INTERNATIONAL DATA SPACES ASSOCIATION

Position Paper | Version 2.0 | November 2019

Usage Control in the International Data Space

● Position Paper of members of the IDS Association

○ Position Paper of bodies of the IDS Association

○ Position Paper of the IDS Association

○ White Paper of the IDS Association

Fraunhofer

Publisher

International Data Spaces Association
Anna-Louisa-Karsch-Str. 2
10178 Berlin
Germany

Editor

Sebastian Steinbuss
International Data Spaces Association

Industrial Data Space

www.fraunhofer.de/en/research/lighthouse-projects-fraunhofer-initiatives/industrial-data-space.html

Industrial Data Space+

www.fraunhofer.de/en/research/lighthouse-projects-fraunhofer-initiatives/industrial-data-space.html

Research Center Data Spaces

www.cit.fraunhofer.de

Authors & Contributors

Andreas Eitel, Fraunhofer IESE

Christian Jung, Fraunhofer IESE

Christian Kühnle, Fraunhofer IOSB

Fabian Bruckner, Fraunhofer ISST

Gerd Brost, Fraunhofer AISEC

Pascal Birnstill, Fraunhofer IOSB

Ralf Nagel, Fraunhofer ISST

Sebastian Bader, Fraunhofer IAIS

Abstract

In the age of Industry 4.0, data exchange between different organizations is an essential prerequisite to add more value to data and to develop modern business models. However, we have to solve several challenges to facilitate a secure and trustworthy data exchange between different organizations. Data sovereignty is a key success factor for data-driven business models. In the Industrial Data Space, we provide solutions to realize a secure and trustworthy data exchange as well as data sovereignty.

In this report, we focus on data usage control and data provenance that are conceptual and technological solutions to cope with data sovereignty challenges. We introduce a common scenario for the Industry 4.0 age, in which a supplier and an original equipment manufacturer (OEM) are exchanging data to mitigate risks in the supply chain management. We describe the difference between access control and usage control, the usage control concepts and related concepts such as digital rights management or user managed access.

We present the implementation of data usage control in the IDS. In doing so, we present the policy language, its integration to the IDS information model and introduce commonly used policies. Thus, we present a policy editor for expressing usage restrictions in the open digital rights language and their transformation to machine-readable policies. Our work includes a discussion about the different expansion stages for implementing usage control named the Usage Control Onion.

As there are different ways to implement data usage control, we present three approaches researched and developed within Fraunhofer: The MYDATA Control Technologies, the Logic-based Usage Control and Degree. Every technology is presented in detail including its integration concepts. Finally, we compare these technologies and discuss them. We address data provenance as additional concept to data usage control to cope with transparency and accountability.

Before we elaborate on the current state and future work of data usage control and provenance tracking within the IDS, we discuss the relation to other core components in the IDS Reference Architecture Model.

Keywords: MYDATA, IND^{2}UCE, LUCON, Degree, Data Usage Control, Data Provenance, Information Model, Industrial Data Space, IDS, FDS, CCIT

Table of Contents

1. Introduction 5
1.1 Motivation and Problem 5
1.2 Accompanying Scenario: Supply Chain Risk Management 6
1.3 Document Structure 7

2. Usage Control Concepts 8
2.1 Access Control 8
2.2 Usage Control 10
2.3 Enforcement 11
2.4 Decision & Information 11
2.5 Specification, Management and Negotiation 13
2.6 Related Concepts 14

3. Implementation of Usage Control in the IDS 15
3.1 Overview 15
3.2 Information Model and Policy Language 15
3.3 Policy Specification 19
3.4 The Usage Control Onion 26
3.5 Integration with the Reference Architecture Model 29

4. Usage Control Technologies 31
4.1 MYDATA Control Technologies 31
4.2 Logic-based Usage Control (LUCON) 38
4.3 Degree (D°) 41
4.4 Usage Control Technologies in Comparison 47
4.5 Discussion 49

5. Data Provenance, Transparency and Accountability 50
5.1 Relationship between Data Provenance and Usage Control 50
5.2 Operating Principle and Architecture 50
5.3 Transformation of ODRL policies 51

6. Discussion and Future Work 51
6.1 Discussion 51
6.2 Future Work 53

A. Glossary 55

B. Appendix: Further Information and Contact Persons 56
B.1 MYDATA Control Technologies 56
B.2 Logic-based Usage Control 56
B.3 Degree (D°) 56
B.4 Data Provenance 56

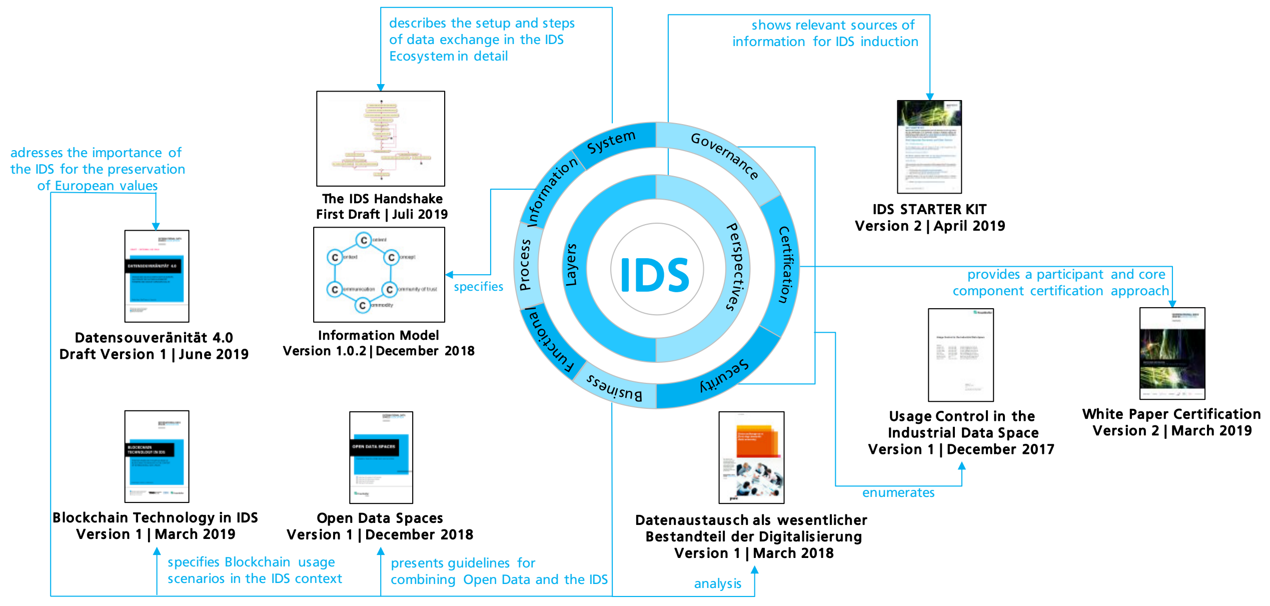
REFERENCES 57

1. Introduction

The Industrial Data Space is about creating a data space where businesses can exchange and exploit data in a secure manner. For the Industrial Data Space as well as other data-driven businesses, data sovereignty is a key success factor. Data sovereignty has the goal to provide a Data Owner [1] with full control over her data. This includes being able to control the usage of her data by the Data Consumer.

In this document, we present data usage control and data provenance as conceptual and technical solution to cope with data sovereignty.

The following illustration provides an overview of IDS documents and their relation. It presents all relevant documents and their relation to each other. The document map (see Figure 1) will be updated based on new produced or adapted artifacts.



describes the setup and steps of data exchange in the IDS Ecosystem in detail
shows relevant sources of information for IDS induction
adresses the importance of the IDS for the preservation of European values
The IDS Handshake First Draft | Juli 2019
IDS STARTER KIT Version 2 | April 2019
Datensouveränität 4.0 Draft Version 1 | June 2019
Information Model Version 1.0.2 | December 2018
specifies
IDS
System
Governance
Information
Process
Layers
Perspectives
Certification
Functional
Business
Security
provides a participant and core component certification approach
Usage Control in the Industrial Data Space Version 1 | December 2017
White Paper Certification Version 2 | March 2019
Blockchain Technology in IDS Version 1 | March 2019
Open Data Spaces Version 1 | December 2018
Datenaustausch als wesentlicher Bestandteil der Digitalisierung Version 1 | March 2018
enumerates
specifies Blockchain usage scenarios in the IDS context
presents guidelines for combining Open Data and the IDS
analysis

Figure 1: International Data Spaces "Document Map"

1.1 Motivation and Problem

Nowadays, business is spurred by continuously exchanging information between business partners. However, data is typically secured by access control mechanisms only. After access to data has been granted by these mechanisms, data can be arbitrarily altered, copied and disseminated by the recipient. Data usage control offers possibilities to control future data usages beyond the initial access (also known as obligations).

In the age of Industry 4.0, there is more critical and sensitive data exchanged between business partners (see Figure 2). In general, companies have intrinsic and extrinsic motivations to apply usage control: On the one hand, companies may use usage control to prevent misuse of their own data, to protect their intellectual property, and to preserve the data value (intrinsic motivation). On the other hand, companies have to comply with legal obligations such as the European Union General Data Protection Regulation EU-GDPR (extrinsic motivation). Hence, companies have to prevent misuse of other persons or companies data.

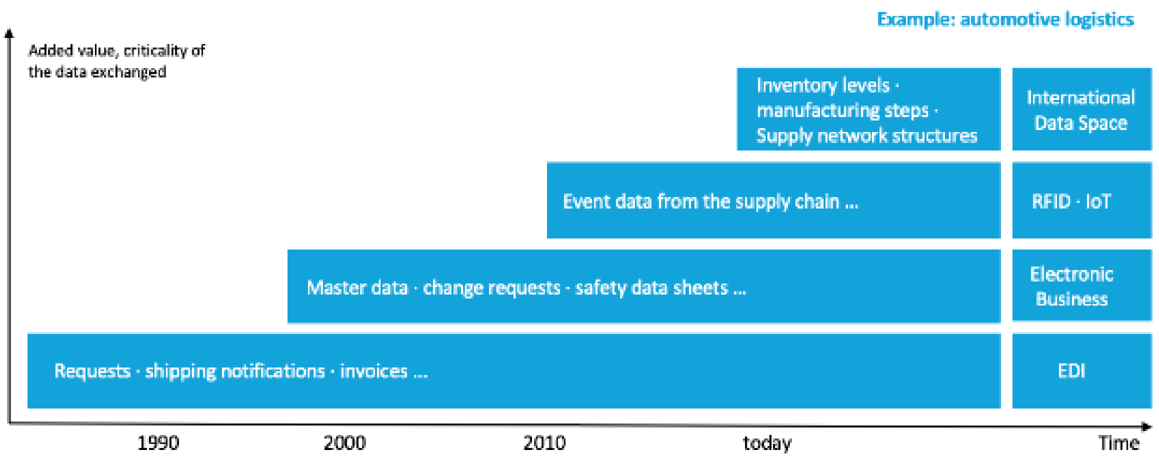


Figure 2: Data exchange in the age of industry 4.0

1.2 Accompanying Scenario: Supply Chain Risk Management

The following subsection presents our application scenario for data usage control. In the age of globalization and high cost pressure, supply networks of automotive original equipment manufacturers (OEM) are complex and interference prone for risks (e.g., earthquake, fire, war). For that reason, supply chain risk management (SCRM) becomes more and more important for a high supply reliability.

Figure 3 illustrates the data exchange between a supplier and the OEM in a collaborative SCRM scenario. On the one hand, there is data flowing from the suppliers to the OEMs such as affected parts and sub-supplier information, which the OEMs use in their supplier management system. On the other hand, the OEMs send data such as part demand or inventory range to the suppliers, which the suppliers process in their risk management system.

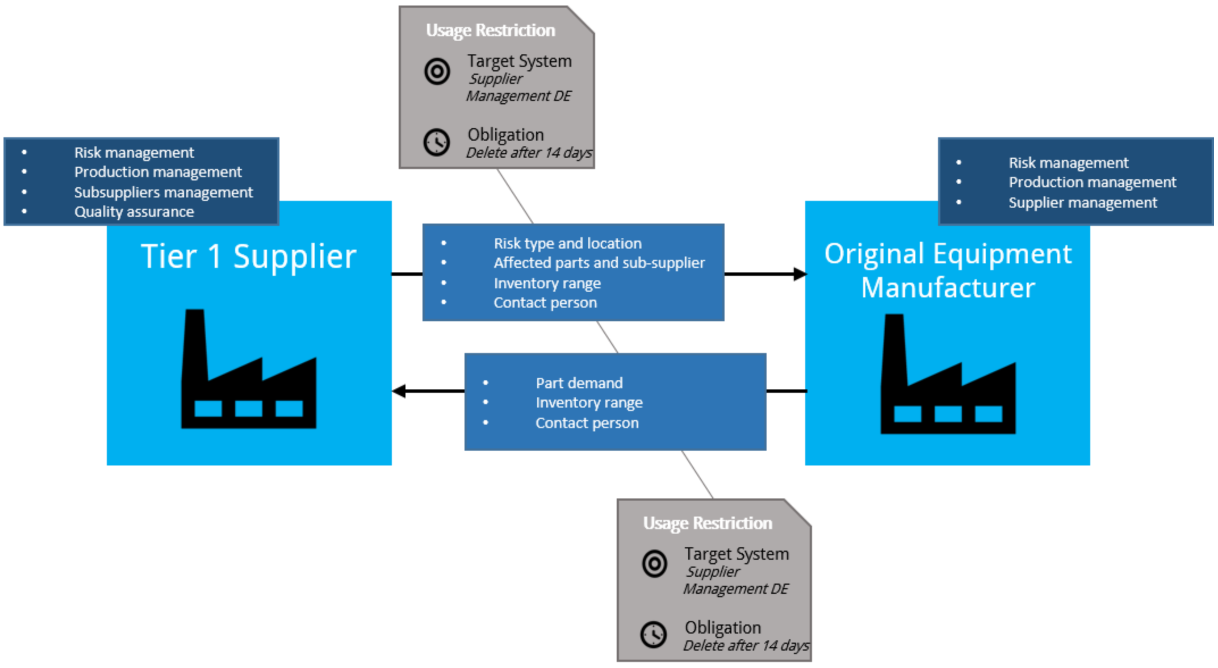


Figure 3: Supply Chain Risk Management illustration

Nowadays in the SCRM processes, most of the communication between the OEMs and the suppliers is done via phone, email, or web conferences. Table 1 shows the attributes of the data exchange from the supplier and OEM perspective (supplier as data provider and OEM as data provider).

From/To	Supplier	OEM
Supplier		• Risk type and location • Affected parts and sub-supplier •Inventory range • Contact person
OEM	• Part demand •Inventory range • Contact person	

Table 1: Attributes of the data exchange from the supplier and OEM perspective

In the process, there is sensitive and valuable data provided by the supplier as well as by the OEM: For example, data about the sub-supplier is very sensitive for the supplier. With such data, the OEM could skip the supplier and purchase directly form the sub-supplier. The part demand and inventory range are sensitive data for the OEM, because they make the production volume and warehouse transparent.

An automation of the data exchange in the SCRM process would lead to time and money savings for suppliers and OEM. In this case, the systems must ensure that the exchanged data is compliant with the company policies. This is where usage control can be used as technical extension to a contract to technically enforce the policies of the respective data provider. In fact, usage control improves security by controlling the data usage on the target system. Examples for appropriate policies in natural language are:

- The OEM can only use supplier data for risk or bottleneck management, but not for purchasing or sales purposes.
- The OEM has to delete all exchanged data in the SCRM process after 14 days.
- The supplier has to delete all exchanged data in the SCRM process after three days.
- The supplier can only import data from the OEM into the system “risk management”.

1.3 Document Structure

We structure the remainder of the document as follows:

Chapter 2 addresses the difference between access control and Usage Control as well as related concepts such as Digital Rights Management and Windows Information Protection. In addition, we introduce the basic concepts of usage control comprising the technical enforcement, decision making and information retrieval and policy specification, management and negotiation.

In Chapter 3, we address the general implementation of Usage Control in the IDS. In doing so, we introduce the information model and the relation to the Usage Control policy language. Hence, we present the policy specification and the derived policy classes within the IDS as well as the policy transformation and how to negotiate them. We discuss the different stages of Usage Control in the Usage Control Onion. Finally, we describe the integration with the Reference Architecture Model.

Chapter 4 is dedicated to the different Usage Control technologies that offer technical solutions to enforce our usage restrictions. We present the MYDATA Control Technologies

(MYDATA), the Logic-based Usage Control (LUCON) and Degree (D°). In more detail, we present communication flow, the integration concept and the transformation of ODRL policies to the technology-dependent policies. Finally, we compare these technologies and conclude with a discussion.

We present Data Provenance in Chapter 5. Data Provenance is a complementary concept to the enforcement technologies to cope with transparency and accountability of data usages. Hence, we present the relation between Data Provenance and Usage Control, the Data Provenance principle, its architecture and how data provenance is reflected in the ODRL policies.

In Chapter 6, we discuss our work by presenting capabilities, current limitations and implications of Usage Control. We end with future work such as the parameters of the policy negotiation process, next activities for Data Provenance and our implementation.

2. Usage Control Concepts

Usage control is an extension to traditional access control (see Figure 4). It is about the specification and enforcement of restrictions regulating what must (not) happen to data. Thus, usage control is concerned with requirements that pertain to data processing (obligations), rather than data access (provisions). Usage control is relevant in the context of intellectual property protection, compliance with regulations, and digital rights management.

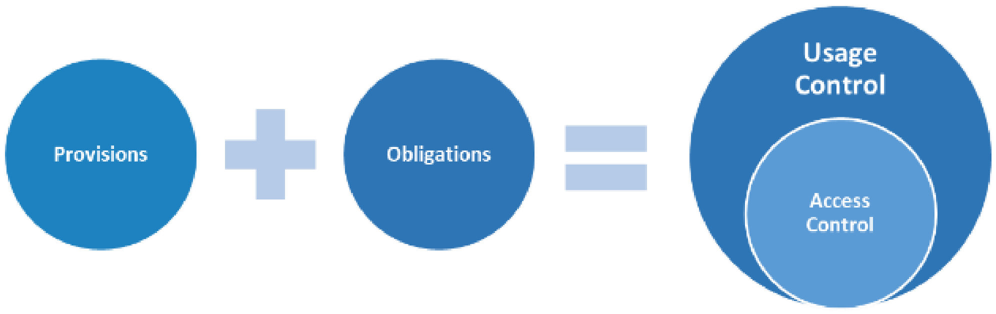


Figure 4: Usage Control consists of provisions and obligations

2.1 Access Control

In information security, access control restricts access to resources. The term authorization is the process of granting permission to resources. Several access control models exist, such as Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-based Access Control (RBAC), Attribute-based Access Control (ABAC), etc. Although such a plethora of access control models exists, RBAC and ABAC are most commonly used.

We will use the XACML (eXtensible Access Control Markup Language) Standard [2] to introduce commonly used terms in the field of access control. XACML is a policy language to express ABAC rules. The main building blocks of the language are subject, action, resource and environment. The subject describes who is accessing a data asset (e.g., a user). The action describes what the subject wants to perform on the data asset (e.g., read, write). The resource describes the data asset. Finally, the environment specifies the context (e.g., time, location).

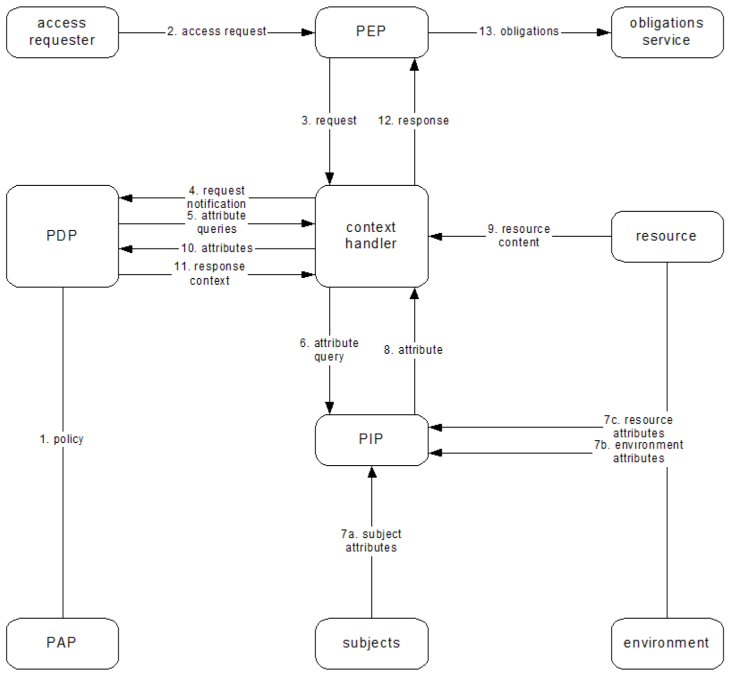


Figure 5: XACML data flow illustration

Figure 5 illustrates the dataflow model of XACML and the main actors or components to implement it: Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Information Point (PIP), and Policy Administration Point (PAP).

