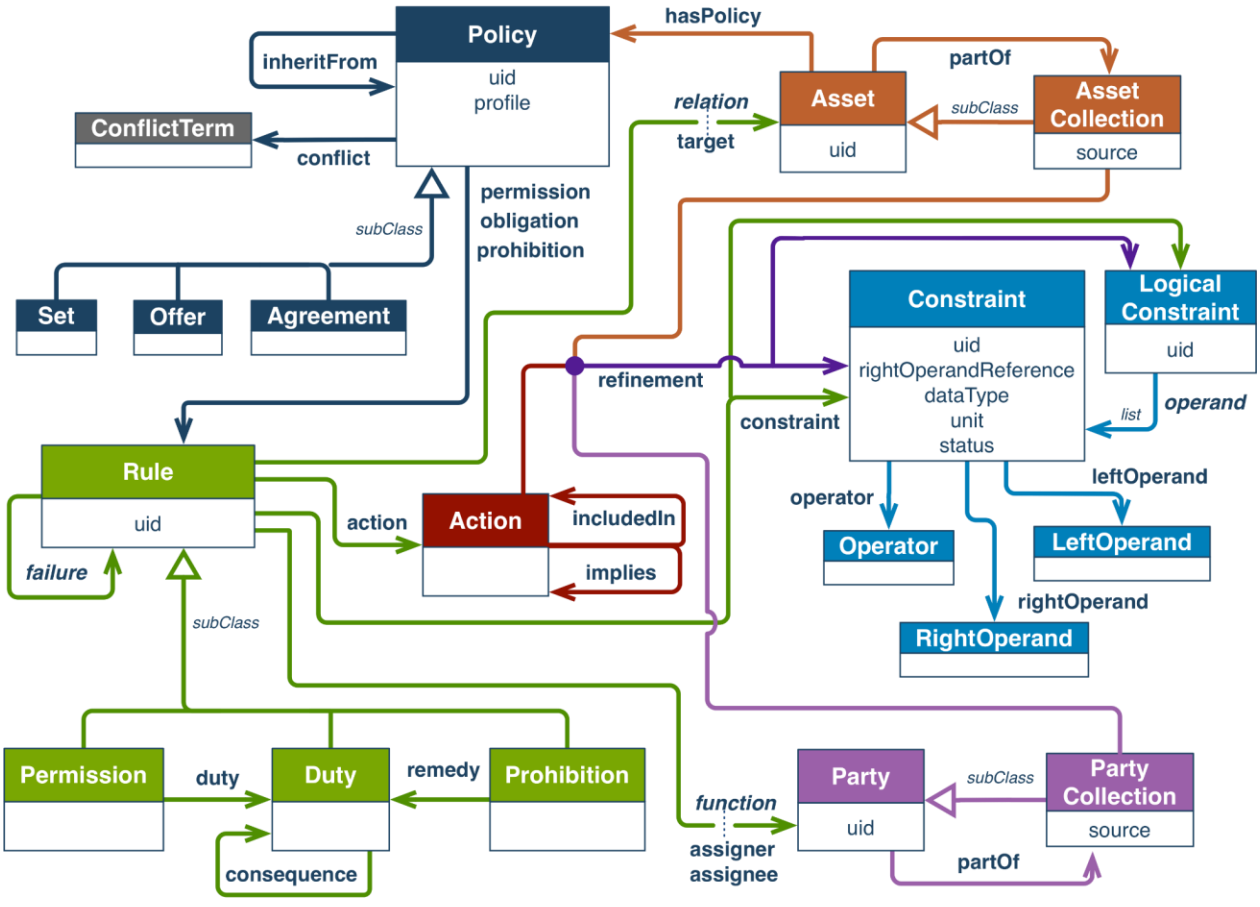

Figure 5: ODRL. The ODRL Information Model. Figure from https://www.w3.org/TR/odrl-model/ accessed on 2021-06-20

[32] https://www.w3.org/TR/odrl-model/, accessed June 2021

ODRL can plug into existing DRM architectures, or into open frameworks, for instance peer-to-peer (P2P) DRM services. It is considered as a way to express DRM policies striving to be compatible with other languages in the DRM community.