In general, attributes can describe anything and anyone, but tend to split into four categories**:**

- **Subject attributes**
  Attributes that describe the user by e.g. age, role or clearance.
- **Action attributes**
  Attributes that describe the action attempted e.g. read, delete or view.
- **Resource (or object) attributes**
  Attributes that describe the resource itself e.g. object type, location or classification.
- **Contextual (environment) attributes**
  Attributes that address time, location or other dynamic aspects.

Access control in the IDS is a resource-centric regulation of access requests from subjects (i.e., IDS participants) to resources (i.e., data services). Resource owners define attribute-based access control policies for their endpoints and define the attribute values a subject must attest in order to grant access to the resource. These attributes may include:

- Specific identity of connector(s) (only access requests from a specific connector / specific connectors will be granted)
- Connector attributes (only access requests from a connector that possesses specific attributes will be granted)
- Security profile requirements (only access requests from a connector that fulfills specific security feature requirements will be granted, e.g., having a TPM >= 1.2 and doing application isolation)

The actual access control decision has to be taken within the connector and can be realized using technologies such as XACML or JAAS, depending on the implementation of the connector. The IDS security architecture does not dictate a specific access control enforcement language or implementation. A discussion paper about access control for industry 4.0 can be found here [3].

2.2 Usage Control

In contrast to access control, where access to specific resources (e.g., a service or a file) is restricted, the IDS architecture additionally supports data-centric usage control. In general, the overall goal is to enforce usage restrictions for data after access has been granted. Therefore, the purpose of usage control is to bind policies to data being exchanged and to continuously control the way how messages may be processed, aggregated, or forwarded to other endpoints. This data-centric perspective allows the user to continuously control data flows, rather than accesses to services. At configuration time, these policies support developers and administrators in setting up correct data flows.

At runtime, the usage control enforcement prevents IDS connectors from treating data in an undesired way, for example by forwarding personal data to public endpoints. Thus, usage control is both a tool for system integrators to ensure they are not building an architecture that violates security requirements, and an audit mechanism, which creates evidence of a compliant data usage.

The following examples illustrate security requirements that cannot be achieved using traditional access control, but rather require data-centric usage control:

- **Secrecy**
  Classified data must not be forwarded to nodes which do not have the respective clearance.
- **Integrity**
  Critical data must not be modified by untrusted nodes as otherwise their integrity cannot be guaranteed anymore.
- **Time to live**
  A prerequisite for persisting data is that it must be deleted from storage after a given period of time.
- **Anonymization by aggregation**
  Personal data must only be used as aggregates by untrusted parties. A sufficient number of distinct records must be aggregated in order to prevent deanonymization of individual records.
- **Anonymization by replacement**
  Data which allows a personal identification (e.g., faces in camera images) must be replaced by an adequate substitute (e.g., blurred) in order to guarantee that individuals cannot be deanonymized from the data.
- **Separation of duty**
  Two data sets from competitive entities (e.g., two automotive OEMs) must never be aggregated or processed by the same service.
- **Usage scope**
  Data may only serve as input for data pipes within the connector, but must never leave the connector to an external endpoint.

It is important to note that the purpose of usage control is to allow the specification of such constraints and enforcing them in the running system. It is a prerequisite to usage control that the enforcement mechanism itself is trusted, i.e. usage control itself does not establish trust in an endpoint. It rather builds upon an existing trust relationship and facilitates the enforcement of legal or technical requirements such as service level agreements (SLA) or data privacy regulations. Thus, users must be aware that usage control will only provide certain enforcement guarantees if applied on highly trusted platforms, such as Trusted Connectors in the International Data Spaces (see [4]).

2.3 Enforcement

For enforcing usage restrictions, data flows need to be monitored and potentially intercepted by control points (i.e., PEPs). These intercepted data flows are given to the decision engine (i.e., the PDP) for requesting permission or denial of the data flow. In addition to just allowing or denying the data flow, the decision can also require a modification of data. A PEP component encapsulates the enforcement.

Regarding our accompanying scenario, OEM and supplier demand the deletion of data after a certain time or that only a limited audience can access the sensitive data. Hence, we have to intercept the data flow and check which audience (i.e., processing system) is using the data. For example, the supplier demands the OEM that only the supplier management system can use the data.

2.4 Decision & Information

The enforcement relies on a decision. A Policy Decision Point (PDP) takes the responsibility to answer incoming requests (i.e., data flows) from a PEP with a decision (see Figure 6). The decision-making based on usage restrictions is also called (policy) evaluation. There are several evaluation possibilities such as event- (see Section 4.1), or flow-based approaches (see Section 4.1.5 and 4.3).

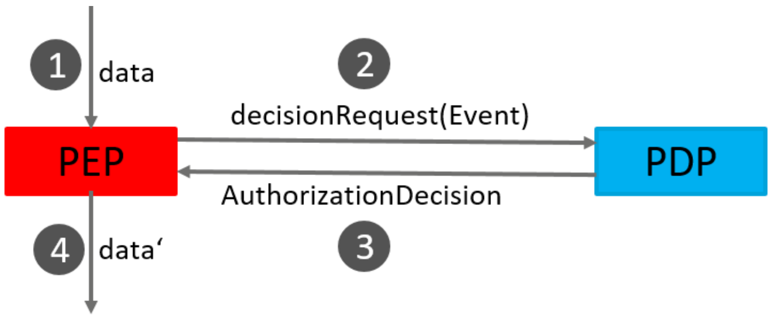


Figure 6: Illustration of a PEP intercepting data with decision making (PDP)

For event-based systems, data usage occurrences are represented as events including attributes to characterize the data usage. The event processing can be differentiated in simple processing (e.g., event-condition-action paradigm) and stream processing (e.g., sliding window) of events. The terms “event stream processing” and “complex event processing” are often used interchangeably.

In our accompanying scenario, we can model the transition of data as event with attributes about the data itself and the recipient. The attributes contain metadata and the target system (e.g., supplier management system). Taking our example from the previous section, the decision engine would draw a deny decision if the target system does not correspond to the expected supplier management system.

The policy decision may also depend on additional information that is not present in the intercepted data flow itself. This includes information about contextual in-formation such as previous data usages or the geographical location of an entity. There is also the possibility for pre- or post-conditions that have to hold before (e.g., integrity check of the environment) and after (e.g., data item is deleted after usage) the decision-making. In addition, there is the possibility to define on-conditions that have to hold during usage (e.g., only during business hours). These conditions usually specify constraints and permissions that have to be fulfilled before, during, and after using the data (see Figure 7).

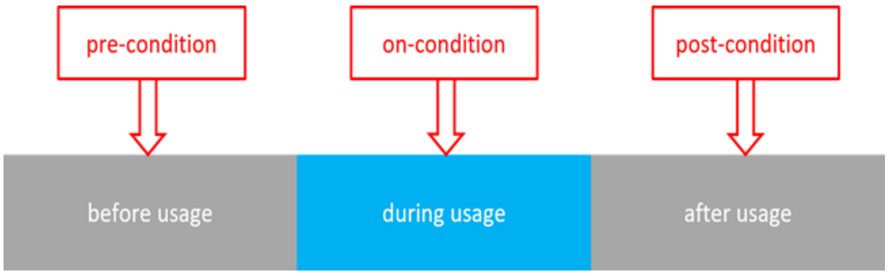


Figure 7: Types of conditions and when they are enforced

A Policy Information Point (PIP) provides missing information for the decision-making. In addition, we can use such a component to get contextual information for or about the intercepted system action (e.g., data flow information, geolocation of the requesting device).

Regarding our accompanying scenario, we may transform the D-U-N-S number [5] of a supplier to a concrete supplier name and address information. For example, if we want to limit the use of data depending on the geolocation of the supplier, a PIP can resolve the D-U-N-S number to a postal address and finally the postal address to GPS coordinates. Supplier and OEM are usually using different part numbers. Therefore, another example for a PIP is the translation of supplier part number to OEM part number and vice versa.

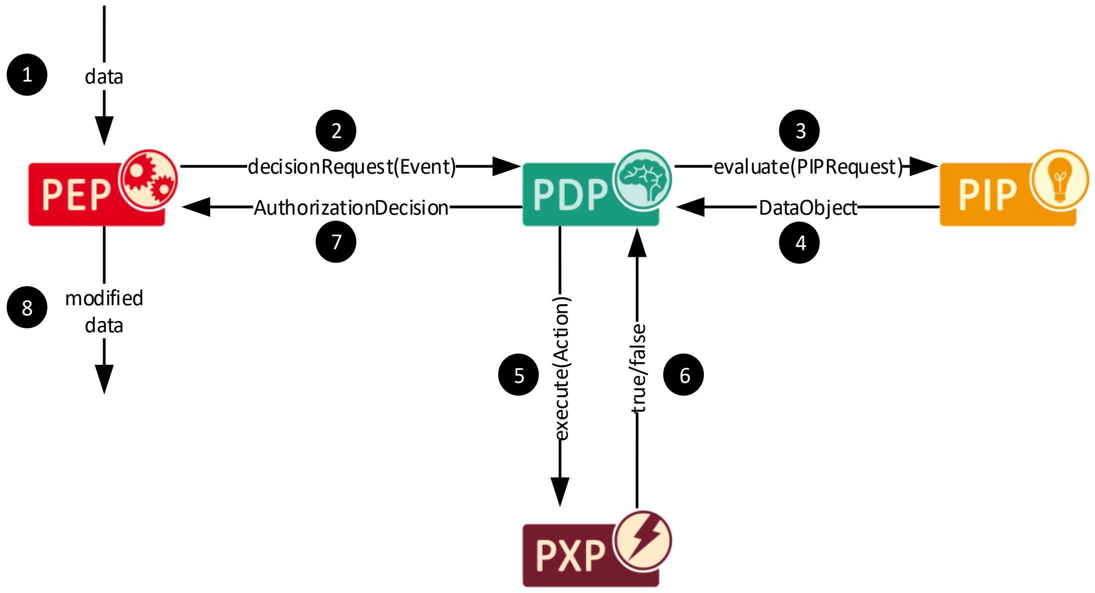


Figure 8: Full illustration of a usage controlled data flow

Finally, there is the concept of a Policy Execution Point (PXP). A PXP is used to per-form additional actions based on the policy rules, such as sending an email when data is used or writing to a specific log system. Figure 8Figure 8 illustrates an exemplary sequence of all processing steps to enforce usage control restrictions on a data flow:

1. PEP intercepts the data flow
2. PEP transforms the data flow to a decision request and sends that decision re-quest to PDP
3. PDP starts the policy evaluation and invokes a PIP to retrieve additional information
4. PIP responds with the requested data to the PDP

5. PDP triggers an additional action at a PXP
6. PXP confirms that the action succeeded to the PDP
7. PDP sends authorization decision to the PEP
8. PEP enforces the decision on the intercepted data flow

2.5 Specification, Management and Negotiation

Another important aspect of usage control is the specification and management of usage restrictions. Data providers have to express their restrictions on their data in a more or less formal way. For a technical enforcement, the specification must produce a machine-readable output. The Policy Administration Point (PAP) is the entry point for specification of usage policies, often via a user friendly graphical interface.

In our accompanying scenario, the Collaborative Supply Chain Risk Management (CSCRM) App takes the role of the PAP. There is a version for the supplier and a version for the OEM to specify their data usage restrictions.

A Policy Management Point (PMP) administers the usage restrictions. Hence, the component is concerned with the policy life cycle. This includes the instantiation, negotiation, deployment, and revocation of usage restrictions, as well as conflict detection and resolution.

There are two ways where usage restrictions are placed. First, usage restrictions can be adhered to the data, which is also called sticky policy [6]. Sticky policies are one way to cope with the distribution of the usage restrictions. In this approach, machine-readable usage restrictions (policies) stick to data when it is exchanged. There exist different realization possibilities. Usually, data is encrypted and can only be decrypted when the adherence to the usage restrictions are guaranteed. Second, policies can be stored independently from the data, for instance, in a central component (i.e., a PMP/PRP). In this case, the management component has to take responsibility to exchange the usage restrictions between different systems.

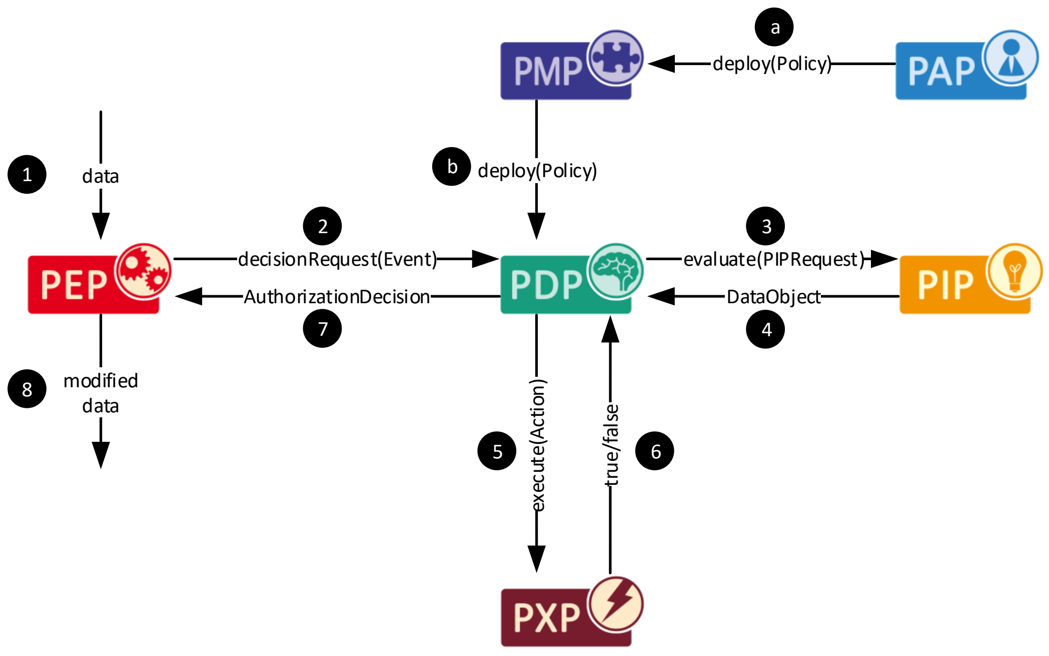


Figure 9: Usage Control illustration with PMP and PAP

In Figure 9, the PAP and PMP interactions are represented in the steps a) and b) to illustrate the deployment as policies as an example sequence.

The management of usage policies becomes especially important when exchanging data across system boundaries. Every time data crosses system boundaries, the target system must be prepared for the protection of incoming data, that is, the corresponding policies need to be deployed. The resulting negotiation of policies is also part of the policy management. As enforcement mechanisms can work differently (e.g., work on different system actions) on different systems or technologies, abstract policies can have different instantiations. Hence, usage policies must be instantiated on the target system.

2.6 Related Concepts

There are related concepts to cope with data sovereignty challenges. We present them in the following subsections.

2.6.1 Access Control

In general, data sovereignty challenges at the providing endpoint may be solved by access control technologies. As its name already implies, access control is suitable to handle the access to data, but has drawbacks in terms of data usages (as described in Section 2.2). However, a limited set of usage restrictions can also be handled by access control technologies.

2.6.2 Data Leak/Loss Prevention

Data Leak/Loss Prevention (DLP) technologies detects and prevents potential data breaches by monitoring sensitive data. Commonly used are Endpoint DLP solutions that run on the client’s operating system (e.g., as extension or feature of a security suite). In addition, there are also DLP solutions available that are monitoring the network or access to central storage devices.

2.6.3 Digital Rights Management

The term Digital Rights Management (DRM) is frequently used in the area of protecting digital content from unintended use, modification, and distribution. Different DRM technologies exist to protect multimedia content such as movies (e.g., DVD, Blu-ray), music (e.g., Audio CDs, Internet music), television, or E-books. In addition, there exist DRM technologies to protect digital documents (e.g., MS Word, PDF) within enterprises. This kind of DRM is also known as enterprise rights management (ERM) or information rights management (IRM) and aims to control of access and use of corporate documents.

2.6.4 User Managed Access

The purpose of User Managed Access (UMA) is to empower the resource owner to control the authorization of data sharing. It is often used to protect resources between online services on behalf of the owner. OAuth-based access management systems are representations of UMA. Several open-source implementations exist that follow the UMA core protocol.

2.6.5 Windows Information Protection

Microsoft introduced several technologies to establish a comprehensive information protection in their operating system and software such as Microsoft Office (e.g., BitLocker, Windows Information Protection (WIP), Office 365 and Azure Information Protection) [7]. WIP, for instance, is an integral part of Windows 10. Goals of the WIP are to protect data on own devices, to separate private and business data (data separation), to prevent unauthorized access and use (data leakage protection), and to protect data when shared. WIP-protected documents can only be used in WIP-compliant apps. For example, WIP prevents pasting sensitive information (e.g., by using ctrl+c and ctrl+v) to non WIP-compliant apps.

3. Implementation of Usage Control in the IDS

The chapter describes the implementation of data usage control in the IDS in general, starting with a brief overview about the two general activity streams. In addition, the chapter describes the policy language and its relation to the information model, IDS-specific usage restrictions and the different stages of usage control enforcement (named as usage control onion). Finally, the chapter addresses the relation of usage control to the reference architecture model.

3.1 Overview

There are two main activity streams within the IDS to implement data usage control:

First, a policy language to express data usage restrictions is developed. The policy language is descriptive, technology-independent and based on the Open Digital Rights Language (ODRL). To express usage restrictions within the IDS, there are several predefined policy classes that express the most commonly data usage restrictions. This activity stream is addressed in chapter 3.

Second, usage control technologies are developed to enforce these usage restrictions at technical level. We differentiate between proactive and retrospective technologies. The proactive technologies enforce provisions and obligations across system boundaries during runtime. It controls the data usages and is called preventive enforcement. The retrospective technologies are rather monitoring and recording technologies, but do not prevent or actively control any data usages. Therefore, it does not prevent undesired data usages and is called detective enforcement. The usage control technologies are described in chapter 4.

The following subsections addresses the IDS policy language and its relation to the information model as well as the IDS policy classes to express usage restrictions. Followed by a description about the different expansion stages of usage control enforcement: the usage control onion. The chapter ends with the integration of usage control to the reference architecture model. The policy language is not addressed in this document. We refer the interested reader to [8] and to [9] for ODRL-specific questions.

3.2 Information Model and Policy Language

Usage restrictions can be exchanged in an infinite amount of forms and models. From a legal point of view, verbally communicated agreements are perfectly fine and can be regarded as valid contracts. However, as proving the details of such an agreement is a challenge, usually textual contracts are of course preferred. Until now, this procedure has been sufficient and created the foundation for any business process. In a more and more digitalized world however, humans are not the only players anymore. As machines -- connectors in terms of the IDS -- take part in the processes, higher degrees of formalizations are necessary. As Figure 10 outlines, several stages are possible and all can provide value in different use cases. For instance, machine-readable policies reduce the degree of fuzziness of natural language texts and can be parsed and exchanged between actors, as e.g. XML or JSON. Formal policies extend that stage with clear definitions and implications of the used attributes and a consistent semantic. They allow, to some degree, already the inferencing of new information through logical axioms. Enforceable policies further specify distinct and deterministic criteria for each clause and can be transformed to executable code. As such, enforceable policies must connect the description of usage restrictions with infrastructure components and existing endpoints and unambiguous instructions. Autonomously negotiated policies on top reduce the human interaction to a minimum. Here, agents independently negotiate the usage restrictions based on predefined criteria and interpret and implement the policies on their own. Such processes in general also require an explicit understanding of usually implicit preferences but also of the relevant internal and external systems.

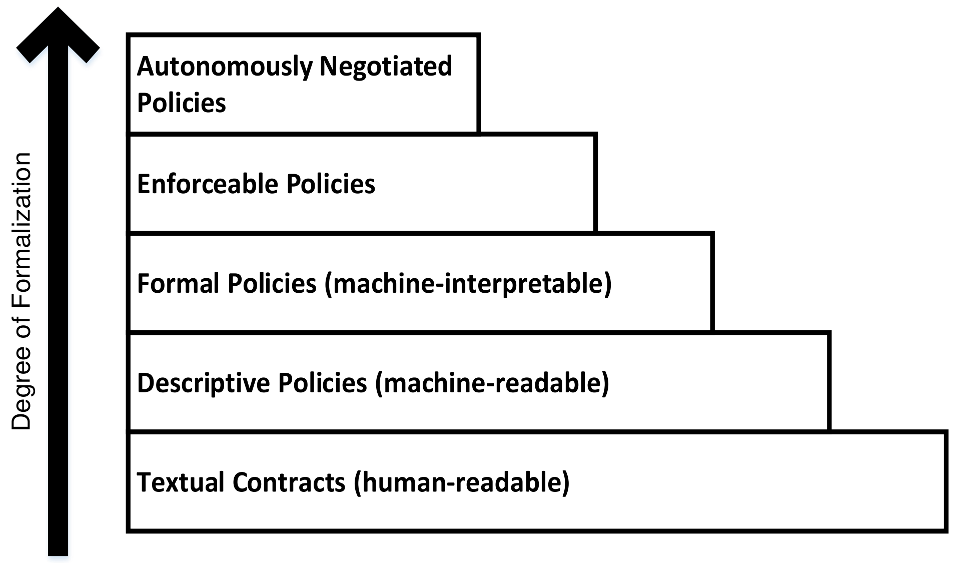

Degree of Formalization
Autonomously Negotiated Policies
Enforceable Policies
Formal Policies (machine-interpretable)
Descriptive Policies (machine-readable)
Textual Contracts (human-readable)

Figure 10: Relations between different policy categories

The selection of the appropriate format depends on the use case. Scenarios with high uncertainty, variety and valuable assets involved require more manual influence and should be located at the lower layers. Less heterogeneous use cases with high interactions rates can pay off the higher formalization effort through an increased automation. Furthermore, several usage control scenarios require at least formal policies in order to independently evaluate whether a usage request is appropriate or not. For instance in the SCRM example, the automated deletion after 14 days must not depend on a human action, but needs to be executed automatically.

Exchanging any usage control policy starting from the descriptive layer requires an explicit understanding of both, the form and the content of the policies. The syntax of the IDS Usage Policy language is therefore relying on the Resource Description Framework RDF (see [10], [11] for details). In the same way that RDF has several serialization formats, e.g. XML or JSON, any IDS Usage Policy can be transformed into any RDF compliant serialization and vice versa without loss of information. Regarding IDS messages, the determined serialization is JSON-LD. Still, the comprehensive modelling of the intended content of usage policies is challenging. In order to achieve a common understanding, the IDS promotes its Information Model (see Figure 11) as the core vocabulary and structure to encode semantic meaning. As such, the IDS Information Model specifies the shared terms, the supported structure and the relations between allowed patterns in the Commodity concern. More precise, this concern presents a profile of the Open Digital Rights Language version 2.2 (see [12], [9] for details). ODRL is a W3C recommendation and specifies a vocabulary and data model for the description of digital and machine-readable contracts. The IDS further extends ODRL towards usage control descriptions and enforcement, provides explanations regarding the compliant interpretation of constructs and defines implications for real-world implementations. This is accomplished in the form of IDS subclass constructs to the according ODRL classes. The design preserves the structure of the introduced terminology, resulting in the compliance of every IDS Usage Policy with ODRL recommendations. Nevertheless, not all ODRL terms are part of the IDS Usage Policy language as the requirements slightly differ. Even though this might change in future versions, it should not be expected implicitly.

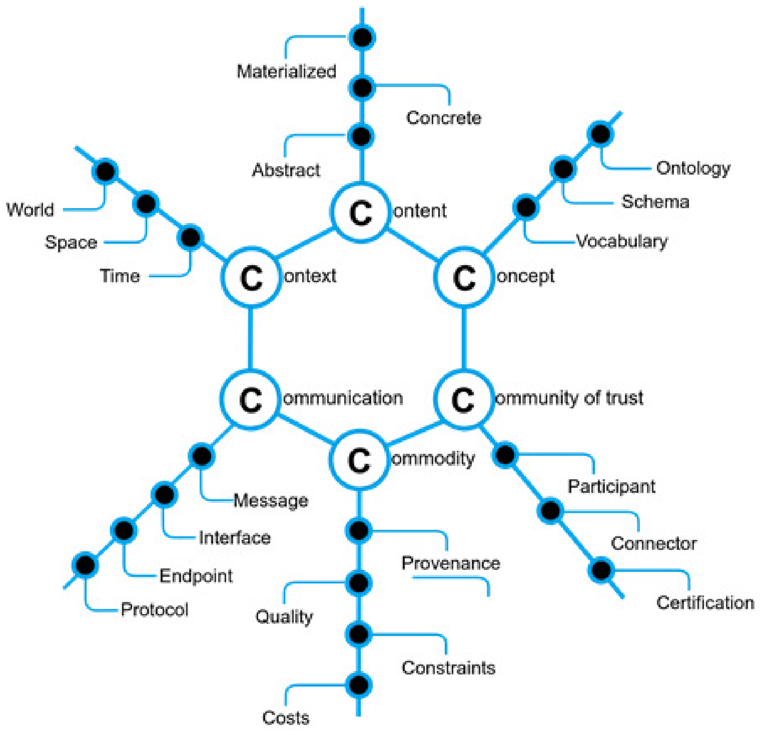


Figure 11: Concerns of the IDS Information Model. The concepts of the IDS Usage Policy Language are defined as part of the IDS ontology. It expresses the constraints in the commodity category.

The IDS Usage Policy Language does not support the full expressiveness of all possible ODRL compliant policies. The reason for this design decision is based on the nearly infinite number of possible combinations of ODRL instances. As the IDS needs to automatically evaluate restrictions in many cases, more fine-grained definitions are necessary. One example is the limitation to only two actors, one provider of any possible target resource and one consumer, which usually reimburses the provider for its effort. In order to formally encode the difference, the IDS Information Model mirrors the supported ODRL classes by own subclasses with further annotations and requirements. No IDS compliant connector can be required to understand plain ODRL constructs or terms but only their IDS counterparts. This is important to ensure a proper and unambiguous understanding between different IDS participants.

The fundamental building blocks for IDS Policies are the Contracts. Contracts present the container of any usage control statement and come in three different realizations: Requests, Offers, and Agreements (see Figure 12). While all share a similar syntax, their interpretation is slightly different. Requests indicate a desire to achieve a certain contract. Therefore, they must be regarded as a suggestion or a query, usually coming from a potential Data User. Requests have no binding character and are used before any actual usage control system is taken into account. Similar to Requests, Offers have only an informative meaning. Contract Offers present a potential willingness to interact under the specified conditions. Usually a Data Provider or Service Provider publishes Offers in order to signal its willingness to exchange data or services as outlined in Contract Offers. However, neither Requests nor Offers obligate the participants to any later commitments.

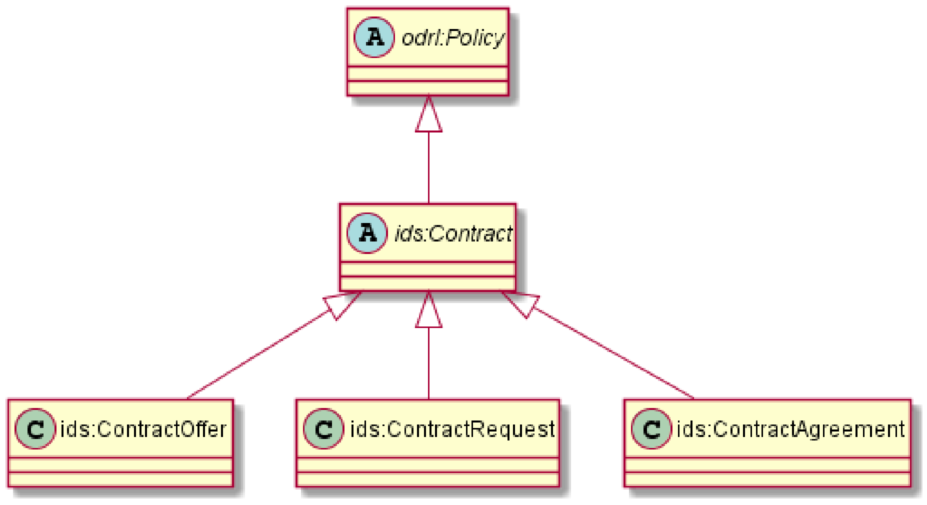


Figure 12: The IDS Information Model defines offers, requests and agreements as subclasses of the abstract contracts

In contrast, the exchange of valid Contract Agreements represents a binding and final consent to the stated constraints and requirements. An Agreement is the IDS terminology for a valid contract, which both sides accepted and therefore is binding as far as the IDS is related. As a result, only Contract Agreements must be regarded by any IDS usage control system. In Table 2, we listed all contract types together with their implications and descriptions.

Contract Type	**Implication**	**Description and Interpretation**
Contract Offer	The usage of a certain data resource might be possible regarding the stated constraints.	An offer is purely informative and voluntary metadata by the data provider. They give a rough idea on the usage restrictions and shall improve the discovery and selection process for Data Users. The Data Sovereign benefits by reaching a better visibility of its preferences.
Contract Request	The usage of a certain data resource is desired under the stated constraints.	The Data User indicates its interest, and may create the request relative to a previously exchanged Data Offer. The Data Sovereign gets to know acceptable constraints of the Data User while the later can further detail a contract.
Contact Agreement	Both participants conclude a contract and agree to the stated constraints. A later adjustment is not possible without a mutual consent.	The constraints have been fixed and accepted by both participants. The usage control systems import the agreement and enable or pre-vent access and usage accordingly.

Table 2: Contract types with their implications and descriptions

3.3 Policy Specification

An important step towards controlling the usage of the data is the policy specification. The Data Provider of the Industrial Data Space need to specify their Data Usage Control policies, although, they are from different technical backgrounds. A policy specification dashboard can support the customers in the process of policy specification. It offers various pre-filled templates that refer to the classes of Data Usage Control policies. The Data Providers can choose a template, fill it with the required information and receive the corresponding policy. Then, they can use a Data Usage Control application to enforce the policy into the system and respectively, protect their data.

The Industrial Data Space includes several use cases such as IDS Connector, Digital Supply Chain, Smart Urban Mobility, Intelligent Sensor, Interconnected ESN, etc. These use cases lead to variety of Data Usage Control policies. The policies are categorized into a set of classes in order to define the previously mentioned templates. Another advantage of classifying the policies, in addition to simplifying the policy specification process, is that the customers can estimate the coverage of their Data Usage Control policies.

3.3.1 Policy Classes

A Data Usage Control policy, in general, may provide permission to an IDS Data Consumer to operate specified action(s) over a Data Asset or prohibit the operation of that specified action(s). As well, a policy may require the operation of a specified action under specific circumstances. Providing permission or prohibition of an operation is extended to variety of actions. A policy can be specified to provide permission to use the data. The action of using the data covers various operations over that piece of data such as displaying it, printing it, making calculation over it, and so on. In addition, a policy may address only a particular fine-grained action. For example, a policy that permits reading data, allows the act of obtaining the Data Asset from the data source without further restrictions, however, the action of printing data is not permitted.

The Data Usage Control applications offer whitelisting, blacklisting or both approaches to protect the data. Thus, it is possible to specify policies that either allow or inhibit the data usage. Obviously, there might be conflicts among the specified Data Usage Control policies. Regardless of the whitelisting or blacklisting approach, a conflict detection and resolution strategy is needed. For example, while a policy provides permission to use the data, another policy might inhibit the data consumer to print the data. A conflict detection method must realize that the action of printing the data has been permitted once, likewise, it has been prohibited. A conflict resolution method in the context of Industrial Data Space might always decide to prohibit the action in a case of a conflict; however, the exact definitions of the data usage actions and the concepts of conflict detection and resolution are still evolving in the context of Industrial Data Space.

The studies on the requirements and use cases of the IDS project shows that several Data Usage Control policies are frequently used among the IDS stakeholders. We have categorized the policies into 14 classes that are explained in the following paragraphs.

3.3.1.1 Allow the usage of the data

This class of policy is an abstract category that gives permission to a specified IDS Data Consumer to operate specified action(s) on the Data Asset without further restrictions. As mentioned before, the action "use" is a very generic action that is utilized to express all targeted usages and therefore, includes fine-grained actions such as "read", "distribute", "print", "delete", "display", and so on. When the permission to “use” the data is issued, the data consumer is allowed to operate any of the aforementioned actions on the data. In order to restrict the type of the actions that are allowed to be operated, the policy must address a particular action. For example, in a whitelisting approach, you want to allow your data consumer to read and display the data, therefore, you specify a policy that only permits the “read” and “display” actions.

3.3.1.2 Inhibit the usage of the data

In contrast to the previous class of policy, this one is used to express that a specified IDS Data Consumer is not allowed to use the data. Similarly, there is no further condition specified and it is possible to address the “use” action or a fine-grained action. This class of policy can be used in a blacklisting or whitelisting approach. For example, you want to allow your Data Consumer to use your data but you want to prohibit few actions such as “distribute” and “print”.

3.3.1.3 Restrict the data usage for a group of users or systems

The Data Usage Control scenarios demand further restrictions on the policies that either allow or inhibit the usage of data. In order to apply the requested restrictions such as restricting the data usage into the specific users or systems, the corresponding policy conditions are specified. This implies that the usage of the data is permitted or prohibited when the specified conditions are met. In a policy, the conditions are indeed the prerequisite to operate the action. For example, you can instantiate a policy of this class that allows only a specified risk management system to use your data. This policy class faces few limitations, i.e., in order to evaluate the conditions, it requires that the systems and the users be certified. Thus, a data usage control application can validate the certifications and enforce the policy.

3.3.1.4 Restrict the data usage for specific purposes

This category represents another highly demanded class of policy that restricts the usage of data to specific purposes. In order to formulate the purpose of usage in a policy and later on, enforce it to the system, we need to define licenses and certifications. This concept is still evolving in the context of Industrial Data Spaces. “If the purpose is risk management, then allow the usage of data and else if the purpose is marketing, then inhibit the usage of data” is an example policy that is instantiated from this policy class.

3.3.1.5 Restrict the data usage when a specific event has occurred

This class of policy represents the permission or prohibition of using data under specific conditions; in the circumstances that the usage of data must be restricted due to the occurrences of specific events, a policy of this type can be constructed. Similar to the previous classes and in order to specify policies such as “if an accident occurred, provide permission to read the geographic location” or “provide permission to a data consumer to use the data during the exhibition”, we need to formulate the events. Therefore, a Data Provider can specify the conditions that address “when accident occurred” or “during the exhibition”. The assumption is that a set of possible events are defined in the context of Industrial Data Space and are available to the ones who specify the policies. As a result, a data usage control application is able to interpret the events and restrict the data usage accordingly.

3.3.1.6 Use or do not use the data in a specific time interval

The Industrial Data Space customers require further time-based constraints, i.e., restrict the usage of data into a specified time interval or a specified duration of time. A policy, for example, specifies the permission to use the data from the beginning of September 2019 to the end of November 2019. The date and time conditions can be expressed in different ways. However, it is important that the system is able to interpret the date and time conditions that are specified in the policies. For example, if “xsd:dateTime” is used as the data type that defines the date and time in the policy, the system must also be able to read it and understand it.

3.3.1.7 Use the data not more than N times

This class of policy demands to restrict the numeric count of executions of the action. For example, a policy specifies that the data can be printed only once or it can be displayed not more than ten times or in total, data cannot be used more than N times. We can only apply this kind of policies to the cases in which, the usage of data is countable. Therefore, a mechanism is needed that counts the usage of data in order to enforce the policy.

3.3.1.8 Use data and delete it after

This class of policy gives permission to a specified IDS Data Consumer to use the Data Asset and requires the Data Consumer to delete the data after. For example, a policy of this type may allow the data consumer to use the data for ten months (or days) and delete it afterwards. We can use the "`xsd:duration`" data type in order to specify the delay period before deleting the data in the policy.

3.3.1.9 Modify data (in transit)

In all aforementioned cases, the policies allow the users to use the entire data, without modifications, after the conditions are met. However, there might be cases where data must be modified or partially anonymized before it is allocated to the user. The data modification must be done before the permission to use the data is granted. This class of policy represents the Data Usage Control policies demanding to modify the data in transit; a Data Usage Control application intercepts the data that is transmitted and applies the modifications on them.

3.3.1.10 Modify data (in rest)

This class of policy demands for the data modifications or anonymizations before the permission to use the data is granted. In contrast to the previous policy class, it demands the modifications to be done when data is stored in a data-base. The Data Consumer is only allowed to use the data after certain modifications have been applied to the stored data.

3.3.1.11 Log the data usage information

The Industrial Data Space Data Provider requests to log the information of transferring data from their sites to their Data Consumer sites. Although, logging the information is a part of the Industrial Data Space infrastructure, a Data Usage Control application can occasionally apply the logging policies to the systems, as well. For example, it might log the information about the data anonymizations.

3.3.1.12 Notify a party or a specific group of users when the data is used

The studies show that the Industrial Data Space Data Providers request to be notified in a stated situation. For example, we can specify policies of this type to request to notify the Data Providers, when their data has left their sites or when it is delivered to the data consumers. This concept needs to be further developed in order to clarify the formats and possibilities of the notifications in the context of Industrial Data Spaces.

3.3.1.13 Share the data under specific circumstances

In most of the cases, a Data Provider specifies a policy to give permission to one or more data consumers to use the data. Although, there might be cases in which the Data Consumer requires permission to further distribute the data to other consumers or third parties. This class of policy exclusively addresses the conditions of data usage in case of sharing the data. For example, you can specify a policy of this type to demand your Data Consumer to share your data only if it is encrypted. The Data Provider may specify additional data usage policies to be provided to the third parties.

3.3.1.14 Restrictions on the fine-grained use actions

This class of policy addresses the policies that include action-specific refinements. For example, you want to provide permission to your Data Consumer to print the data, although, you want to refine the rendition resolution of the printing data. You can specify a policy of this type apply the resolution refinement on the “print” action. Defining a recipient for an “inform” action, or a delivery channel for a “distribute” action are other examples of this type of refinements.

3.3.1.15 Summary

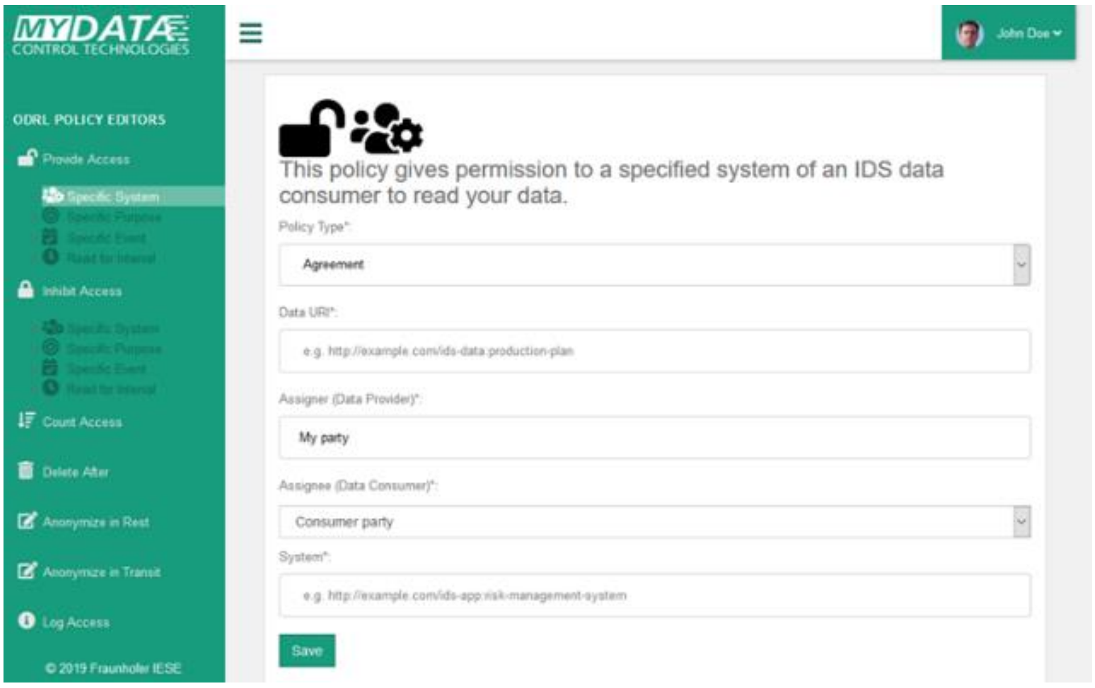
Overall, a policy can be an atomic instantiation of a policy class or can be constructed from several classes of policies, i.e., it is a combination of several atomic policies and the Data Usage Control policy classes assist the Data Provider to define their policy instances. These policies can be specific in ODRL language and later on, be translated to further technology dependent policy languages such as MYDATA, LUCON, etc. Furthermore, the above-mentioned policy classes may evolve over the time in the context of Industrial Data Spaces, depending on the stakeholders’ demands as well as public rules and regulations.