4.3 DCMI

DCMI (Dublin Core Metadata Initiative) or Dublin Core as it is commonly known is an initiative to create a digital library card catalog for the Web. Dublin Core is composed of 15 metadata elements that offer expanded cataloguing information and improved document indexing for search engines. Two forms of Dublin Core exist: Simple Dublin Core and Qualified Dublin Core. Simple Dublin Core expresses elements as attribute-value pairs using solely the 15 metadata elements from the Dublin Core Meta Element Set. Qualified Dublin Core increases the specificity of metadata by adding information about encoding schemes, enumerated lists of values, or other processing clues. Qualifiers are more complex and can pose challenges to interoperability. Dublin Core position is that resource discovery should be independent from the medium of the resource. While Dublin Core targets electronic resources, it aims to be flexible enough to help in searches for more traditional formats of data as well.

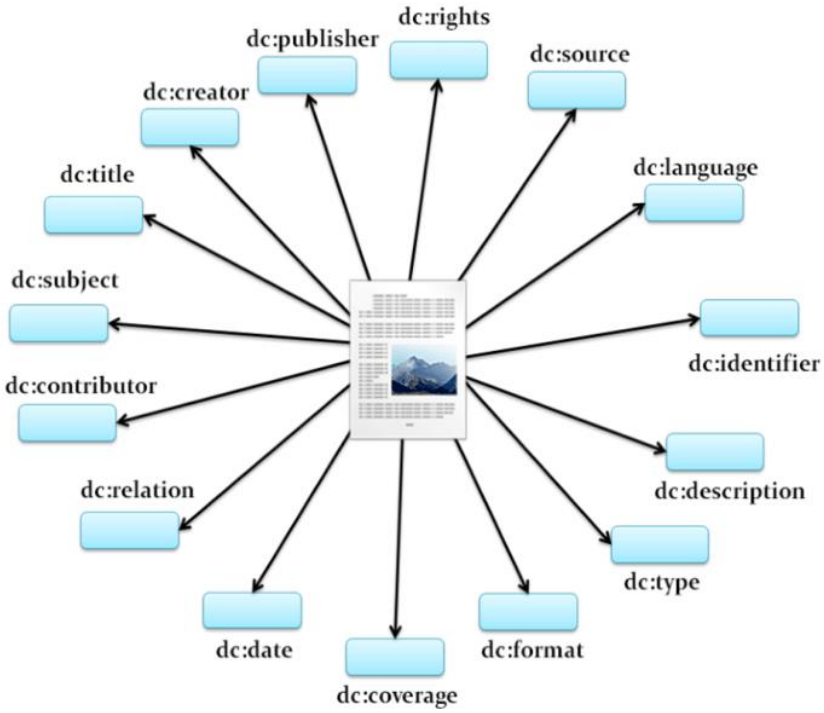

Figure 6: DCMI. The DCMI Element Set. Image from [9]

Each term is identified with a Uniform Resource Identifier (URI), a global identifier usable in Linked Data. Term URIs resolve to the “DCMI Metadata Terms” document when selected in a browser or, when referenced programmatically by RDF applications, to one of four RDF schemata. The scope of each RDF schema corresponds to a "DCMI namespace” or set of DCMI metadata terms that are identified using a common base URI, as enumerated in the DCMI Namespace Policy. In Linked Data, the URIs for DCMI

namespaces are often declared as prefixes in order to make data, queries, and schemata more concise and readable.

The four DCMI namespaces are:

- **http://purl.org/dc/elements/1.1/ The /elements/1.1/** namespace was created in 2000 for the RDF representation of the fifteen-element Dublin Core and has been widely used in data for more than twenty years. This namespace corresponds to the original scope of ISO 15836, which was published first in 2003 and last revised in 2017 as ISO 15836-1:2017 [ISO 15836-1:2017.
- **http://purl.org/dc/terms/** The /terms/ namespace was originally created in 2001 for identifying new terms coined outside of the original fifteen-element Dublin Core. In 2008, in the context of defining formal semantic constraints for DCMI metadata terms in support of RDF applications, the original fifteen elements themselves were mirrored in the /terms/ namespace. As a result, there exists both a dc:date (http://purl.org/dc/elements/1.1/date) with no formal range and a corresponding dcterms:date (http://purl.org/dc/terms/date) with a formal range of "literal". While these distinctions are significant for creators of RDF applications, most users can safely treat the fifteen parallel properties as equivalent. The most useful properties and classes of DCMI Metadata Terms have now been published as ISO 15836-2:2019 [ISO 15836-2:2019]. While the /elements/1.1/ namespace will be supported indefinitely, DCMI gently encourages use of the /terms/ namespace.
- **http://purl.org/dc/dcmitype/** The /dcmitype/ namespace was created in 2001 for the DCMI Type Vocabulary, which defines classes for basic types of entities that can be described using DCMI metadata terms.
- **http://purl.org/dc/dcam/** The /dcam/ namespace was created in 2008 for terms used in the description of DCMI metadata terms.