3.3.1.16 Policy Editor

A policy editor or in XACML terminology a Policy Administration Point (PAP) supports data owners and data providers in specifying their usage restrictions. Policy editors usually comprise a graphical user interface and offer different levels of guidance to the user, depending on knowledge and skill level. The IDS Lab offers an ODRL policy editor to express the aforementioned policy classes within the IDS. Interested reader can access the web user interface via the following link: https://odrl-pap.ids.isst.fraunhofer.de/

The policy editor (see Figure 13) is regularly updated. At the moment, it supports the IDS policy classes, the creating of ODRL policies for the above-mentioned fourteen policy classes that can be used within the information model.

In addition, it supports the transformation to machine-readable policies for the MYDATA usage control technology. The support for other technologies and further extension will follow.



This policy gives permission to a specified system of an IDS data consumer to read your data.

Policy Type*: Agreement

Data URI*: e.g. http://example.com/ids-data:production-plan

Assigner (Data Provider)*: My party

Assignee (Data Consumer)*: Consumer party

System*: e.g. http://example.com/ids-app:risk-management-system

Figure 13: Screenshot of the policy editor

3.3.1.17 Policy Transformation

In general, we can differentiate the degree of formalization in the process of specifying Data Usage Control policies. For example, a policy can be specified using natural language (e.g., “use my data only for billing and delete after fourteen days”). In the context of IDS, ODRL has been chosen as a common language to represent Specification Level Policies (SLP). In order to enforce these policies to the systems, the Data Usage Control applications must be able to transform the ODRL policies to their Implementation Level Policies (ILP). Therefore, ILPs have a direct mapping to technical enforcement. IDS uses different languages for ILP (e.g., MYDATA or LUCON Policy Language). These technology-dependent policy language are further explained in this document.

The policy transformation can be part of the policy editor (i.e., PAP) or part of the policy instantiation process. The latter is the most common case as the target system and environment of the Data Consumer is usually not known during the specification of the usage restriction at the Data Provider.

3.3.2 Policy Handshake and Negotiation

A negotiation process takes care of two aspects, the mapping of usage restrictions towards the internal system landscape and the potential bargaining of the usage conditions. The mapping itself targets the challenge of instantiating the stated requirements to decidable features according to the deployed systems. For instance, the data provider is usually not supposed to know the types and variants of the IT architecture of the data consumer. Furthermore, neither the data provider nor the data consumer are willing to reveal more information about their local settings than necessary. However, any automatically enforceable restriction must state the exact parameters which, through a binary decision process, deterministically conclude whether any possible action is allowed or not. Regarding the scenario, the maintenance service provider could need to store the incoming data in a relational database, and then process it on one of its clusters. As the plant operators want to limit the data spreading, the service provider can interpret their demand by stating that only these two systems shall be allowed to see the original data. At this point, the condition to not distribute the data is grounded and can be implemented in a local usage control system.

The second aspect of a negotiation step is the bargaining of the actual conditions. When the usage restrictions are specified, the requirements and preferences of the data user are usually unknown. Following a simple accept or reject pattern drastically reduces the amount of potential users and thereby reduces business opportunities. In addition, fixing obligations without knowing the context and implementation details of the potential usage is not reasonable as the information gap between specification and implementation time leads to unforeseen mismatches and conflicts. Therefore, an interested data user should be enabled to respond to an usage offering with a slightly adjusted counter offer. Still, it must be always in the authority of the data sovereign to accept or reject the request, or even make an additional offer regarding the details of the received counter offer.

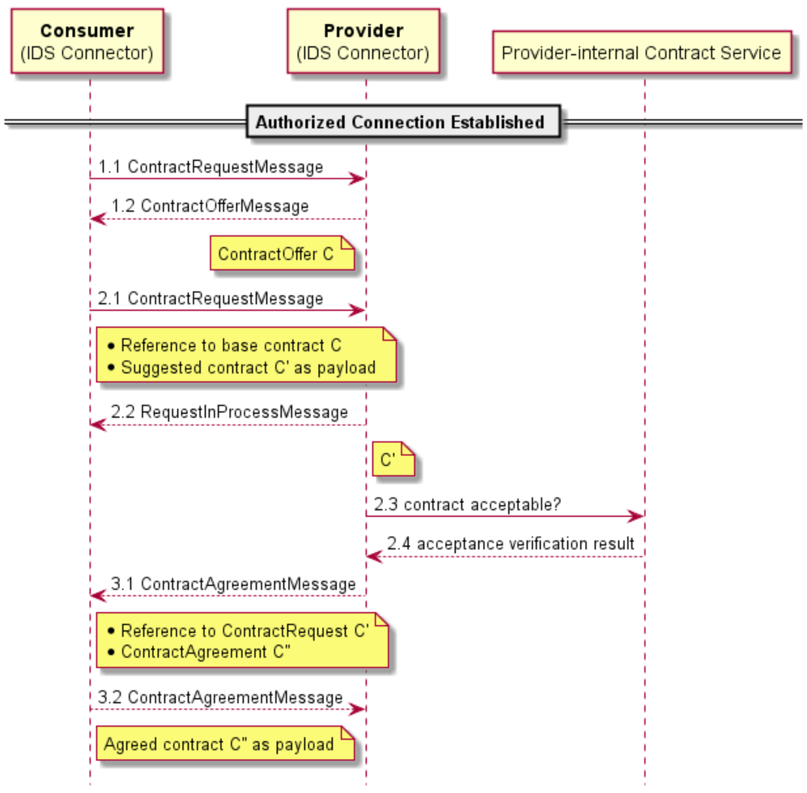

Consumer
(IDS Connector)
Provider
(IDS Connector)
Provider-internal Contract Service
Authorized Connection Established
1.1 ContractRequestMessage
1.2 ContractOfferMessage
ContractOffer C
2.1 ContractRequestMessage
• Reference to base contract C
• Suggested contract C' as payload
2.2 RequestInProcessMessage
C'
2.3 contract acceptable?
2.4 acceptance verification result
3.1 ContractAgreementMessage
• Reference to ContractRequest C'
• ContractAgreement C"
3.2 ContractAgreementMessage
Agreed contract C" as payload

Figure 14: Example policy negotiation handshake between a data consumer and a provider connector

A possible negotiation procedure is depicted in Figure 14 The Data Consumer, in the SCRM example the part supplier, would like to use a dataset that is accessible through the IDS. The OEM, here in the role of a Data Provider, restricts the usage to a certain amount of time. In this scenario, the Data Consumer announces its interest by creating a Contract Request as part of an IDS ContractRequestMessage and sends it to the providers IDS connector (1.1). The provider answers the request by sending a first ContractOfferMessage (1.2). In a second iteration, the Data Consumer reflects his gained knowledge of the previous Contract Offer and transforms it into a new proposal C', which better fits its preferences (2.1). For instance, one can easily imagine that the OEM wants to minimize the usage time in order to protect its data and therefore asks to only use the dataset for one week. The consumer, on the other hand, would like to gain more flexibility, and asks for a usage permission for at least one month.

Consequently, the Data Provider must decide whether it:

- rejects the request and terminates the interaction, , or
- responds with an adjusted Contract Offer, or
- accepts it and reacts with a Contract Agreement (3.1).

As this decision usually requires extensive knowledge about the context of the scenario and deep insights into the business logic, it is very challenging to automate. In most scenarios, the provider connector therefore acknowledges the second Contract Request Message by stating that some time is needed (2.2) and forwards the request to a human operator (2.3). Nevertheless, in case a proper business logic is in place, this task may also be completed by an autonomous agent or software module. After a decision has been made (2.4), the respective response message is sent. The IDS ensures the transparent linkage between the single messages through certain correlation message attributes in the message header but also references in the following contracts. Each message, contract, or any other resource and information entity has a unique URI, and is therefore identifiable throughout the whole negotiation process. This fact is also illustrated by the Contract Agreement shown below. It reflects the constraint to use the targeted dataset for only one month (from 01.12.2019 to 31.12.2019) as proposed in the Contract Request C'.

At this stage of the interaction process, it is important to note that even though the provider already signaled its general willing to allow a usage though its offers, it has still every right to withdraw its offers at any point. The IDS regards it as a successful interaction only after both parties have exchanged Contract Agreement Messages with exactly the same content (3.2). Furthermore, only the Data Provider has the authority to create a Contract Agreement object (3.1). The Contract Agreement C'' with the unique URI 'http://example.org/policy-id-1', represents the result of the described negotiation. In general, the iterations in such negotiations are not limited. Still, both connectors can terminate the interaction at any step and decide to reject the potential data exchange.

The following illustrates a time-restricted Usage Contract Agreement:

```
{
  "@context": "http://w3id.org/idsa/contexts/context.jsonld",
  "@type": "ids:ContractAgreement",
  "@id": "http://example.org/policy-id-1",
  "provider": "http://oem.com/ids#me",
  "consumer": "http://supplier.com/",
  [...]
  "ids:permission": {
     "@type" : "ids:Permission",
     "ids:targetArtifact" : "http://oem.com/ids/inventory/scrm-
dataset-1",
     "ids:action": "https://w3id.org/idsa/code/action/USE",

     "constraint":[
            {
              "@type" : "ids:Constraint",
              "ids:leftOperand": "ids:DATE_TIME",
              "ids:operator": "ids:gt",
              "ids:rightOperand": { "@value": "2019-12-
01T00:00:00+00:00",
              "@type": "http://www.w3.org/2001/XMLSchema#dateTime"}
            },
            {
              "@type" : "ids:Constraint",
              "ids:leftOperand": "ids:DATE_TIME",
              "ids:operator": "ids:lt",
              "ids:rightOperand": { "@value": "2019-12-
31T23:59:00+00:00",
              "@type": "http://www.w3.org/2001/XMLSchema#dateTime"}
            }
     ]
  }
}
```

Code 1: Time-restricted Usage Contract Agreement

3.4 The Usage Control Onion

We can characterize and implement the enforcement of data usage restrictions in different shapes. Organizational rules or legal contracts can be substituted or at least accompanied by technical solutions, which introduce a new level of security. Vice versa, technical solutions can be accompanied by organizational rules or legal contracts to support the overall goal achievement (e.g., to compensate missing capabilities of the technical solution).

Although it is a commonly used solution to solve usage control restrictions with organizational rules, we will focus on the technical enforcement in this document.

3.4.1 Technical enforcement, organizational rules and legal contracts

Usage control can be implemented in different ways. The solutions range from organizational rules or legal contracts to complete technical enforcement of usage restrictions. Intermediate levels may contain parts of both enforcement manifestations. We will describe a transition of enforcing usage restrictions from organizational rules/legal contracts to a complete technical enforcement that we align to our accompanying application scenario.

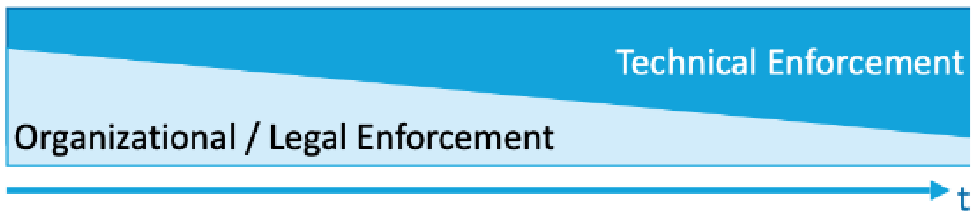


Figure 15: Technical Enforcement vs. Organizational Rules

Usage control should be seen as a machine-readable contract, which is expected to be fulfilled by a party. It is a way to track and trace data as it is used within different systems and to collect evidence of the violation of agreed usage constraints. With that in mind, solutions range from organizational rules or legal contracts to a completely technical enforcement of usage restrictions. For example, an organizational rule could state that the company rules forbid using removable storages such as USB sticks. Similarly, a technical enforcement such as group policies by the windows operating system can prevent the employees from using removable storage media. In some scenarios, we can interchangeably use organizational rules/legal contracts and technical enforcement. In other scenarios, we can use both enforcement forms to complete each other. In the long term, we assume a substitution of organizational rules/legal contracts by technical enforcement (as illustrated in Figure 15.

Although it is not in the focus of this document and for the sake of completeness, we can express usage restrictions as organizational rules or as part of a legal contract between two companies. In this case, we have no technical measure, but may enforce some violations by disciplinary penalty or lawsuit. Regarding our accompanying scenarios, there is a legal contract between the two companies stating that the exchanged data can only be used in the specific target systems (i.e., supplier management, risk management) or for the purpose of predictive maintenance. Furthermore, the contract may state that data must be deleted after a certain time. For violations, the contract states fines that are a multiple of the total contract value. In this case, organizational measures are applied to enforce usage restrictions.

The following presents the different stages of usage control that we name the usage control onion, starting from the inner part of the onion, which is the IDS connector, and ending in the outer onion shells with external systems.

3.4.2 Usage Control within the IDS Connector

The inner part of the usage control onion is the IDS connector. Depending on the usage restrictions, they are applied at the data provider connector or at the data consumer connector.

At the data provider connector, usage control enforces policies such as how often data can be accessed, at what times (e.g., only within business hours), or that data must be filtered or masked (e.g., anonymized) before leaving the company. The usage restrictions at data provider connector are usually provisions that are technically handled by a PEP.

At the data consumer connector, usage control enforces policies that are usually obligations for the data consumer such as "data can only be used for fourteen days" or "data can only be used for the purpose of predictive maintenance". The technical enforcement is handled by a PEP or PXP, depending on the usage restriction. Limiting data flowing to a specific target system to ensure the correct usage purpose is handled by a PEP, the deletion of data in storage infrastructure outside the connector is handled by a PXP that performs the delete operation.

3.4.3 Usage Control within the Storage Infrastructure

The usage control enforcement can also be implemented at the storage infrastructure. Storage infrastructure may be any kind of storage to persist data such as a file system or a database. In general, there are two possibilities for usage control capabilities at the storage infrastructure.

First, the storage infrastructure is used without modification. In this situation, usage control is implemented by encrypting the data within the connector before transferring the data to the storage infrastructure. Using the data is only possible by using the IDS connector to decrypt the data. Hence, every usage is controlled by the IDS connector. In such cases, usage restrictions such as data lifetime or time constraints can be enforced by deleting the cryptographic key material. There are no changes at the storage infrastructure needed, it can be used as is. However, every usage of the data must be handled by the IDS connector, which could lead to a bottleneck.

Second, the storage infrastructure is enriched with usage control enforcement component (i.e., PEP) that controls the usage of the data. It offers all usage control enforcement capabilities offered by PEPs, but demands changing or adapting the implementation of the storage infrastructure. However, such an adaptation may not be possible without support of the storage infrastructure vendor.

The two possible solutions are presented in the following Figure 16. Encryption and decryption is handled by the PEP in the connector (illustration on the left) or the usage of the data is controlled by the PEP inside the storage infrastructure (illustration on the right).

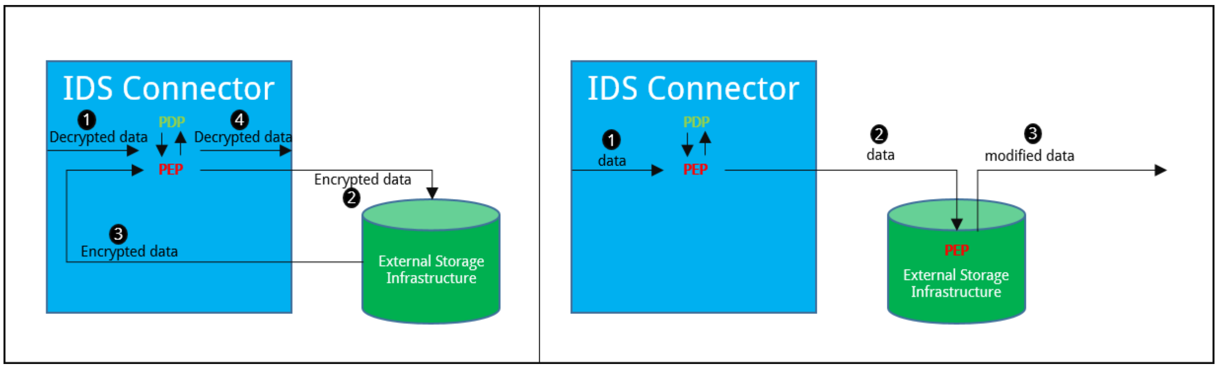


Figure 16: Usage Control within the Storage Infrastructure

3.4.4 Usage Control within the Storage Infrastructure

The next onion shell is the application that uses the data. Hence, the data flow within the application has to be controlled and adhere to the usage restrictions. Similar to the storage infrastructure, a PEP can be integrated into the application that controls the data flows. In addition, the application is developed by using technologies such as D° to directly integrate usage control capabilities at compile time. Both approaches have to address usage control in the development time and therefore need the support of the software vendor (as long as the software is not developed by the company itself or it is open source software that the company is willing to adapt).

3.4.5 Usage Control within the Clients

However, there are still possibilities how data may be used without adhering to the usage restrictions. For example, the user may print the data or take a screenshot. To tackle this issue, the client operating system such as Windows, Linux, Apple iOS, or Android has to be adapted. The integration of usage control capabilities is still possible, but demands deep knowledge about the operating system. In addition, not every operating system is open source. However, some vendors already started to implement security measures such as Windows Information Protection or the Android Enterprise (previously known as Android Device Administrator) that offer usage control capabilities (e.g., prevent printing, prevent screenshots, or prevent screen casting).

Finally, when data is flowing out of the Usage Control ecosystem, there are still possibilities how data may leak. For example, instead of making a screenshot, the users could take a picture of the screen by using a mobile device (i.e., external system or media disruption). Hence, we cannot achieve a perfect and comprehensive protection of data, but we can put controls to the system to reduce the possibilities for potential misuses.

To conclude, the enforcement of data usage restrictions can be characterized and implemented in different shapes. Organizational rules or legal contracts should be accompanied by technical solutions; and vice versa, technical solutions should be accompanied by organizational rules or legal contracts to support the overall goal achievement. In order to implement comprehensive usage control, we have to integrate control points into different systems and abstraction layers (see Figure 17) that are working together to achieve the overall goal of data sovereignty.

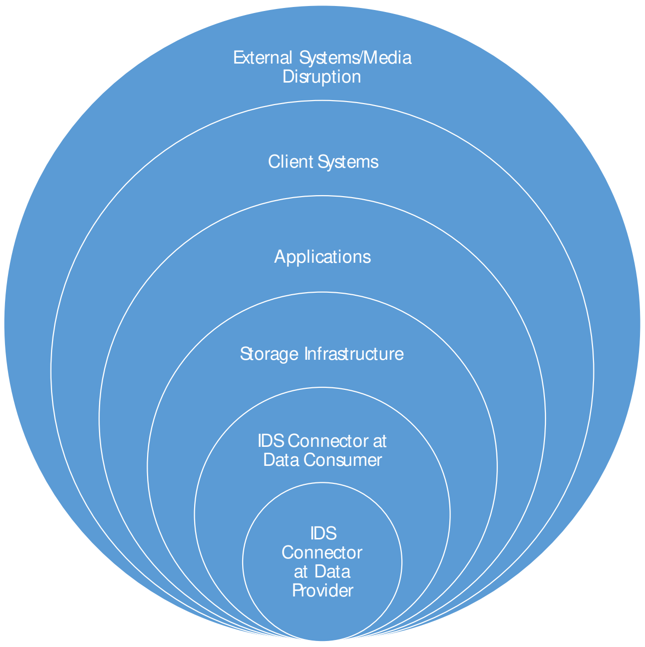

External Systems/Media Disruption
Client Systems
Applications
Storage Infrastructure
IDS Connector at Data Consumer
IDS Connector at Data Provider

Figure 17: Illustration of the Usage Control Onion

3.5 Integration with the Reference Architecture Model

The subsection addresses the integration of data usage control with the reference architecture model. In more detail, it describes the relation to other core components such as Identity Provider, Clearing House and Broker. In addition, roles such as data provider and data consumer are mentioned as well as context sources such as the DAPS.

3.5.1 Identity Provider

IDS identities for both connectors and participants are created, maintained and revoked through Identity Providers. These components serve as the interface from the preceding certification processes to the technical onboarding in any data space. By supplying X.509 certificates, the Identity Providers serve the fundamental proofs for identity claims. Any interaction with connectors providing incorrect, expired or otherwise compromised identity tokens must be terminated instantly. Therefore, the Identity Providers supply the most basic protection against misuse and are the most basic trust provider. One part of the Identity Provider is the Dynamic Attribute Provisioning Service (DAPS). In addition to the long-term identity certificates, DAPS tokens constitute short-term proofs of several security and usage control related features. For instance, the compliance to certain security requirements is encoded in its attributes. In case these characteristics change at a connector, for instance because a new update supplies higher security features, an updated DAPS token reflects this development. The valuable X509 certificate can stay untouched.

3.5.2 Clearing House

The Clearing House is a trusted third party, which protocols interactions and events in the IDS. As such, the Clearing House is the component, which logs negotiation results, access requests or usage events. More importantly, it is also supposed to get notifications about validations against usage constraints and store them in a secure manner. In addition, the Clearing House provides access to these notifications. This implies that the Clearing House is the perfect spot for any usage control system to log activities and document events. As this information also constitutes sensitive content, the Clearing House itself may provide information on its usage constraints and can publish those as Contract Offers. Furthermore, usage control systems may add usage restrictions to their notifications, either by adding the contracts to every interaction or by referring to a previously accepted one.

However, it is explicitly not the task of a Clearing House to decide on conflicts. It supports transparency to the IDS participants by storing notifications in an immutable storage backend system and provides trustworthy statements in cases of disputes or conflicts. As such, the Clearing House itself is not required to actually understand the stored information. Even further, in many use cases the participants may not want the Clearing House to even have a chance to gain insights into the details of their interactions while still being able to prove their existence. In such scenarios, the participants themselves need to store the content of events and only send technical proofs to the Clearing House. Encrypted data objects or hash values can serve as such. The Clearing House must accept any of such messages independent of their content as long as the sending participant satisfies the business terms of the Clearing House. As the operator can have a business interest, it can ask for according reimbursements. However, it is of course open to the operator to decide on its business model on its own.

As a trustworthy IDS component, the Clearing House must fulfill highest requirements regarding its reliability and credibility. It is one central trust anchor in the IDS and must not be compromised by any means. Therefore, it is the responsibility of the Clearing House operator to always guarantee the proper functionality and protection of its instance as otherwise all relying usage control mechanisms directly become compromised as well. Apart from the IDS specification and certification criteria, the Clearing House can implement additional security and reliability techniques. In order to protect its data, a distributed storage can be feasible. Moreover, a decentral storage, as for instance realized by BlockChain technologies, can provide benefits. The IDS specifications however do not define the internal functionalities.

In case of data provenance tracking (see section 5), information must be accessible via a centralized component. The Clearing House provides a Provenance Dashboard which returns a provenance graph for the unique identifier of a data asset. The DataFlowTracking and ProvenanceStorage-Component of each domain have to register at the Clearing House so that it is possible to aggregate data flows over several domains.

3.5.3 Broker

The IDS Broker is a purely informative component, intended to provide registry and discovery functionality to the IDS participants and components. The main task of the broker is the provisioning of search functionalities for desired data offerings. As such, a certain openness is necessary. However, in some scenarios the knowledge of the existence of certain resources might already impose a threat to the Data Sovereign. Therefore, certain broker implementations might be able to regard usage restrictions for the received metadata. For instance, a broker could hide the existence of a company's connector to its direct competitor even though it provides this information to the company's own supplier and customers. In contrast to Identity Providers or the Clearing House, no data protection mechanism must be expected by default. Nevertheless, some brokers may indicate such behavior through certified self-descriptions and signed attributes.

3.5.4 Data Provider, Data Owner and Data Consumer

A Data Owner is responsible to specify usage restrictions in ODRL. The Data Provider has to attach the usage restrictions as part of the information model to the data to be shared within the IDS. Data Owner and Data Provider may be the same party, but must not be necessarily. In an ODRL policy, the Data Provider is called Assigner.

The Data Consumer is the Assignee in an ODRL policy and responsible to ensure the adherence to the usage restrictions.

3.5.5 Context Sources

The provisioning of context information is essential in order to integrate external events or states into the usage control process. The IDS Reference Architecture does not explicitly define a general PIP component. However, certain security- and participant-related features are supplied by the Identity Provider, more specific through the Participant Information Registry (PIR) and the Dynamic Attribute Provisioning Service (DAPS).

Some usage control frameworks offering context information by default, like date and time. If there are standardized Policy Information Points, it enables a connector to process various context information. However, there is the open question, how trustworthy this information is, and how it could be made trustworthy. An example would be a connector app with an information point that offers the purpose of this app. The knowledge could be used to apply a policy, which only allows the use of data for a specific purpose. Therefore, the data is just transferred if the purpose of the app fits.

4. Usage Control Technologies

In the first part of this section each Usage Control technology available in the IDS is presented in detail. Every technology section covers the topics of communication flows, integration concepts including their characteristics and how the necessary Usage Control permissions and obligations are extracted out of the IDS ODRL policies. At the end of this section, all IDS Usage Control technologies are compared with respect to their general characteristics, their capabilities and policy languages as well as their location of implementation. A discussion closes the chapter.

4.1 MYDATA Control Technologies

MYDATA Control Technologies (MYDATA for short) is a technical implementation of data sovereignty, which represents an essential component for informational self-determination. It is based on the IND2UCE framework for data usage control developed at Fraunhofer IESE.

In general, MYDATA implements data sovereignty by monitoring or intercepting security-relevant data flows. This enables fine-grained masking and filtering of data flows in order to make them anonymous, for example. Compared to classical access control systems, MYDATA can enforce partial filtering and masking of data, context and situation restrictions as well as restrictions on the purpose of use.

4.1.1 Communication Flow and Integration Concepts

The overall communication flow from a Data Source at the Data Provider to a Data Sink at the Data Consumer is illustrated in Figure overall-flow. There are several data sources at the Data Provider side that can be connected to the IDS Connector (e.g., database, file system, application). The dataflow starts at the Data Source and is handled by the core container in the IDS Connector. As part of the routing and before data is flowing to the Data Consumer, the Usage Control container is invoked and Data Provider specific Usage Control rules are applied to the data (e.g., remove person-related information). Data is then sent to the IDS Connector at the Data Consumer. The same procedure is happening within this connector, hence, the Usage Control container is also invoked as part of the data routing and applies the Data Consumer specific Usage Control rules (e.g., data is only allowed to be used by a specific target system). Data Sources and Data Sinks can be located inside the IDS Connector (e.g., data apps) or outside the IDS Connector (as depicted in Figure 18). If data resists within the IDS Connector, Usage Control can be easily achieved.

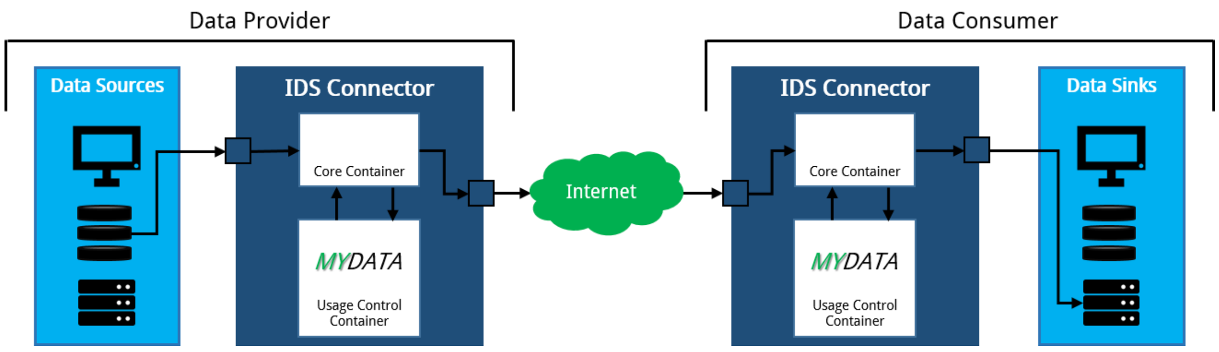


Figure 18: MYDATA Communication Flow and Integration Concepts

Hence, Usage Control can be separated into Usage Control at the Data Provider Side and Usage Control at the Data Consumer Side. Basically, MYDATA can be integrated as part of the message routing or as interceptor pattern. In the following, we will differentiate these two approaches and explain them in more detail.

4.1.1.1 Usage Control at the Data Provider Side

The Usage Control at Data Provider Side is applied whenever data is processed by the IDS Connector. The core container is responsible to handle the data processing. We use Apache Camel as message-oriented middleware that handles the message router in the following. To integrate Data Usage Control into the data processing of the IDS Connector, we enforce that all messages are routed to the Usage Control container. In doing so, we are able to control any data that is processed by the IDS Connector. To ensure full data control, the Usage Control container has to be invoked last before data is leaving the IDS Connector at Data Provider Side. However, the Usage Control container may additionally invoked between other processing steps of the IDS Connector. For example, before or after data is processed by Apps that are running within the IDS Connector. For the sake of simplicity, we will not address this kind of Usage Control enforcement at this point.

4.1.1.2 Usage Control at the Data Consumer Side

If an IDS Connector is consuming data, the simplest case is the forwarding of the received data to a Data Sink at the infrastructure of the Data Consumer. Similar to the Data Sources, the Data Sinks can be any kind of system (e.g., application, database). Contrary to the Usage Control at the Data Provider Side, we have chosen to implement the Usage Control at the Data Consumer Side as interceptor pattern. In this way, we can hook into every message routing step and apply the Usage Control rules to the data flow.

In summary, Usage Control at the Data Provider Side as handled by message routing as part of the route configuration, Usage Control at the Data Consumer Side is handled by message interception that is part of the IDS Connector implementation. The two integration concepts are depicted in Figure 19.

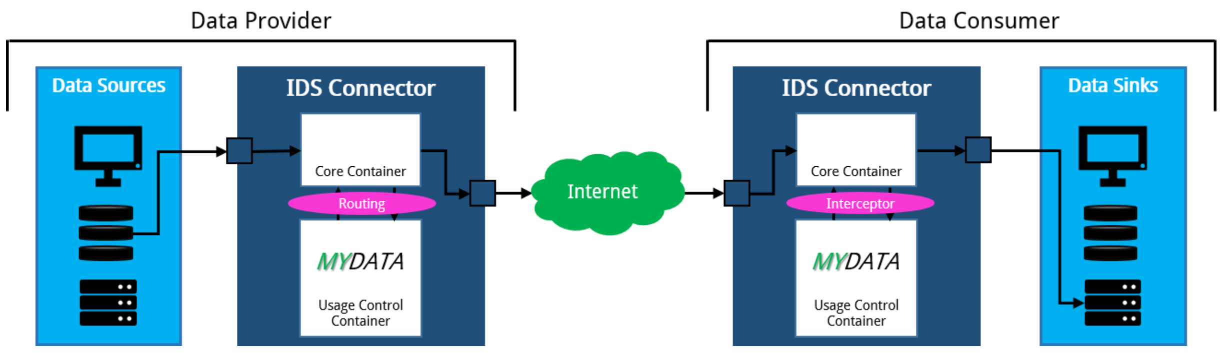


Figure 19: Routing and Interceptor Approach

In 3.4.3, we presented two ways of storing data at storage infrastructure at the Data Consumer. However, there is also the possibility to store the data within the IDS Connector. In this case, the communication to and from the internal storage is handled by the Core Container, which uses the interceptor pattern to control the data. Hence, the Usage Control enforcement is handled analogously.

4.1.2 Detailed Integration Concepts

We presented the two integration concepts in the previous sections. We will have a closer look to them in the following. However, before we start, we will explain some more details.

First, we will explain the Usage Control Container. The Usage Control Container contains at least a PEP, a PDP and a PMP. It offers interfaces for calling the PEP interfaces and the PMP interfaces. Invoking the Usage Control container, as used in the previous sections, results in invoking the PEP within the container (following the processing explained in Figure 20). In addition, there is the possibility to deploy machine-readable policies by calling the PMP interfaces.

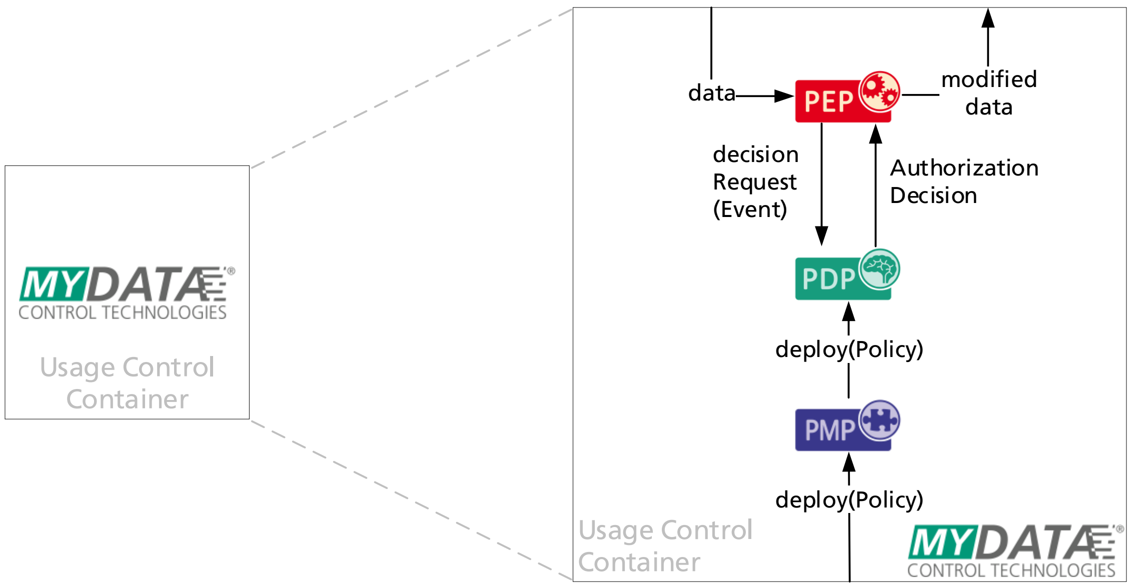


Figure 20: MYDATA PEP invocation and processing by the PDP

Second, we explain the message routing of a message-oriented middleware such as Apache Camel. The middleware coordinate the data flow between different systems and applications. From a technical point of view, Apache Camel provides this by a routing data between different nodes. In a nutshell, it passes the output from one node and puts it as input to the next node. The routing may comprise several nodes as depicted in Figure 21. Data Source and Data Sink are named Endpoints in Apache Camel terminology, every node in such a route is named a processor.

Third, the interceptor pattern is used to invoke or handle some processing of the output or the input data before and after every each node. The interceptor is implicitly called contrary to the message routing, which has to be explicitly configured.

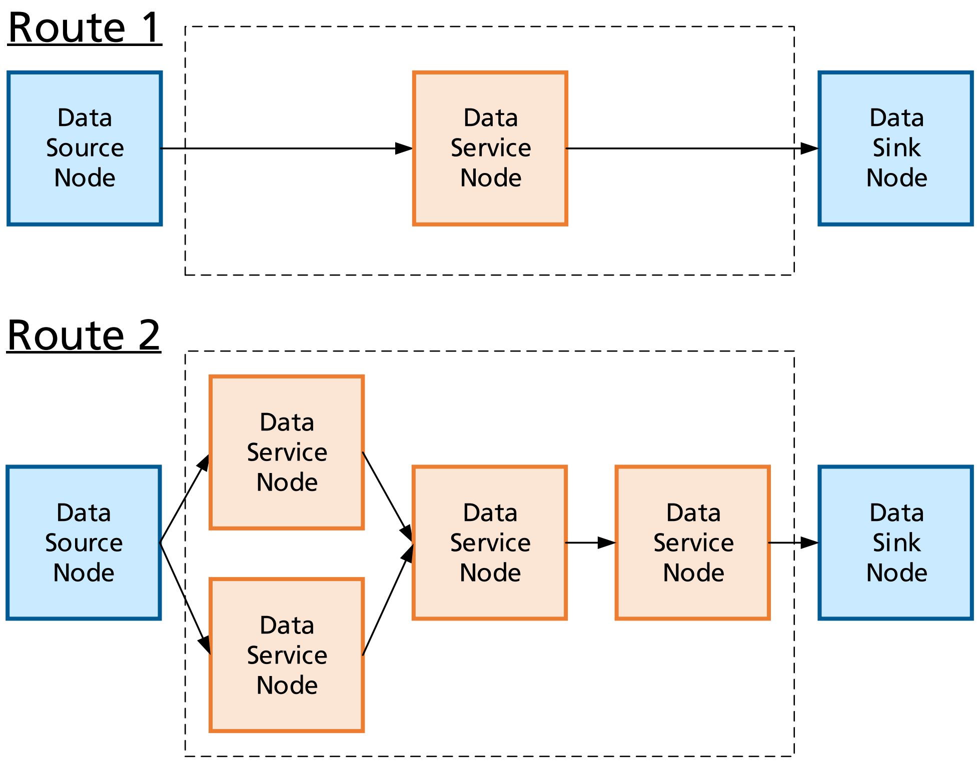


Figure 21: Routing example without interceptor

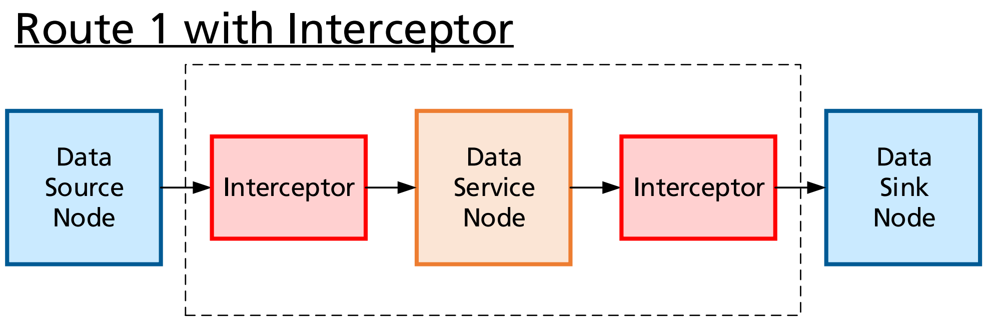


Figure 22: Routing example with interceptor

We use "Route 1" for explaining the different approaches to integrate Usage Control with MYDATA (see illustrations in Figure 21 and Figure 22). See also Table 3 for details.

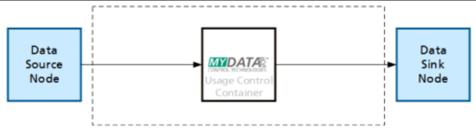
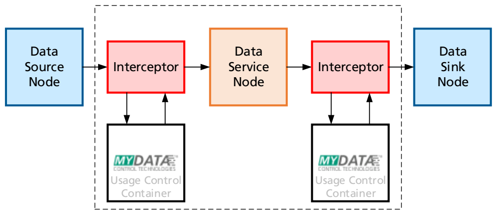
Usage Control by Routing	Usage Control by Interceptor
 Data Source Node; MYDATA Usage Control Container; Data Sink Node	 Data Source Node; Interceptor; Data Service Node; Interceptor; Data Sink Node; MYDATA Usage Control Container; MYDATA Usage Control Container
If we integrate the Usage Control by Routing concept to control the usage of data, the Usage Control container has to be explicitly configured as a processor in the route. When using this approach, it is important to consider where the processor is located within the route. For example, in case of counting it should be the last processor in the route before data is leaving the organization. Contrary, if you want to prevent the processing of person-related data, it should be the first processor in the route. In case an Apache Camel Processor is used, the Processor delegates its call to the Usage Control container.	If we integrate the Usage Control by Interceptor concept to control the usage of data, the Usage Control container is implicitly called between every processing step. Apache Camel offers the possibility to integrate interceptors that it executes every time before and after a processor is working. In this approach, MYDATA is completely integrated into the Camel Interceptor and forwards every interception as a decision request to the PDP. In case an Apache Camel Interceptor is used, the Interceptor internally delegates its call to the Usage Control container.