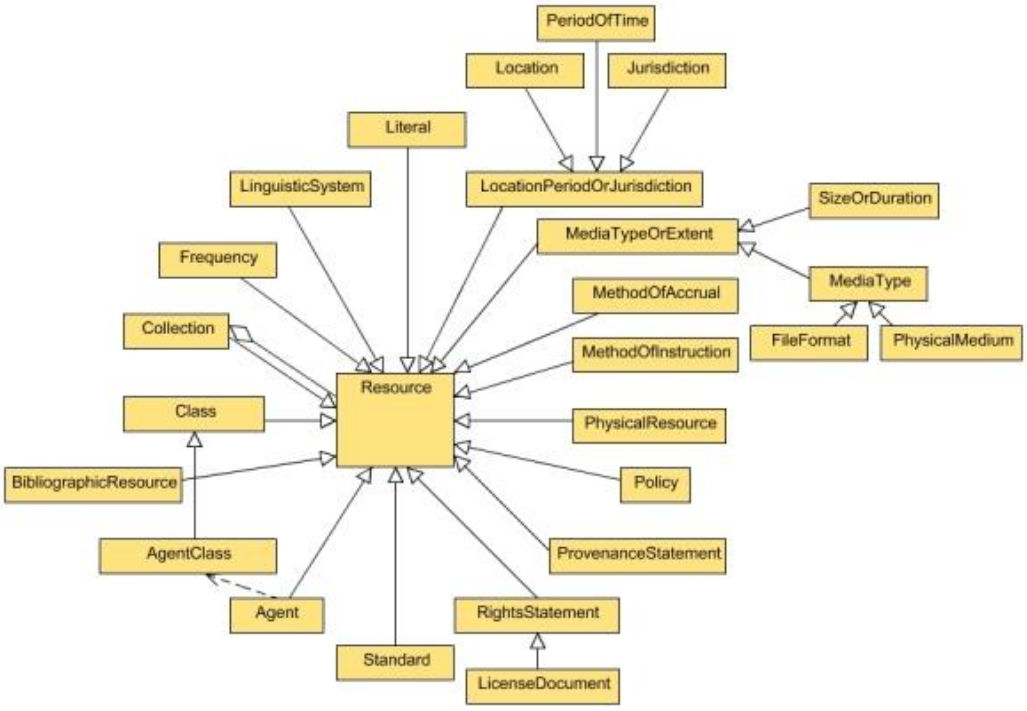

Figure 7: Dublin Core (DC). The relationships between the classes defined by Dublin Core. Image from https://www.dublincore.org/specifications/dublin-core/domain-range/2007-07-02/, accessed on 2021-06-29

5 The TRUSTS Information Model

The TRUSTS Information Model (TRUSTS-IM) specifies what classes of entities are described in metadata, what properties describe each of them, and what relations can exist between entities of different classes. It is, therefore, the schema of the TRUSTS KG. In the following, we describe the TRUSTS-IM in a textual and tabular manner, mostly aimed to be a reference for humans implementing components that deal with metadata. However, and following the different description levels of the IDS-IM, the authoritative version of the TRUSTS-IM will be described in RDF using OWL. The description provided here is provisional and will be revised as the next deployment stages of the TRUSTS platform progress.

For brevity of descriptions, we make use of the following namespaces in what follows:

Table 23: Use of namespaces

Prefix	Namespace
ids:	https://w3id.org/idsa/core/
trusts:	https://www.trusts-data.eu/ontologies/IM/

tdm:	https://www.trusts-data.eu/ontologies/TDM/
dcat:	http://www.w3.org/ns/dcat#
dct:	http://purl.org/dc/terms/
edmi:	None known to date

A high-level overview of the TRUSTS-IM can be seen in **figure Schema,** where the broad classes of entities that are present in the TRUSTS KG are depicted**.** Central to it is the notion of Asset with its different levels of concreteness, as defined in the IDS-IM. In brief, assets can be either served by a member organization or harvested from an external source (e.g., a datamarket or EOSC initiative). Accompanying is a set of configuration parameters which have effect on both the provider and consumer connectors, each of which in turn resides in a node. In turn, assets are subjects of contracts. All of these relations, and the corresponding metadata have effects on various components and processes in the TRUSTS platform, as depicted in the **figure Schema**.