Table 3: Comparison of Usage Control by Routing and Usage Control by Interceptor

4.1.3 Comparison and Discussion of the Integration Concepts

The decision which approach to use depends on the requirements to be fulfilled by Usage Control. For example, integrating the Apache Camel Interceptor approach leads to a higher performance impact than implementing as Apache Camel Processor (due to the (probably) fewer number of interceptions). On the other side, implementing the Usage Control container only at one place in the route lowers the security as the messages are only intercepted at that one point in the route. This kind of approach may be suitable for the Data Provider Side, but probably not for the Data Consumer Side, because data entering the route can be processed by apps without being checked by the Usage Control container. At the Data Consumer Side, a Data Provider probably wants the Usage Control container to check the data usage before and after every app processor (e.g., data apps). When implementing the Usage Control by Routing, the developer has to select suitable location(s) so that the usage Control container is able to enforce all required Usage Control policies. Table 4 summarizes the comparison in a short form.

	Independent Development	**Additional Component**	**Performance Impact**	**Suitable for Consumer/ Provider**
Usage Control by Interceptor	partly (Usage Control container independently; integration with Camel needed for the interceptor)	yes	High	Consumer/ Provider

Usage Control by Routing	yes	yes	Low	If Provider (preferred), Consumer (partly, in exceptions)

Table 4: Comparison of integration concepts

4.1.4 Transformation of ODRL policies

The focus of this chapter is on MYDATA policy language and the transformation of an ODRL Policy to a MYDATA Policy. (As mentioned before, the ODRL language has been chosen as a standard language in the context of IDS project.) The ODRL policies [13] represent statements about the usage of the contents and services. An ODRL Policy consists of at least one rule and it is specified with a unique identifier. The ODRL Target, ODRL Action, ODRL Assigner and ODRL Assignee are the main elements of an ODRL Rule. The ODRL Target addresses the data asset to be protected. The ODRL Action is the type of access to the data asset. Example of actions are read, display, delete, use, etc. The ODRL Assigner and the ODRL assignee are the parties that issue the rule and receive the rule, respectively. An ODRL Rule is the permission, prohibition or obligation of operating one or more Actions over a specified Target asset. Moreover, The ODRL Constraints are Boolean or logical expressions that refine the semantics of the Actions, Parties and Assets, although, it might declare the conditions applicable to a Rule.

Likewise, a MYDATA Policy consists of one or more mechanisms and it is specified with a policy identifier. The MYDATA Mechanisms follow the if-then-else schema. The MYDATA Event, MYDATA Solution, MYDATA Decision, PIP, PXP and MYDATA Modify are the main elements of a MYDATA Mechanism. A MYDATA Event is an attribute of a MYDATA Mechanism. An Event is a hooking point of a system in which data is used. Additionally, it is a point of a system in which data must be anonymized, deleted and so on, i.e. protected. It contains a name, the time that it has occurred and a list of key-value parameters.

A MYDATA Solution is a site in which the MYDATA technology is used to protect their data. A site can be a company, an organization, an end user device or an IDS Connector. The MYDATA Solution identifier represents the site's name. A MYDATA Decision is an authorization decision that is specified in the then-block of a MYDATA Mechanism. It can be specified as "allow" or as "inhibit" in order to permit or prohibit the occurrence of the corresponding event, respectively. Moreover, a MYDATA Decision can be an execution of a PXP or an application of a MYDATA Modify. A PXP executes a requested action on the occurrence of an event. A MYDATA Modify anonymizes the flowing data according to the user's preferences. The data is given as an Event parameter.

A PIP returns a requested value from an external information resource. It is mostly used to indicate and evaluate a condition. A MYDATA Condition is specified in the if-block and leads to a specified MYDATA Decision, when it matches.

The MYDATA Policy and MYDATA Mechanism are the equivalent concepts for the ODRL Policy and ODRL Rule, respectively gives an overview about comparable elements in ODRL and MYDATA policy languages.

	MYDATA	**Description**
Policy uid	Policy pid	A policy must have a unique identifier in ODRL and MYDATA policy languages.

Rule	Mechanism (Decision)	An ODRL policy consists of one or more rules. Similarly, a MYDATA policy consists of one or more mechanisms. The ODRL rules and MYDATA mechanisms reflect a decision for specific usage of data.
Rule Permission	Decision Allow	A permission allows access and/or usage of a data asset with further specifications in the form of constraints or connected duties. In order to allow the usage of data, a corresponding permission rule must exist.
Rule Prohibition	Decision Inhibit	A MYDATA inhibit decision represents an ODRL prohibition rule.
Rule Obligation	Execute (PXP)	A MYDATA execute action (PXP) represents an ODRL obligation or duty. In addition, ODRL allows the declaration of consequences for not fulfilling an ODRL duty. Currently, the relevant concepts to the ODRL consequences are not defined in the IDS context.
Assigner	Solution	An assigner is a party that issues a rule. In a provider-side policy, the assigner exposes the enforcement site.
Assignee	Solution	An assignee is a party that receives a rule. In a consumer-side policy, the assignee exposes the enforcement site.
Target	Event Parameter, PIP condition	An ODRL target represents a data asset. In MYDATA, the data can be addressed as an event parameter or can be examined using a PIP.
Action	Event	A MYDATA event is a representation for an ODRL action which defines an operation on a data asset and also, clarifies where the policy must be enforced.
Duty Action	Execute (PXP)	Duties frame any type of obligations connected to the usage of a data asset. When the operation of a duty is about anonymizing the data, a MYDATA modifier can be used instead of a PXP. It is because MYDATA has specific tag for anonymizing data in transit.
Left Operand Delay Period	Timer, PIP condition	The MYDATA timer can be used to represent a delay period. For example, when a delay period attribute is set to 5 days, we translate it to a MYDATA timer that sends a corresponding event daily. In addition, we need a PIP that on each day, evaluates whether 5 days has passed since the starting point.
Left Operand System	PIP condition	Most of the ODRL constraint left operands such as system, purpose, etc., can be represented as a MYDATA PIP condition.
Left Operand Purpose	PIP condition	Most of the ODRL constraint left operands such as system, purpose, etc., can be represented as a MYDATA PIP condition.
Left operand Date Time	Date and Time condition	In MYDATA, the date and time functions can be used to specify a condition that represents an ODRL constraint with a date time left operand.
Left Operand Event	Count condition	A MYDATA count function can be used to check if a specified event happened at least once (in this hour).
Left Operand Count	Count condition	The MYDATA count function counts the occurrences of a specified MYDATA event (ODRL action).

Table 5: ODRL equivalents in the MYDATA language ODRL

Furthermore, in order to transform an ODRL Rule to a MYDATA Mechanism, one can consider the MYDATA policy as a template, and fill the relevant elements in it one by one starting from the authorization decision.

We assume that the ODRL Assignee exposes the enforcement site and therefore it reveals the MYDATA Solution. It must be considered that the ODRL Rules with different assignee parties can be collected to one single ODRL Policy. However, it is not possible to collect MYDATA Mechanisms with different solutions into one single MYDATA Policy and therefore, we need to split them to more than one MYDATA Policies.

For more information, refer to the document [8].

4.1.5 Policy Instantiation

For the policy instantiation, we differentiate again between Data Provider and Data Consumer. The Data Provider can directly instantiate the data usage restrictions in the Usage Control container. For MYDATA, this is done by calling the deploy policy interface of the MYDATA management service. The procedure at the Data Consumer Side is similar, but the Data Consumer receives the usage restrictions as part of the policy negotiation in ODRL format as part of the Information Model. Hence, the Data Consumer has to transform the ODRL policies to machine-readable policies and then call the deploy policy interface of their technical Usage Control enforcement (analogously to the instantiation at the Data Provider Side).

The policy negotiation and the policy instantiation must have been successfully completed before any content is sent from the Data Provider to the Data Consumer.

4.2 Logic-based Usage Control (LUCON)

LUCON (Logic based Usage CONtrol) is a policy language for controlling data flows between endpoints. The Trusted Connector uses Apache Camel to route messages between services (such as MQTT, REST, or OPC-UA endpoints). The ways how messages may be processed and passed around between services is controlled by LUCON, a simple policy language for message labeling and taint tracking. The LUCON policy language comes with an Eclipse plugin for syntax highlighting, code completion and compilation into a format that is understood by the policy decision point within the Connector. Thus the typical workflow is as follows:

1. Write a LUCON policy in Eclipse (Install Eclipse Plugin). As you type, Eclipse compiles the policy into a .pl (Prolog) file in the compiled-policies folder.
2. Load the compiled policy into the Connector & activate it

Security goals (c.f. section 2.2) LUCON can help to achieve are:

- **Secrecy**
  By defining a policy that disallows forwarding of labelled data to nodes that are explicitly marked as trusted in that context.
- **Integrity**
  Integrity can be monitored by comparing in- and output of a node.
- **Time to live**
  A state monitoring number of usages in a workflow can be defined to track allowed number of usages.
- **Anonymization by aggregation**
  A policy can be defined that specifies how many samples need to be aggregated before an aggregation may be forwarded. This functionality can be wrapped inside a microservice (e.g., aggregating five samples each) and this service then gains the according property.
- **Anonymization by replacement**
  This is a task that needs to be wrapped inside a microservice. This service container

could then even be certified to attest its functionality. The service then gains the property to satisfy the anonymization policy.

- **Separation of Duty**
  A policy can be specified that disallows conflicting labels.
- **Usage scope**
  A policy can be defined that disallows data with a specific label to leave the connector.

Two examples illustrate how this can be applied to real world problems:

Example 1: Automotive Supplier and OEM must restrict forwarding of data and limit data processing
An automotive supplier is providing valuable data about its current production capacities to an OEM. This information is extremely helpful for the OEM as it allows realtime and precise prediction of logistic chains, alignment of purchase strategies, and automated planning of production capacities at the OEM's side. However, for the supplier this information is highly critical and while it is willing to share it with the OEM in the context of a bilateral agreement, it needs to ensure that the data is only used for the agreed analytics and never shared with any competitor. Access control cannot solve this problem as it is only able to decide whether the OEM should be granted access to the data or not. With usage control on trusted endpoints, however, the supplier can state the following requirements and have them enforced at the OEM's side:

- Data must only flow into the known (and trusted) analytics applications
- If data flows from the Trusted Connector at the OEM's side into any other (untrusted) endpoint, the supplier wants to be informed about this event so evidence of the violation is created and legal actions can be taken.

Example 2: Enforcing Data Protection Regulations in Health Care Applications
A company is processing patient records for the sake of accounting and billing as a service to doctors and insurances. In addition, it is running data analytics as a service to the healthcare industry to assess drug sales in certain regions and support planning of drug productions and logistics. Strict legal regulations on personal identifiable information (PII) such as the German Bundesdatenschutzgesetz and the EU-GDPR require that PII must only be used for the purpose the user consented to. It is thus in the interest of the company to ensure (and potentially prove in an auditable way) that it complies to those regulations and is not leaking PII to the health care industry in the context of its analytics services. With usage control, the company can express the following requirements and have them automatically enforced within the Trusted Connector:

- Incoming raw data from patient records is marked as PII
- PII must not leak to any endpoint except insurances
- PII only becomes uncritical, if at an aggregation of least 2 data fields with at least 200 individual records with at least 3 undistinguishable is calculated
- Such uncritical data may be sent to any other public endpoint

4.2.1 Communication Flows and Integration Concepts

The policy set is translated into a rule set that specifies how data is handled while being kept in the consumer connector. LUCON is integrated into the "Trusted Connector". It could be used in other Camel based implementations. However, without accompanying security means, Usage Control with technical enforcement is hardly sensible.

The Trusted Connector is based on Apache Camel as a routing engine. It is possible to define routes between data sources and data sinks. A route can contain multiple steps and even deviations, forks and aggregations. In this example (see Figure 23) we illustrate a simple route. Data is pulled from a data source and forwarded to "Data Service Node 1". The output is forwarded to "Data Service Node 2". That output is finally sent to a data sink (e.g., another connector).

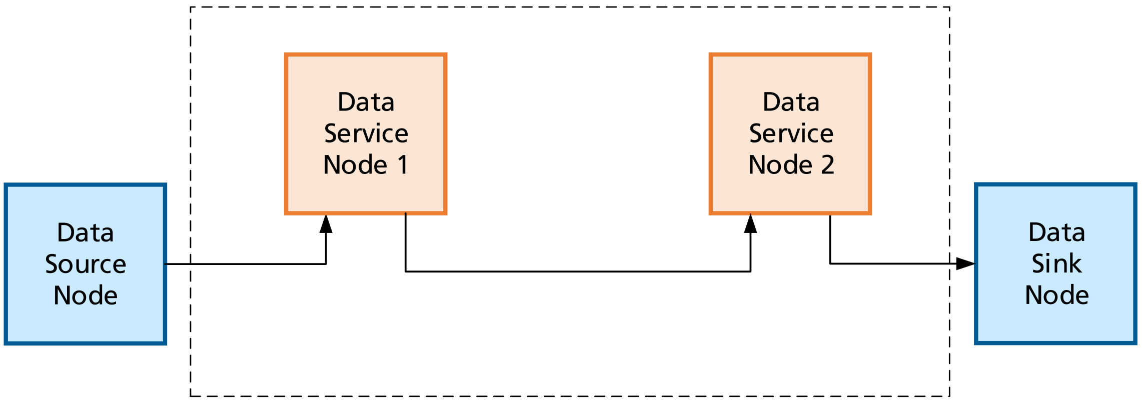


Figure 23: LUCON Camel route without interceptor

Apache Camel provides means to configure an Interceptor. This means, that a Java class is called whenever an Exchange object (the Camel wrapper for a message) leaves or enters a node. This is why the Interceptor is called twice between "Data Service Node 1" and Data Service Node 2" (see Figure 24).

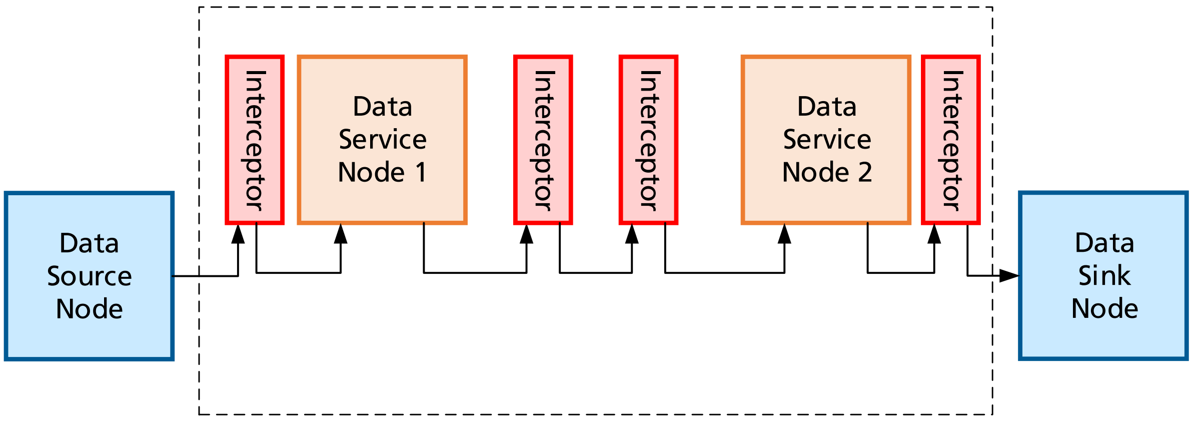


Figure 24: LUCON Camel route with interceptors

The integration is transparent for the administrator of data routes and defined policies can be seen as an allowed corridor for data routes. The overall design of communication flow is similar to that of MYDATA. The concept, however, is based on information flow control.

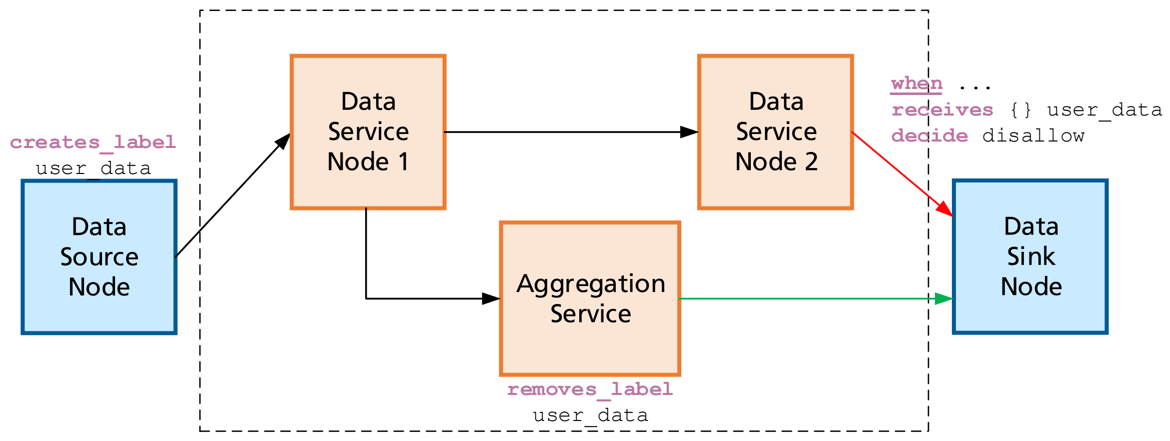


Figure 25: LUCON Camel route with aggregation service

If we take a look at an exemplary route, we can illustrate the usage of labels. For incoming data items from "Data Source Node", a label "user_data" is attached. This label is only removed if a service that provides the required anonymization or aggregation functionality (as the "Aggregation Service" does). The Data Sink is not allowed to receive data items that are labelled "user_data". Thus, no unaggregated user information may leave the connector (see Figure 25). This kind of policy set could either be deployed on data provider side or as well on data consumer side.

4.2.2 Transformation of ODRL policies

LUCON is embedded into the Trusted Connector software stack. Policies can be submitted in LUCON Domain Specific Language or soon via ODRL. Since LUCON focuses on information flow control, only a subset of ODRL policies will be supported. This makes it reasonable to generate ODRL templates with a specific scope, supported by an editor. ODRL policies need to be processed and translated into valid LUCON policies locally. The policy editor which is used for MYDATA is forked to support templating for valid ODRL policies. This way, all solutions support a similar user experience.

4.3 Degree (D°)

While LUCON and MYDATA aim at providing usage control for existing applications and workflows, D° takes another approach. D° is a Domain Specific Language (DSL) for the development of data processing applications (so called Data Apps) and takes usage control into account from the beginning of the development. D° uses Java as host language. Through the use of Model Driven Software Development (MDSD) Data Apps which are developed with D° are transformed into Java applications which are finally compiled into executable applications.

It is important to notice that D° is not based on/an implementation of the XACML architecture. Nevertheless XACML is referenced at various points in this chapter to adapt the general structure of this document, which has some kind of focus on XACML. Even though functionalities of D° are mapped to XACML constructs through this document, it is important to keep in mind that D° is a programming language which produces applications which have most of the components directly integrated. These applications do not exist separately from these components. Therefore the described components are not separate services or even servers which is a big difference to the XACML architecture.

Before detailed information about D° and its components is given, the following picture (Figure 26) shows the big picture of D° with all relevant components and their relationships.