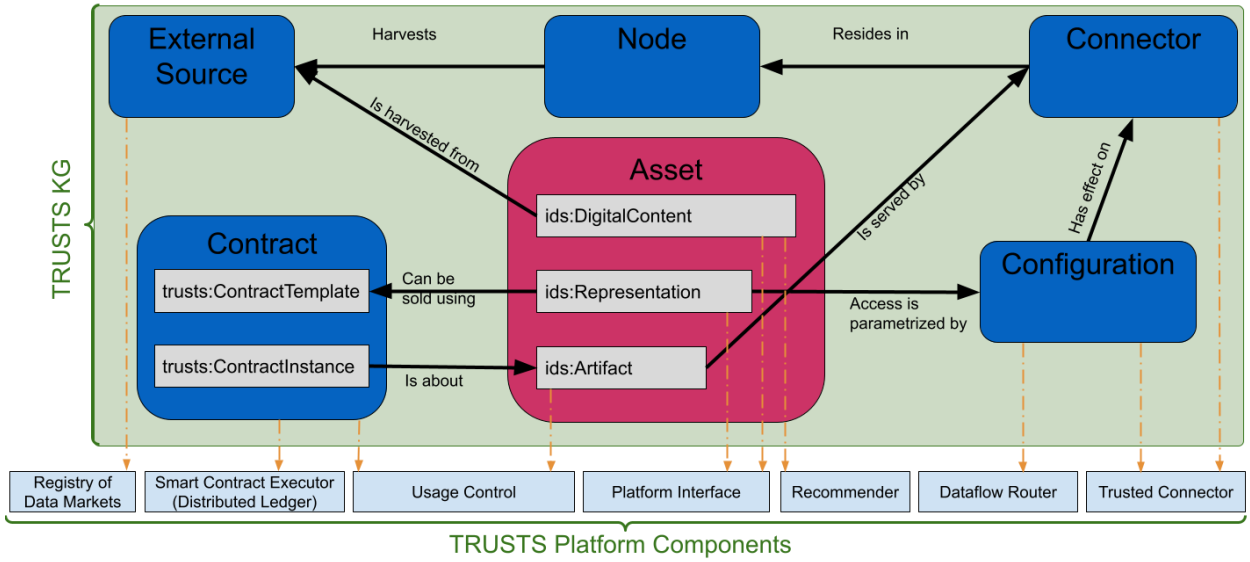

Figure 8: Schema: A high level overview of the TRUSTS-IM. Shown are the broad classes of entities that make up the TRUSTS-KG, and the components of the TRUSTS platform which they affect.

The detailed description of these broad classes and their relation to the IDS-IM is provided below. The ⊣ symbol denotes the subclass relation, and green colored elements are those which are not currently part of the IDS-IM

5.1 Asset

Classes of entities

Table 6: Asset, Classes of entities

Class Name	Description	URI
Catalog	A collection of resources.	ids:ResourceCatalog
Digital Content	An asset that will appear in the catalogue.	ids:DigitalContent
⊣Service	A software artifact which is executed on the provider's premises and the access to which can be traded in the TRUSTS platform.	dcat:DataService
⊣Application	A software artifact which is executed on the consumer's premises and whose executable image can be traded in the TRUSTS platform.	ids:AppResource
⊣Dataset	A file, or collection thereof, that can be traded in the TRUSTS platform, in which case a copy of which is transmitted from provider to consumer.	ids:DataResource
Representation	A specific representation of a Digital Content, with a particular access method and associated configuration.	ids:Representation
⊣Application Representation	Representation of an Application, to which configuration can be ascribed	ids:AppRepresentation
⊣Data Representation	Representation of a Dataset, for example a particular format or language in which the dataset can be distributed.	ids:DataRepresentation
⊣ Service Representation	Representation of a service, for example a deployment of a service with particular performance	trusts:ServiceRepresentation
Artifact	A particular instance of representation, which can be transferred to, or accessed by, the consumer. For example, an actual copy of a file, or a running server providing a service, or a container image for an app.	ids:Artifact
⊣Application Artifact	A specific instance of an application which is subject to contract and usage control	trusts:AppArtifact

⊣Service Artifact	A specific instance of a service	trusts:ServiceArtifact
⊣Dataset Artifact	A specific instance of a dataset, with a specific location, filesize, and file hash.	trusts:DatasetArtifact