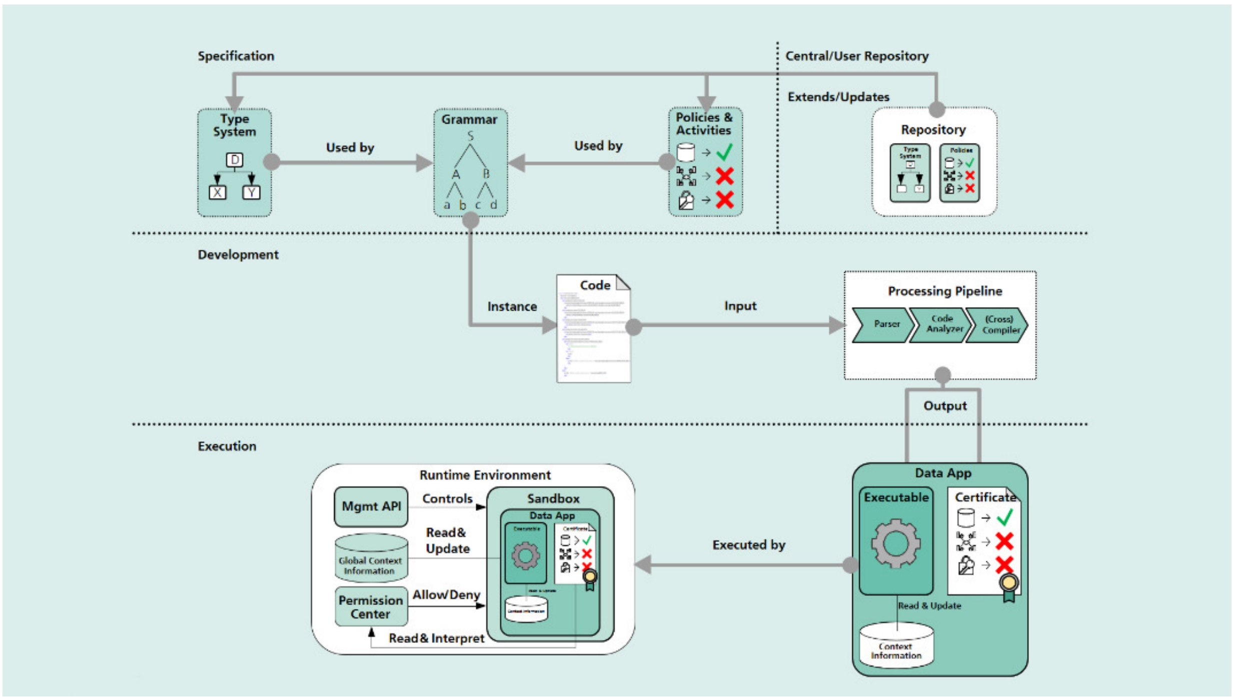


Figure 26: Big picture of D°

4.3.1 Communication Flows

Since usage control is one of the central features of D°, most relevant components are directly integrated into D° components. Therefore there aren't much communication flows required for operating a Data App. Nevertheless a Data App can, at any time, establish a connection to external systems (e.g. DAPS) if it is necessary for operation. Although a direct mapping of the XACML terminology to D° is not possible, the concepts can be mapped to each other with some drawbacks in accuracy:

4.3.1.1 Policy Administration Point

The Policy Administration Point (PAP) is the language itself. The grammar of D° allows the definition of policies as well as the definition of where the policies need to be enforced. D° distinguishes between two different entities for policies: Constraints and Policies.
A Constraint is the representation of a single, simple rule which needs to be enforced (e.g. "Use data only before 00:00 01.01.2020"). Each constraint needs a piece of Java (or any compatible JVM language) code which is responsible of the enforcing. The following code block shows the textual D° definition of a constraint that ensures that a given content does not exceed a defined length.

This text will not give any detailed information about the syntax of D° since this is out of the scope of this document.

```
MaxLength := $Constraint(
    @attribute["maxLength", $Integer],
    @attribute["content", $Text]
)
```

Code 2: Constraint in D°

The corresponding implementation for this constraint can be found in the next code block. It is written by using the JVM language Kotlin.

```
package de.fhg.isst.degree.policies.core.date


import de.fhg.isst.degree.policies.annotations.PolicyAnnotation
import de.fhg.isst.degree.policies.api.EmbeddedPolicyApi
import de.fhg.isst.degree.policies.execution.PolicyInputScope
import de.fhg.isst.degree.types.CompositeInstance
import de.fhg.isst.degree.types.PrimitiveInstance
import java.util.*


@PolicyAnnotation("MaxLength")
class MaxLengthConstraint : EmbeddedPolicyApi {


    override fun acceptPrecondition(input : PolicyInputScope):
Boolean {
        val maxLengthInstance = input.get("maxLength")!!
        val contentInstance = input.get("content")


        val maxLengthValue = (maxLengthInstance as
PrimitiveInstance).read().toIntOrNull()
        val content = (contentInstance as PrimitiveInstance).read()


        return content.length <= (maxLengthValue ?: 0)
    }


    override fun acceptSecurityManagerIntervention(input:
PolicyInputScope): Boolean {
        return true
    }


    override fun acceptPostcondition(input: PolicyInputScope):
Boolean {
        return true
    }


}
```

Code 3: D° constraint in Java

The second type of policy entities is the policy itself. It does not provide any code/logic for its enforcement. Instead the policy is a container for an arbitrary amount of other policy entities. That way it is possible to construct complex policies by using (simple) constraints and policies. If a Data App is using a policy, during the enforcement all contained elements (policies and constraints) will be evaluated and only if none of the elements enforcement fails the execution is continued.

The following code block shows a simple policy which restricts the usage to a given time interval. To achieve this goal two constraints are placed in the policy: One for the start of the usage interval and one for the end.

```
AllowedTimeInterval := $Policy(
    @dependency["useNotBefore", $UseNotBeforeTimeStamp],
    @dependency["useNotAfter", $UseNotAfterTimeStamp]
)
```

Code 4: Constraint for Time Interval

The mapping of definitions to implementations for constraints is done by using the `@PolicyAnnotation` within the implementation. Each policy entity has an execution container which encapsulates the implementation. If there is no implementation available or needed (e.g. for policies) a special NOOP (no operation) execution container is used.

While policies only contain other policy entities, constraints contain actual attributes. For the enforcement, it is necessary to bind these attributes to actual values. If we take a look at the MaxLength example again, there are two attributes. The content attribute can only be resolved during runtime since it is probably that the length of some input variable should be restricted. But the maximal length should already be known during the development. To prevent later misuse (intentional or inadvertently) the known values should be set as early as possible.

To support this and also greatly increase the reusability of constraints with different attribute values, D° uses policy entity instances within Data Apps instead of policy entities. So if within a Data App two values should not exceed a maximum length, two instances of the MaxLength constraint are used, each with its own set of attribute values.

The concept of instances is used for policies, too. It allows to create a policy instance which contains constraint instances and other policy instances.

If instances for constraints and policies are created they must match to their definitions. The following UML class diagram (Figure 27) shows all classes and their relations which have been described in this section.

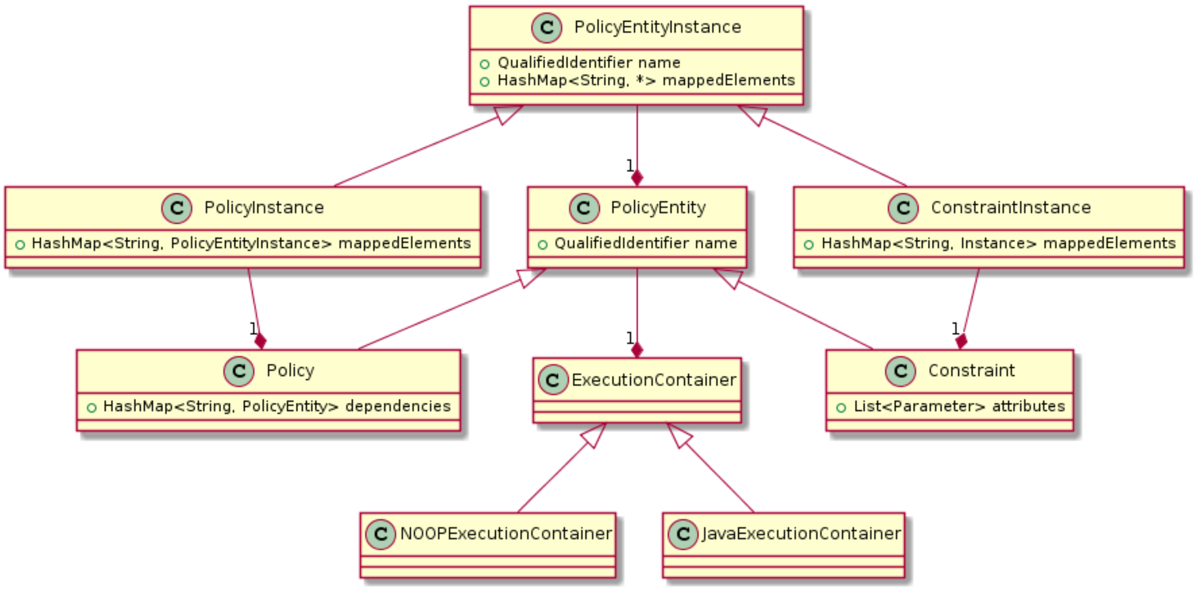


Figure 27: UML diagram describing all the PAP classes of D°

4.3.1.2 Policy Information Point

D° allows to retrieve additional required information (e.g. contextual or historical) from different locations, depending on the nature of the required information.

If some information which is local to the executing Data App (e.g. "How many times this Data App was executed") is required, the execution context of the Data App itself is queried. The execution context is accessible within the whole Data App and all parts of it (policies, activities, etc.) can retrieve information from it. It contains a hierarchy of context modules which contain various types of data.

The following UML class diagram (Figure 28) shows the main components which are part of a single Data App. All shown components (and additional ones which are not included into this diagram for clarity reasons) can access the execution context at any time. The execution storage is persistent between multiple executions of a single Data App and every state changing (add entity, write) activity is written to a persistent log.

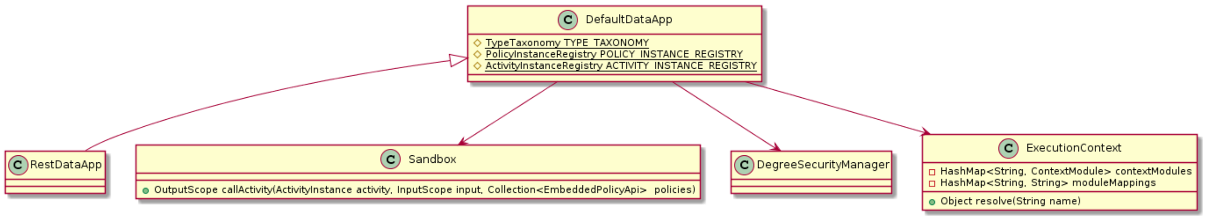

DefaultDataApp
TypeTaxonomy TYPE_TAXONOMY
PolicyInstanceRegistry POLICY_INSTANCE_REGISTRY
ActivityInstanceRegistry ACTIVITY_INSTANCE_REGISTRY
RestDataApp
Sandbox
OutputScope callActivity(ActivityInstance activity, InputScope input, Collection<EmbeddedPolicyApi> policies)
DegreeSecurityManager
ExecutionContext
HashMap<String, ContextModule> contextModules
HashMap<String, String> moduleMappings
Object resolve(String name)

Figure 28: UML diagram describing all the PIP classes of D°

To get a better overview about the capabilities of the execution context, the following class diagram (Figure 29) shows the different types of context entities which are used to store the data, as well as the context module which is a hierarchically element.

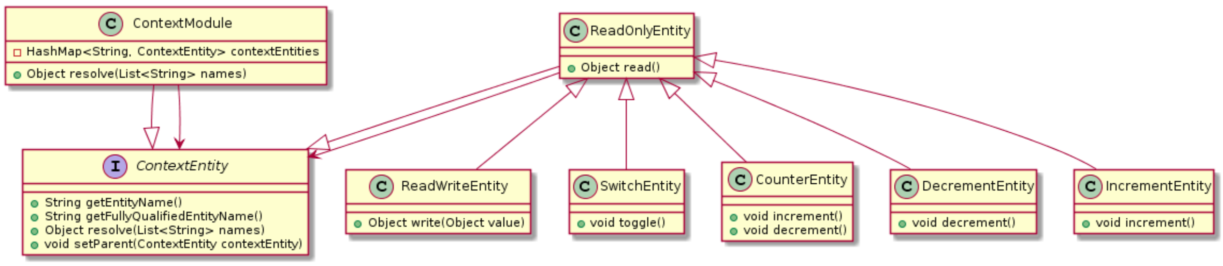


Figure 29: UML diagram of context entities of D°

In addition to additional information which is of local nature, there is global information which may need to be used during execution or policy decision making (e.g. "Which Data Apps accessed my Data?"). For these scenarios a global execution context similar to the one of Data Apps is required. Therefore the D° runtime environment provides a global execution context which can be accessed by all Data Apps it executes.

4.3.1.3 Policy Enforcement Point

The Policy Enforcement Point (PEP) is responsible of intercepting actions and making decision request to the Policy Decision Point (PDP). This functionality is provided by two elements within Data Apps:

On the one hand side the Data App contains a sandbox. Each called activity within a Data App is delegated to the sandbox. In the context of D° an activity is a (complex) functionality which is executed atomically within Data Apps. For every policy which must be enforced for this activity call, the sandbox checks if the precondition is fulfilled before executing the activity, and if its post condition is fulfilled after execution.

The second element is the DegreeSecurityManager. It allows to intercept the execution prior specific actions (e.g. writing a file, network communication). During these interceptions the relevant policies can perform checks and if one of them fails the desired action will not be executed.

4.3.1.4 Policy Decision Point

The actual decision making if a policy is met or not is performed by the PDP. D° does not provide a single PDP. Instead the implementation for each constraint is a PDP for exactly this constraint. Each constraint can perform checks prior execution, after execution, and if the security manager intercepts the execution.

4.3.2 Integration Concept

Since D° is a programming language instead of a piece of software which is operated separately/in addition to applications, no cumbersome integration is required. As long as a system is capable of executing the D° runtime environment as well as Docker, all requirements are met to execute Data Apps which have been developed with D°.

The D° runtime environment is a Java application which orchestrates, isolates, and controls all running Data Apps. Since it is a Java application just like the final output of the D° compiler, D° can be operated on many different devices.

4.3.3 Transformation of ODRL policies

While a transformation of arbitrary ODRL policies into the format D° uses is neither reasonable (- since D° has a limited set of enforceable policies just like every other solution -) nor technically feasible, it is necessary to process the ODRL policies which are used within the IDS.

Since the amount of ODRL policies which are supported by the IDS and in addition are available as templates within the MYDATA policy editor, the problem is better manageable.

For this reason the policy editor which is used for MYDATA has been forked and extended to support the generation of D° policies. That way the user only has to fill simple forms to generate policies which can be integrated into Data Apps.

The output of this policy editor can be integrated into data apps by binding it to (e.g.) activities, which need to fulfill the generated policy.

4.4 Usage Control Technologies in Comparison

There are various technologies available in the IDS which can be used to implement Usage Control (see section 3.5). In the following sub-sections, these technologies are compared based on general characteristics, their capabilities and policy languages as well as their location of implementation as described in the previous sections (see section 3.3.2). All these tables also give a compact overview about all the technologies and their capabilities.

The comparison in Table 6 considers general characteristics such as purpose, documentation, license, programming language, management capabilities, graphical user interface and the technology readiness level (TRL).

	MYDATA	**LUCON**	**Degree (D°)**
Purpose	Usage Control Enforcement	Control of data flows, enforcement of obligations dependent on data flows	Development of data processing applications with integrated usage control.
Documentation	https://www.mydata-control.de/ https://developer.mydata-control.de/	https://industrial-data-space.github.io/trusted-connector-documentation/docs/usage_control/	T3 – Deliverables (available on request)
License	Open Source SDK: Apache 2 Decision and Management Service: Proprietary License	Apache 2	No license
Programming Language	Java	Java	D° Java as Host
Management	Cloud service, on premise hosting, Java library	Eclipse plugin for policy authoring Administration via the Trusted Connector web interface	IDE & D° runtime environment

Graphical User Interface	Web UI	XText Editor for policy definition. Policy display over Trusted Connector GUI.	No
TRL	7-8	5	4

Table 6: General Comparison

Table 7 considers the capabilities of the presented usage control technologies.

	MYDATA	LUCON	Degree (D°)
Support of IDS Meta Data	Yes	Yes	Yes
Policies address Message Content	Yes, for structured data	No	Yes
Support of additional Information Sources	Yes	Yes	Yes
Policies support Permissions	Yes	Yes	Yes
Policies support Obligations	Yes	Yes	Yes
ODRL Policies supported	Yes, Policies are transformed	No, but in preparation	No
Modification of Data in Transit	Yes	No	No
Extension Possibilities	Yes Own modifiers (data modification in transit), Own PIPs (information connection), Own PXP (actions)	Yes, open source framework, customizable	Yes Own datatypes Own policies/constraints Own activities

Table 7: Policy Language Comparison

Table 8 presents all policy languages for the aforementioned usage control technologies. In addition, ODRL is added as technology-independent specification level policy language.

	MYDATA	LUCON	Degree (D°)	ODRL
Format	XML	LUCON DSL	D° DSL	JSON-LD, XML, RDF Triple
Features	Bool Temporal Cardinal Time-Based External Sources	Bool ~~Temporal~~ Cardinal ~~Time-Based~~ External Sources	Bool Temporal Cardinal Time-Based External Sources	Bool Temporal Cardinal Time-Based External Sources

	Context-Awareness Execution Block Modify Monitor	Context-Awareness ~~Execution~~ Block ~~Modify~~ Monitor	Context-Awareness Execution Block ~~Modify~~ ~~Monitor~~	Context-Awareness Execution Block Modify Monitor
Default Policy Decision	Black- or Whitelisting (configurable)	Black- or Whitelisting	Blacklisting	Black- or Whitelisting
Standard	No	No	No	W3C
Version	Stable: 3.2.52 (30.01.2018) RC: 4.0.74 (10.01.2019)	Version 2.0 (31.07.2019)	Version 2.0 (31.03.2019)	Version 2.2

Table 8: Policy Language Comparison

Finally, Table 9 compares the IDS Usage Control technologies with respect to their enforcement location (explained in section 3.3.2). The table lists all possible enforcement locations for the respective technology. Which location is necessary depends on the respective Usage Control requirements that need to be fulfilled.

	MYDATA	**LUCON**	**Degree (D°)**
IDS Connector Message Bus **(intercept between Apps and within a route)**	Yes	Yes	No
Usage Control enabled IDS Apps **(enforce policies within Apps)**	Yes	No, in development	Yes
Infrastructure **(e.g., Database, external Systems)**	Yes	No, in development	No
Client System and Services **(e.g., Operating System, Services on (external) servers)**	Yes	No	No

Table 9: Enforcement Locations

4.5 Discussion

The chapter presented three IDS Usage Control technologies for enforcing Data Usage Control. The three technologies differ in their capabilities, license and technology readiness level. The decision on what kind of technology to choose depends on the usage scenario. For example, integrating Usage Control capabilities into an application is supported by D° and MYDATA. Both technologies are integrated into the application at development time. D° and MYDATA needs to be integrated as Java library. However, D° is integrated using annotations and MYDATA by using function calls.

LUCON and MYDATA are integrated as part of the routing or as interceptor in the message routing. Hence, both integration concepts are very similar to each other. However, LUCON supports the use of labels, which must be specified as parameters in MYDATA. Moreover, MYDATA supports the modification of data in transit, which is not supported by LUCON. In contrast to LUCON, MYDATA is not open source, but has the highest TRL.

MYDATA offers different extension possibilities with their Open Source SDK to develop own PEPs, PIPs and PXPs that are supported by the MYDATA policy language.

5. Data Provenance, Transparency and Accountability

Data provenance tracking is closely related, but also complementary to distributed data usage control. It has its origins in the domain of scientific computing, where it was introduced to trace the lineage of data. Data provenance tracking thereby allows finding out when, how and by whom data was modified, and which other data influenced the process of creating new data items.

This kind of traceability is similar to the data protection requirements a data controller is confronted with, so as to be able to fulfill the data subjects' right to access. It is also closely related to the question of proving compliance with contracts, agreements, or legal regulations. And data provenance tracking can be used to facilitate clearing in decentralized data ecosystems, since it is capable of providing information concerning data transactions and data usage.

5.1 Relationship between Data Provenance and Usage Control

However, while distributed data usage control is concerned with the enforcement of rights and duties when exchanging data across system boundaries, the focus of data provenance tracking is on transparency and accountability. In other words: While a Policy Enforcement Point (PEP) serving for distributed data usage control in most cases needs to be able to proactively intercept data usage actions within the control flow (i.e. preventive enforcement), a PEP for data provenance tracking only needs to passively observe, interpret and log data transactions and data usage for retrospective examination. In terms of usage control, this kind of enforcement is denoted as "detective enforcement". Despite this fact, a data provenance tracking infrastructure can be built upon the same PEPs as distributed data usage control. Furthermore, data provenance tracking does not require a policy specification language, but rather a specification of how observed actions are to be interpreted in terms of data flow or data usage (i.e., a so called data flow semantics specification). By this means, data provenance tracking maintains a data flow model that keeps track of the particular representations of data items. This kind of information can also be leveraged for data usage control enforcement. For this a data provenance storage serves as a Policy Information Point (PIP), i.e., it is accessible via a PIP interface.

Please note that the requirements of data provenance tracking in terms of establishing trust into remote infrastructures is equivalent to the requirements of distributed usage control. If data provenance shall serve as context information for usage control enforcement, i.e., decisions by a PDP, it must have been collected by a trustworthy infrastructure. The same applies if data provenance shall be used to prove compliance with contracts or agreements.

5.2 Operating Principle and Architecture

The operating principle of data provenance tracking is very similar to the operating principle of distributed data usage control. The architecture of data provenance tracking is given in Figure 30. Data provenance tracking relies on passive monitoring technology (e.g., PEPs), which deliver events indicating data usage or data flows to be logged. For this, a PEP needs to convey a semantic description of the data usages or data flows its observed events indicate. The data provenance tracking infrastructure provides a data flow tracking component, which understands such semantics specifications. The PEP also needs to forward events together with metadata (including a unique identifier of the data's content), so that logged transactions can be attributed to data content when data provenance queried.

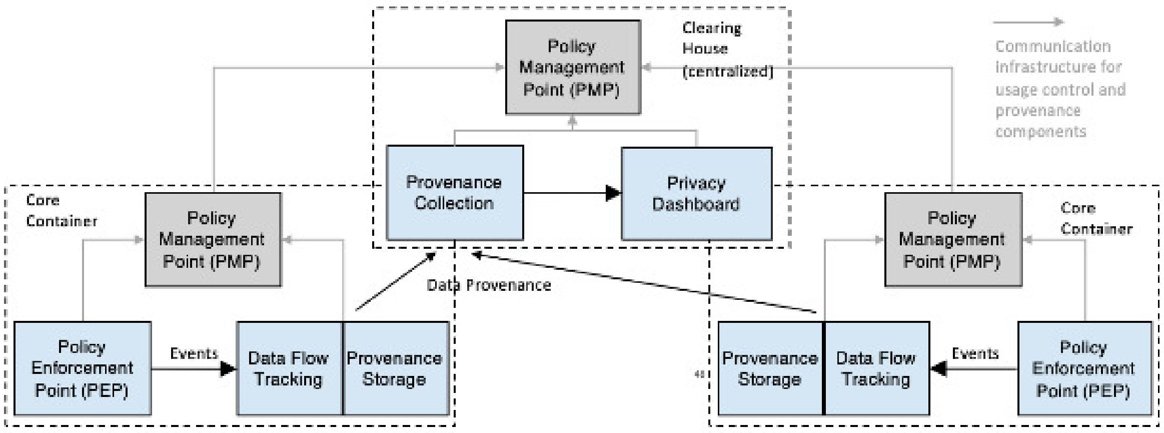

Policy Management Point (PMP)
Clearing House (centralized)
Communication infrastructure for usage control and provenance components
Provenance Collection
Privacy Dashboard
Core Container
Policy Management Point (PMP)
Policy Enforcement Point (PEP)
Events
Data Flow Tracking
Provenance Storage
Data Provenance
Provenance Storage
Data Flow Tracking
Events
Policy Enforcement Point (PEP)
Policy Management Point (PMP)
Core Container

Figure 30: Distributed Data Provenance Tracking Architecture

The PEP resides within the message routing component of the Connector (or Data App). It is registered at the data flow tracking component via a registry component (e.g., a local Policy Management Point, PMP). The same applies for the data flow tracking component. Thereby a PEP can query the registry for the communication interface of the local data flow tracking component, which is then used to deploy semantics specifications for its observed events and to forward actual events during operation.

Data provenance information is queried at a Provenance Dashboard, which is accessible via a Clearing House. The Provenance Dashboard returns a provenance graph for the unique identifier of a data asset. In case data provenance shall be used as contextual information for usage control, a provenance architecture with a single centralized data provenance storage component per usage control domain has the advantage that for queries for data provenance concerning a specific data item provenance must not be aggregated from several data provenance storage components so that events intercepted for usage control are not blocked for an arbitrarily long time.

Please note that provenance data storages of particular domains may also reside within a usage control infrastructure provided as a cloud service, such as in the case of MYDATA.

5.3 Transformation of ODRL policies

Data provenance tracking can either be instantiated to track any observable transaction of any data item exchanged over the IDS, or to be only collected for specific data items where it is explicitly required by the data provider. In the latter case this can be done using a provenance tracking policy in ODRL in the information model, which is translated into a MYDATA policy at the data consumer, since so far data provenance tracking builds upon MYDATA.

6. Discussion and Future Work

This section discusses what usage control in its current state can achieve in the IDS and future work.

6.1 Discussion

The subsection discusses the current state of Usage Control in the IDS. Therefore, it summarizes the capabilities and limitations already mentioned in the previous sections and its implications.

6.1.1 Capabilities

By using the current state of usage control that is implemented into the IDS it can support developers and administrators in setting up correct data pipes that comply with policies and do not leak data via side effects. For example, usage control prevents IDS connectors from treating data in an undesired way such as forwarding personal data to public endpoints. The capabilities and reliability depends on the concrete Connector implementation and its trust level. In addition, usage control in the IDS can also be used as an audit mechanism, which creates evidence of a compliant data usage. For instance, usage control mechanisms can monitor and log usages of data.

With usage control, it is possible to modify the messages exchanged between endpoints to comply with a policy. For example, personal data can be removed or data can be aggregated. It is furthermore possible to change the route of the package or drop it completely if demanded by a policy. Moreover, apps running on the connector can implement PEPs, which connect the usage control infrastructure and further enhance the functionality by allowing a more detailed control and data flow tracking. In addition, apps may use D° to implement data usage control capabilities.

Data usage control at the data consumer side is a special topic, which is also addressed in the document. There are possibilities to interact with third party software. For example, the PXP concept offers standardized interfaces that can be used in the policies and offers a flexible way to implement additional features. In addition, there is the possibility to encrypt the data at the provider side and decrypt the data at the consumer side (e.g., modify data in transit). In doing so, only the IDS connector is capable to perform the decryption operation. Moreover, the data processing within the IDS connector must adhere to the policies deployed.

6.1.2 Limitations

Usage control does only work within its ecosystem where it has the full control over the data. Achieving full control does also mean that there are cases that expect developers to integrate usage control components (such as PEP, PIP, PXP) into their application or services to fulfill usage restrictions (e.g., to interact with third party components). In most cases, developers have to integrate at least the PEP component to control data flows.

Although usage control uses several abstraction layers, there will always be a possibility to circumvent the system. One of the best-known examples for that is media disruption. For example, a usage control system may control taking screenshots and printing, but it cannot prevent a person to take a photo from the screen displaying the sensitive data. That said, if data leaves the ecosystem, it needs additional protection (such as encryption) in order to keep control over the data.

Usage control is no hard security feature such as cryptography, which one can mathematically prove. It is rather a complementary solution to have more fine-grained control over data flowing in a system and goes well together with organizational rules (see Section 2). In addition, it is rather an extension to access control than replacing it.

Implementing a usage control technology does not automatically establish trust in an endpoint. It necessarily builds upon an existing trust relationships such as existing contracts and a secure computing environment like highly trusted platforms (such as the IDS Trusted Connector).

When physical access is granted to administrators, protection against data theft by persons with malicious intents is almost impossible to prevent. As the administrator of the data consumer will act on behalf of the data consumer organization's management, he is a reasonable attacker for usage control enforcement. It is part of future work to evaluate possible countermeasures.

6.1.3 Implications

Implementing usage control into an existing system has various implications. Creating events, the decision-making and the transfer of events between the affected components takes extra time as well as some computational power. Besides, all usage control components need memory to persist information or to perform the computation. In sum, it will reduce the performance of the overall system and demands machines with more power.

As already stated, the basic idea of usage control is to control the dataflow. In a case where a developer enhances an application with usage control technology, he needs to integrate at least one PEP. Depending on the complexity of the enforcement, he needs to integrate even more than one PEP within one or several applications. As all of those integrations also need planning and testing, it increases the development and testing time and effort in comparison to a system without usage control.

In addition to the enforcement components, the system needs policies. Therefore, a policy specification process needs to be established. During this process, the policy experts of the data owner have to collect information about how others should use the data. This process costs additional time and communication effort for the data owners and leads in the end to higher costs.

6.2 Future Work

The subsection addresses future work for usage control in the IDS. We focus in policy negotiation, provenance tracking and the usage control object, which are currently addressed in IDS-related research projects. Other topics such as policy lifecycle and evolution is neglected

6.2.1 Policy Negotiation Parameters

In a Policy Negotiation process between two IDS parties, the Data Owner (Data Provider) party is an Offer policy creator that proposes a policy that meets at least its mandatory demands and benefits and the Data Consumer party is a Request policy creator that negotiates with the Data Provider in order to achieve an agreement. The outcome of the Policy Negotiation process is an Agreement policy that both IDS parties confirm to enforce it during the data exchange period.

The IDS parties bargain over a set of policy parameters in the Policy Negotiation process. As a future work, we will investigate these policy parameters and in addition, the feasible approaches of acquiring an Agreement policy. Moreover, we will implement some of the (semi-)automated Policy negotiation approaches.

6.2.2 Provenance Tracking

The DataFlowTracking and ProvenanceStorage-Component should be integrated in a Trusted Connector with usage control framework MYDATA. For this purpose, a systematic way should be given of how an information flow semantic can be specified for a Policy Enforcement Point (PEP).

Furthermore, a demonstrator for the Provenance-Collection and Provenance-Dashboard is planned.

6.2.3 IDS Usage Control Object

A Connector has to identify the data, which is transferred. Therefore, an object that contains metadata, including an identifier and the data as payload, is needed. We have to define what exactly constituent element of this Usage Control Object is.

As an example we can assume, that usage control object contains an unique identifier such as a UUID (META) and the data as payload (DATA). If the camel interceptor intercepts the route, the usage control framework checks, if there are policies for this unique identifier. If policies exist, they have to be applied on the usage control object. We can differentiate between allowing or denying the entire object. If we allow the object, there is the possibility that the policies demand the modification of the payload.

A. Glossary

D°	Degree
DAPS	Dynamic Attribute Provision Service
DSL	Domain Specific Language
IDS	International Data Spaces (previously Industrial Data Spaces)
LUCON	Logic based Usage CONtrol
MDSD	Model Driven Software Development
MYDATA	MYDATA Control Technologies
ODRL	Open Digital Rights Language
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PII	Personally Identifiable Information
PIP	Policy Information Point
PIR	Participant Information Registry
PMP	Policy Management Point
PXP	Policy Execution Point
RC	Release Candidate
TRL	Technology Readiness Level
UML	Unified Modeling Language
XACML	eXtensible Access Control Markup Language
UUID	Universally Unique Identifier

B. Further Information and Contact Persons

A webinar on data usage control can be found here:

- Webinar: https://www.youtube.com/watch?v=6KHqxMKOmHo

We present references to further information and the responsible Fraunhofer Institute as well as the responsible contact person next.

B.1 MYDATA Control Technologies

Further information about the MYDATA Control Technologies can be found here:

- Website: https://www.mydata-control.de/
- Developer Website: https://developer.mydata-control.de/
- Source Code: https://git.iese.fraunhofer.de/ind2uce
- Binaries: https://search.maven.org/search?q=g:de.fraunhofer.iese.ind2uce

The responsible institute is Fraunhofer IESE in Kaiserslautern, contact persons: Dr.-Ing. Christian Jung and Andreas Eitel.

B.2 Logic-based Usage Control

Further information about the Logic-based Usage Control can be found here:

- https://industrial-data-space.github.io/trusted-connector-documentation/docs/usage_control

The responsible institute is Fraunhofer AISEC in Garching, contact person: Gerd Brost

B.3 Degree (D°)

Further information about Degree can be found here:

- https://www.isst.fraunhofer.de

The responsible institute is Fraunhofer ISST in Dortmund, contact person: Fabian Bruckner.

B.4 Data Provenance

Further information about Data Provenance can be found here:

- https://www.iosb.fraunhofer.de/servlet/is/87303/
- https://www.iosb.fraunhofer.de/servlet/is/86726/

The responsible institute is Fraunhofer IOSB in Karlsruhe, contact person: Dr.-Ing. Pascal Birnstill.

REFERENCES

[1] B. Otto, S. Lohmann, S. Auer, G. Brost, J. Cirullies, A. Eitel, T. Ernst, C. Haas, M. Huber, C. Jung, J. Jürjens, C. Lange, C. Mader, N. Menz, R. Nagel, H. Pettenpohl, J. Pullmann, C. Quix, J. Schon, D. Schulz, J. Schütte, M. Spiekermann und S. Wenzel, „Reference Architecture Model for the Industrial Data Space," Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V. AND Industrial Data Space e.V., München, 2017.

[2] B. Parducci, H. Lockhart und E. Rissanen, „eXtensible Access Control Markup Language (XACML) Version 3.0," OASIS Standard, 22 January 2013. [Online]. Available: https://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html. [Zugriff am 16 08 2019].

[3] P. I. 4.0, „Zugriffssteuerung für Industrie 4.0 - Komponenten zur Anwendung von Herstellern, Betreibern und Integratoren," Bundesministerium für Wirtschaft und Energie (BMWi), Berlin, 2018.

[4] A. Eitel, C. Jung, C. Haas, C. Mader, G. Brost, J. Schütte, J. Pullmann, J. Zrenner und P. Birnstill, „Usage Control in the Industrial Data Space," Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung, IESE-Report No. 056.17/E, Kaiserslautern, 2017.

[5] Wikipedia, „Data Universal Numbering System," [Online]. Available: https://en.wikipedia.org/wiki/Data_Universal_Numbering_System. [Zugriff am 16 08 2019].

[6] M. C. Mont und S. Pearson, „Sticky Policies: An Approach for Managing Privacy Across Multiple Parties," Computer, pp. 60-68, 2011.

[7] N. Mercer, „Introducing Windows Information Protection," Microsoft, 28 02 2018. [Online]. Available: https://techcommunity.microsoft.com/t5/Windows-Blog-Archive/Introducing-Windows-Information-Protection/ba-p/166564. [Zugriff am 16 08 2019].

[8] C. Jung, A. Hosseinzadeh da Silva, A. Eitel und R. Brandstädter, „Documentation of Usage Restrictions, Language Constructs and their Technical Refinement," Fraunhofer Gesellschaft zur Förderung der angewandten Forschung, IESE-Report No. 049.18/E, Kaiserslautern, 2018.

[9] W3C ODRL Community Group, „ODRL Overview," [Online]. Available: https://www.w3.org/community/odrl/. [Zugriff am 16 08 2019].

[10] Wikipedia, „Resource Description Framework," [Online]. Available: https://en.wikipedia.org/wiki/Resource_Description_Framework. [Zugriff am 16 08 2019].

[11] RDF Working Group, „Resource Description Framework (RDF)," [Online]. Available: https://www.w3.org/RDF/. [Zugriff am 16 08 2019].

[12] Wikipedia, „ODRL," [Online]. Available: https://en.wikipedia.org/wiki/ODRL. [Zugriff am 16 08 2019].

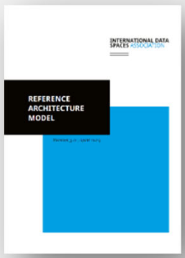
[13] R. Iannella, M. Steidl, S. Myles und V. Rodríguez-Doncel, „ODRL Vocabulary & Expression 2.2," 15 02 2018. [Online]. Available: https://www.w3.org/TR/odrl-vocab/. [Zugriff am 16 08 2019].

OUR MEMBERS

A! Aalto University	ADVANEO	agmadata DIGITALE PROZESSE FÜR GUTE LEBENSMITTEL	Allianz	Atos	Audi
bill-X Collect. Manage. Execute.	Boehringer Ingelheim	Brainport Industries	bdr. BUNDESDRUCKEREI	CAICT 中国信息通信研究院	CANADA'S DIGITAL TECHNOLOGY SUPERCLUSTER
CDQ SHARING DATA EXCELLENCE	cea	Cefriel POLITECNICO DI MILANO	iti Information Technologies Institute	CHALMERS UNIVERSITY OF TECHNOLOGY	COSMOPlat
Cybus	CTU CZECH TECHNICAL UNIVERSITY IN PRAGUE	DAIMLER	DATA AHEAD	DATATRONiQ	DB SCHENKER
Deloitte.	Deutsche Bank	DGZfP	DHBW Duale Hochschule Baden-Württemberg Ravensburg	Digital Green	DIMECC
DR. SCHNEIDER UNTERNEHMENSGRUPPE	eccenca command your data!	EcoDataCenter	ELDORADO	engie	ENGINEERING
8760 Fastems	fir an der RWTH Aachen	FIWARE	Leibniz Universität Hannover	Fraunhofer	GateHouse Logistics
GERMAN EDGE CLOUD	GESIS GESELLSCHAFT FÜR INFORMATIONSSYSTEME	Google	HITACHI Inspire the Next	HUAWEI	i2cat
automotive engineering iav	IBM	ILVO Instituut voor Landbouw-, Visserij- en Voedingsonderzoek	imec	Imperial College London	Institut Mines-Télécom
INDUSTRY 2025 INDUSTRIE INDUSTRIA	INNOPAY	innovalia ASSOCIATION	Insight	Irish Manufacturing Research	ITI INVESTIGATE TO INNOVATE
Klarrio	K	KOMSA DIE BESSERE VERBINDUNG	LOSEC LEADERS IN SECURITY	Lobster	Logata Digital Solutions
LOGENIOS	minnosphere	.msg	nexedi	nicos AG	olmogo data ownership solutions

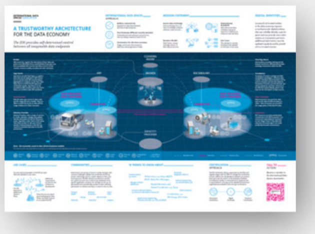
orbiter	ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ	pi-lar GmbH	POLITECNICO MILANO 1863	pwc	Quin Scape
realworldsystems	RI. SE	REWE GROUP	RITTAL	SALZGITTERAG Stahl und Technologie	SAP
SCHAEFFLER	SETLOG The Smart Connective	SICK Sensor Intelligence.	SIEMENS Ingenuity for life	Signal Solutions	SINTEF
SMEV AG Smart Mobility Evolution	TARC Tunku Abdul Rahman University College	tecnalia Inspiring Business	IK4 TEKNIKER Research Alliance	T · ·	mtc Manufacturing Technology Centre
thyssenkrupp	TNO innovation for life	TÜV NORD GROUP	TÜV SÜD	UNIKLINIK RWTH AACHEN	UNiSCON Sealed Cloud Technologies
UNITY CONSULTING & INNOVATION	universität Witten/Herdecke	University of Amsterdam	UNIVERSITY OF TWENTE.	vastuu group	VDMA
VOLKSWAGEN AKTIENGESELLSCHAFT	VTT	welotec a byte smarter	WeTech IDS-China	ZVEI: Die Elektroindustrie	

OVERVIEW PUBLICATIONS


Reference Architecture Model

Executive Summary

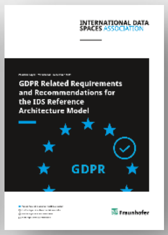
Image Brochure


Infographic

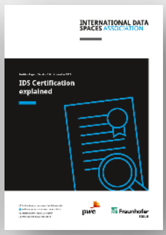
Use Case Brochures

Study on Data Exchange

Position Paper Implementing the European Data Strategy


Position Paper GDPR Require-ments and Re-commendations

Position Paper Usage Control in the IDS


Position Paper IDS Certification Explained

White Paper Certification

Sharing data while keeping data ownership

Magazine Data Spaces_Now!

For these and further downloads: www.internationaldataspaces.org/info-package

Code available at: https://github.com/industrial-data-space

CONTACT

Head Office

INTERNATIONAL DATA SPACES ASSOCIATION

Emil-Figge-Str. 80
44227 Dortmund | Germany

phone: +49 231 70096 501
mail: info@internationaldataspaces.org

WWW.INTERNATIONALDATASPACES.ORG

@ids_association

international-data-spaces-association