Relationships between these entities

Table 7: Asset, Relationship between entities

Domain	Range	Relation	URI
ids:ResourceCatalog	ids:DigitalContent	offers resource	ids:offeredResource
ids:DigitalContent	ids:Representation	has representation	ids:representation
ids:DigitalContent	ids:Representation	has default representation	ids:defaultRepresentation
ids:Representation	ids:Artifact	has instance	ids:instance
dcat:DataService	ids:DataResource	serves dataset	dcat:servesDataset
ids:DigitalContent	ids:AppResource	requires application to be accessed	trusts:requiresApplication

Properties of entities

Since ids:Catalog is a subclass of dcat:Catalog, all the properties of the latter are also applicable to the former. Likewise, the properties of dcat:Distribution are applicable to entities whose class is either ids:Representation or any of its subclasses, and the properties of dcat:Dataset are applicable to entities whose class is either ids:DigitalContent or any of its subclasses. Finally, EDMI-specific metadata was added to datasets.

Table 8: Asset, Properties of entities

Classes	Property	Description in TRUSTS	URI
Catalog, Digital Content, Representation, inc. Dataset, Services & Applications	title	A human readable title	dct:title
	publisher	The organization making this asset available in TRUSTS	dct:publisher
	creator	The original producer of an asset	dct:creator

	contact point	Information on how to contact the publisher	dcat:contactPoint
	description	Human readable description of an asset.	dct:description
	keyword	Human readable, free text keywords for identifying an asset.	dcat:keyword
	theme	A main category of the resource, coming from a controlled vocabulary	dcat:theme
Catalog	homepage	The URL of the node publishing the catalog.	dcat:homepage
Digital Content	metric	Metric to provide some quantitative or qualitative information about the dataset	edmi:metric
	sample	A sample of the resource	idsa:sample
	citation	A citation or reference to another work that describes the dataset	edmi:citation
	referenceCitation	A citation or reference that describes the dataset	edmi:referencecitation
	authorisation	Type of authorisation required to access the dataset	edmi:authorisation
	Time frame	Time frame of dataset delivery (e.g. (Near) real time, Multiple)	rod:Timeframe
	Data access	Types of data access (e.g. API, Download)	rod:Dataaccess
Dataset	measurementTechnique	A technique or technology used in a dataset corresponding to the method used for measuring the corresponding variables	edmi:measurementtechnique

	variablesMeasured	The variables that are measured in the dataset	edmi:variablesmeasured
	scientificType	Scientific domain or type of the information provided in the dataset	edmi:Scientifictype
	creation date	Date of creation of the dataset	trusts:creationdate
	data collection type	Type of data collection (eg. survey, questionnaire, log)	trusts:collectiontype
	instrument	Instrument with which the data were collected	trusts:instrument
	publication	Type of publication	trusts:publication
Representation	licence	A URL of a human-redable copy of the contract template	dct:license
	contract template	One or more contract templates from where contract instances for the asset can be created	odrl:hasPolicy
	access URL	The URL that can be used to access the resource within the TRUSTS platform	dcat:accessURL
Data Representation	byte size	Number of bytes of a dataset	dcat:byteSize
	compression format	Compression format, if any	dcat:compressFormat
	format	Format in which the dataset is presented. From a controlled vocabulary.	dct:format
	identifier	The identifier property represents any kind of identifier for any kind of dataset (eg. ARK identifiers, ISBN)	edmi:identifier

	accessInterface	The type of interface to present the dataset	edmi:accessInterface
	structure	The description of the structure of the dataset (eg XML, database schema, codebook)	edmi:structure
	content Type	Type of content provided in the dataset based on its origin and type of processes (eg. raw, processed, summarised)	edmi:contentType
ids:Artifact	has contract	A contract that is attached to a specific instance of a resource	ids:contract

5.2 Configuration

Classes of entities

Table 9: Configuration: Classes of entities

Class Name	Description	URI
Resource Configuration	Configuration for apps and services	trusts:ResourceConfiguration
⊣App Configuration	App-specific configuration	ids:DataApp
⊣Service Configuration	Service-specific configuration	trusts:ServiceConfiguration
Endpoint	Service or Application Endpoint	ids:Endpoint
⊣App Endpoint	Application specific Endpoint	ids:DataAppEndpoint
⊣Service Endpoint	Service specific Endpoint	trusts:ServiceEndpoint
Parameter[33]	A possible entry in a key value structure transmitted as part of a request	trusts:Parameter

[33] This notion is mentioned in the IDS-IM extensively, but no class is provided for it in the Ontology

Relationships between these entities

Table 10: Configuration: Relationships between of entities

Domain	Range	Relation	URI
ids:Artifact	trusts:ResourceConfiguration	has configuration	trusts:hasConfiguration
trusts:ResourceConfiguration	ids:Endpoint	hasEndpoint	ids:appEndpoint
ids:Endpoint	trusts:Parameter	required headers	trusts:requiredHeaders
ids:Endpoint	trusts:Parameter	optional headers	trusts:optionalHeaders
ids:Endpoint	trusts:Parameter	query parameters	trusts:queryParameters

Properties of entities

Table 11: Configuration: Properties of entities

Classes	Property	Description	URI
Resource Configuration	protocol	TCP-based protocol used to communicate with the application. From a controlled vocabulary.	trusts:protocol
	OpenAPI 3.0 Specification	OpenAPI 3.0 Specification of the applications endpoints	ids:endpointDocumentation
	deployment name	Name, within the execution file, that the application's endpoints belong to	trusts:hasDeployName
	start of validity	starting time of configuration validity	trusts:configValidityStart
	end of validity	end time of configuration validity	trusts:configValidityEnd
Application Configuration	execution file	Content of the docker-compose file that determines how this application can be executed	trusts:executionFile
Service Configuration	authentication standard	The authentication standard that must be used to access this service. From a controlled vocabulary.	ids:authStandard
Endpoint	endpoint path	The HTTP route, relative to the resource's accessURL	ids:path

	endpoint description	Human readable information and description of the endpoint	ids:endpointInformation
	MIME type	the type of media returned by the endpoint. From a controlled vocabulary.	ids:mediaType

5.3 Connector

The notion of connector in TRUSTS is similar to the notion of connector in IDS. We refer the reader to the original IDS specification of connector[34] and its related classes and properties. We list here only those relevant to the discussions presented in this document

Classes of entities

Table 12: Connector: Classes of entities

Class Name	Description	URI
Connector	Represents the Trusted Connector in a TRUSTS node	ids:Connector
Connector Endpoint	One of several endpoints the endpoints to of a Trusted Connector	ids:ConnectorEndpoint

Relationships between these entities

Table 13: Connector: Relationship between entities

Domain	Range	Relation	URI
ids:Connector	ids:ConnectorEndpoint	The endpoint with which one Trusted Connector is available to other Trusted Connectors over IDSCPv2	ids:hasDefaultEndpoint
ids:Connector	ids:ConnectorEndpoint	The endpoint by which other components in the same TRUSTS node can reach this Trusted Connector	trusts:hasInternalEndpoint

[34] https://international-data-spaces-association.github.io/InformationModel/docs/index.html#Connector, accessed June 2021

Properties of entities

Table 14: Connector: Properties of entities

Classes	Property	Description	URI
Connector Endpoint	has port	the port that defines this connector endpoint	trusts:connectorPort
Connector Endpoint	has url	the URL where the connector can be reached	ids:accessURL

5.4 Node

In TRUSTS, a node is a computing infrastructure (e.g., virtual machine) that is operated by a given organization and that is only accessible within that organization's internal network and, through a Trusted Connector installed in it, from other nodes of the TRUSTS platform. A detailed description of this architecture, and the network setup that it implies, can be found in D2.6. The IDS-IM does not provide any functionality for the notion of node.

Classes of entities

Table 15: Node: Classes of entities

Class Name	Description	URI
Node	A computing infrastructure in which TRUSTS components are deployed.	trusts:Node
Organization	A person or organization who takes part in the TRUSTS platform.	ids:Participant
Component	A component of the TRUSTS platform	ids:InfrastructureComponent
⌙Connector	Represents the Trusted Connector in a TRUSTS node	ids:Connector

Relationships between these entities

Table 16: Node: Relationship between entities

Domain	Range	Relation	URI
trusts:Node	ids:Site	The node is deployed in this site	trusts:deploymentSite

trusts:Node	ids:Participant	The node is operated by this participant	trusts:operatedBy
ids:InfrastructureComponent	trusts:Node	The component is deployed in a node	trusts:deployedInNode

5.5 Contracts

Contracts are represented in TRUSTS as instances of the ODRL:Policy class. Both instance and template contracts are policies, the difference being that the template has a series of placeholders that are to be replaced by actual rdf resources or literals when the contract is instantiated. Thus, the act of instantiating a contract is, from the TRUSTS KG perspective, equivalent to making a copy of a Contract Template and replacing the placeholders with the correct values. The Contract Instance itself (its identifiers and the data/object properties mentioned below) are to be entered in the KG in order to enable usage control. However, the details of the policy may not necessarily be copied into the KG as nodes surrounding the Contract Instance. Rather, they can be written directly into the Smart Contract Executor, after a proper translation from ODRL into the smart contract scripting language. This translation is, as of writing, pending. An alternative would be to prepare in parallel a smart contract code and an ODRL description of a policy and link them together in a way that instantiating a contract amounts to simultaneously entering a new smart contract into the distributed ledger, and into the knowledge graph.

Here we include only the ODRL classes and properties which are relevant for operating components other than the smart contract executor, thus ignoring, for the moment, the intricacies of the policy language. The whole of ODRL, as described in the previous chapter, is however part of the TRUSTS-IM.

Classes of entities

Table 17: Contracts: Classes of entities

Class Name	Description	URI
Policy	A formalized description of a contract template or instance. From this, both a human readable text and a machine-actionable smart contract can be generated	odrl:Policy
⊣Contract Template	Blank invoice that is filled up for each purchase	trusts:contractTemplate
⊣Contracts instance	Has concrete date, buyer, money “invoice number”	trusts:contractInstance

Relationships between these entities

Table 18: Contracts: Relationships between entities

Domain	Range	Relation	URI
Organizations	Network Peers	A network peer belongs to an organization	trusts:blockchainPeer
trusts:contractInstance	trusts:contractTemplate	A contract instance is derived from a contract template	trusts:contractDerivedFrom
ids:representation	contract template	One or more contract templates from where contract instances for the asset can be created	odrl:hasPolicy
ids:artifact	trusts:contractInstance	A given artifact can have its usage regulated by a specific contract instance	ids:contract

Properties of entities

Table 19: Contracts: Properties of entities

Classes	Property	Description	URI
trusts:contractInstance	textual representation	A contract instance can be rendered into human-readable text	ids:contractDocument
trusts:contractInstance	programmatic representation	A contract instance can have a programmatic representation. This property points to its ID in, e.g., a Blockchain distributed ledger.	trusts:smartContractID
trusts:contractInstance	hash	A contract instance can have an associated hash value of its programmatic representation	trusts:smartContractHash

5.6 External Sources

In order to investigate the importance of individual classes and properties for interoperability, it was analysed which fields are already used by existing data markets. The following additional classes and properties were identified.

The interoperability solution envisaged in D3.4 leverages a taxonomy of business models for datamarkets [8]. Consequently, we indicate this aspect by using the abbreviation “tdm:...” (Taxonomy of Data Markets) in respective URIs.

Classes of entities

Table 20: External Sources: Classes of entities

Class Name	Description	URI
Dataset	A collection of data, published or curated by a single agent, and available for access or download in one or more representations.	dcat:Dataset
Participant	The member of the TRUSTS platform who is harvesting an external data source	ids:Participant
Provider	An external provider of resources	ids:Provider

Relationships between these entities

Table 21: External Sources: Relationships between entities

Domain	Range	Relation	URI
Participant	Participant	Smart contract	tdm:Smartcontract
Participant	Provider	Platform sponsor	tdm:Platformsponsor
Participant	Data service	Offers data service	tdm:offersdataservice
Participant	Provider	Contact affiliation	trusts:contactaffiliation

Properties of entitie

Table 22: External Sources: Properties of entities

Classes	Property	Description	URI
Provider	Country	Country of data market	tdm:Country
Provider	Data market description	Description of data market	tdm:Datamarketdescription
Provider	Payment currency	Payment currency	tdm:Paymentcurrency
Provider	Website	Website of data market	tdm:Website
Provider	Price discovery	Type of price discovery (e.g., negotiation, set by buyers)	tdm:Pricediscovery
Provider	Revenue model	Type of revenue model (e.g., asset sales, commissions)	tdm:Revenuemodel
Provider	Matching mechanism	Type of matching mechanism (e.g., many-to-many, many-to-one)	tdm:Matchingmechanism
Provider	Marketplace participants	Type of marketplace participants (e.g., Any, B2B, C2B)	tdm:Marketplaceparticipants
Provider	Platform architecture	Architecture of data market (centralized, decentralized)	tdm:Platformarchitecture
Provider	contact	Name of data market contact	trusts:contact

6 Conclusions and Next Actions

The TRUSTS KG will constitute the storage of metadata for driving the different functionalities of the TRUSTS platform. Any component which requires information about another component, asset, participant or contract can query the KG to acquire this information and act upon it.

The TRUST-IM which was presented in this document constitutes the schema of the TRUSTS KG. By being provided with formally defined schema, with explicit semantics, the statements contained in the KG can be unambiguously interpreted by the different components. Additionally, any external systems interoperating with the TRUSTS platform will benefit from this formalized description in order to react adequately to metadata exchanged with TRUSTS.

This document presents only the schema, that is, the shape of the TRUSTS KG. As this KG becomes populated with metadata, and as this metadata is acted upon, this schema will most likely be refined. Further refinement and adjustment will occur as a result of the parallel evolution of the other schemata in which the TRUSTS-IM is based on, especially the IDS-IM. Continuation of the exchange with the IDS-IM working group will be necessary to guarantee smooth integration between the two initiatives.

In the upcoming months, the different pipelines for the population and use of metadata will be outlined and deployed. For this, a series of technological developments will also be undertaken, based on existing infrastructure from IDS and DMA projects. The next report on the TRUSTS KG will include the above-mentioned refinements to the information model, as well as reports on the operation of metadata ingestion and consumption. Finally, the tests of interoperability with EOSC and GAIA-X will also bring new insights, and the results of this information exchanges will also be reported upon.

Another important aspect of metadata management that will be addressed during the remainder of the project is the scalability of the Knowledge Graph as a centralized metadata storage. In particular, when millions of assets are present in the catalogue, and every transaction triggers the registration of a new contract instance, it can be expected that the limits of triplestores and graph databases will be reached. As already deployed in the current status of the platform, a layer of caching based on a document indexed (Apache Solr) can be put on top of the KG for cataloguing purposes, that is, to power the search mechanisms. Further caching mechanisms, coupled with replication of relevant segments of the TRUSTS-KG in the individual platform nodes, and processing using big data tools, will also be investigated and reported upon in the final deliverable.

The flexibility that comes with a knowledge graph, its fluidly evolving schema and the way it can be linked to external data sources allows for further extensions by use of additional services, without affecting the metadata ingestion mechanism already in place. For example, machine translation of literals in the graph (e.g., names, descriptions, etc.), coupled with multilingual vocabularies, can be used to turn the catalogue of the TRUSTS platform into a truly multilingual one. Furthermore, smart information extraction mechanisms that couple NLP with the information contained in the graph can aid in discovery and recommendation by linking resources in TRUSTS with those in external knowledge graphs.

7 References

[1] Hogan, A., et al. (2020). Knowledge graphs. *arXiv preprint arXiv:2003.02320*.

[2] Bader, S., et. al (2020, November). The International Data Spaces Information Model–An Ontology for Sovereign Exchange of Digital Content. In *International Semantic Web Conference* (pp. 176-192). Springer, Cham.

[3] Pellegrini, T., Mireles, V., Steyskal, S., Panasiuk, O., Fensel, A., & Kirrane, S. (2018). Automated rights clearance using semantic web technologies: The DALICC framework. In *Semantic Applications* (pp. 203-218). Springer Vieweg, Berlin, Heidelberg.

[4] Ben Ellefi, M., Bellahsene, Z., Breslin, J. G., Demidova, E., Dietze, S., Szymański, J., & Todorov, K. (2018). RDF dataset profiling–a survey of features, methods, vocabularies and applications. Semantic Web, 9(5), 677-705.

[5] Ivanschitz, B. P., Lampoltshammer, T. J., Mireles, V., Revenko, A., Schlarb, S., & Thurnay, L. (2018). A data market with decentralized repositories.

[6] Lagoze, C., & Van de Sompel, H. (2001, January). The Open Archives Initiative: Building a low-barrier interoperability framework. In *Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries* (pp. 54-62).

[7] Schütte, J., Brost, G. & Wessel, S. (2018). Datensouveränität im Internet der Dinge – Der Trusted Connector im Industrial Data Space. Whitepaper, Fraunhofer AISEC. (https://arxiv.org/pdf/1804.09442.pdf)

[8] van de Ven, M.R. (2020). Creating a Taxonomy of Business Models for Data Marketplaces. Master Thesis, TU Delft Technology, Policy and Management.

[9] Chekry, A., ORICHE, A., KHALDI, M., & ELKADIRI, K. (2011). Semantic web technologies for the reuse and adaptation of educational documents in e-learning. In *The 2nd International Conference on Multimedia Computing and Systems*.

[10] Hasnain, A., & Rebholz-Schuhmann, D. (2018, June). Assessing FAIR data principles against the 5-star open data principles. In European Semantic Web Conference (pp. 469-477). Springer, Cham.