ODRE Policy Directory Service: A Trust-Based ODRL Service for Decentralised and Policy-Aware Ecosystems

Lucía Martín-Núñez[1,*,†], Andrea Cimmino[1,†] and Raúl García-Castro[1,†]

[1]Universidad Politécnica de Madrid, Spain

Abstract

Policies are a critical component in decentralised data ecosystems, where ensuring secure and compliant data usage is an ever-growing challenge. The W3C standard Open Digital Rights Language (ODRL) has been widely adopted for expressing access and usage control policies by several initiatives, such as Data Spaces and Solid Pods. However, ODRL has only promoted an ontology to express policies without any standardised recommendation for policy management, discovery, or enforcement. Although several proposals outside the standard have been presented to address these limitations, a complete solution to be deployed by decentralised initiatives like Data Spaces or Solid remains unexplored. This article introduces the ODRE Policy Directory Service (ODRE-PDS), a Web service that provides the features needed to rely on ODRL in practical scenarios and use cases. The directory includes policy management features, discovery, policy enforcement mechanisms, and an architecture that facilitates its integration with external trust-based systems. The directory has been used in two use cases: a time-based policy scenario and an AI-driven facial recognition access control system. In addition, several experiments on enforcement performance, scalability, and computational overhead advocate its usability by decentralised ecosystems.

Keywords

Data usage control, ODRL Policies, ODRL Directory.

1. Introduction

Decentralised environments such as European Data Spaces, Solid Pods, Internet of Things (IoT), or Knowledge Graphs increasingly require robust and adaptable mechanisms to manage access and usage control of digital resources [1, 2, 3, 4, 5]. As data flow across organisational and national boundaries, ensuring compliance with legal constraints becomes essential, particularly in light of regulations such as the General Data Protection Regulation (GDPR) [2] and the Digital EU Artificial Intelligence Act [6]. The W3C standardisation group Open Digital Rights Language (ODRL) has published as a recommendation a semantic model [7], i.e., an ontology, to define access and usage control policies. However, ODRL focuses on policy specification and lacks recommendations for other related policy tasks such as discovery or enforcement mechanisms. As a result, there are no standard guidelines on how to operationalise or implement these features within policy-aware service architectures.

Outside the standard, several proposals have been presented to rely on ODRL in real-world scenarios, tackling problems such as policy specification, policy management, policy enforcement, or supporting different scenarios; from known access control to usage control. However, up to the authors' knowledge, despite the numerous proposals, no existing proposal offers a comprehensive solution that combines all these features, providing a general Web-service-orientated architecture suitable for decentralised infrastructures.

In this article, the ODRE Policy Directory Service (ODRE-PDS) is introduced to address these limitations. ODRE-PDS is a Web service designed with a modular architecture that supports the management,

ODRL and Beyond: Practical Applications and Challenges for Policy-base Access and Usage Control. OPAL 2025 Co-located with the Extended Semantic Web Conference, Portorož, Slovenia · June 1 or 2, 2025.

*Corresponding author.

†These authors contributed equally.

lucia.martin.nunez@alumnos.upm.es (L. Martín-Núñez); andreajesus.cimmino@upm.es (A. Cimmino); r.garcia@upm.es (R. García-Castro)

0009-0001-4043-4167 (L. Martín-Núñez); 0000-0002-1823-4484 (A. Cimmino); 0000-0002-0421-452X (R. García-Castro)

discovery, and enforcement of ODRL policies via a RESTful interface. The ODRE-PDS follows a design that aligns with W3C standards, for instance, supporting different RDF serialisations and the SPARQL protocol and queries. The ODRE-PDS follows an approach applied to other policy-related initiatives where there is a service, or several components, providing the aforementioned features; like XACML [8].

The contributions of the ODRE-PDS address three limitations in current ODRL-based initiatives: (i) it provides policy management capabilities and provides discovery features over stored policies; (ii) it enables policy enforcement based on contextual information, including data stored in or retrieved through ODRE-PDS; and (iii) it provides an out-of-the-box implementation ready to use in real-world scenarios or infrastructures such as Solid Pods and European Data Spaces.

To showcase the ODRE-PDS integration in real-world scenarios, the article presents two use cases derived from the European project AURORAL[1] where the directory has been adopted and a Spanish National Project. The first entails time-restricted access to public documents based on policies, and the second relies on policies that grant access to documents based on biometric recognition for identity verification. These scenarios demonstrate context-aware and auditable enforcement capabilities while maintaining compatibility with the ODRL model. In addition, three experiments have been carried out to test the ODRE-PDS performance, the results of which advocate the suitability of the directory to be used in real-world use cases.

This rest of the paper is organised as follows: Section 2 surveys similar proposals in the literature, then in Section 3 an implementation agnostic architecture is presented for the ODRE-PDS, and in Section 4, a specific implementation is introduced. Section 6 presents the experiments carried out and, finally, Section 7 states the conclusions of the article.

2. State of the Art

Enforcing policies, regardless of access or usage control, is a critical challenge for data-centric environments and particularly decentralised ecosystems, such as the Internet of Things (IoT) [4, 5], Solid Pods [2], and European Data Spaces [1]. In these contexts, data governance is increasingly based on self-sovereign and federated approaches [9], where control must be preserved beyond initial access decisions. This paradigm is known as *usage control*, which extends traditional access control by ensuring that data consumers continue to respect policy constraints after access has been granted [10]. To enable reliable and interoperable data sharing, policy initiatives must support different operations such as policy specification, the enforcement of access and usage control.

This section analyses the capabilities of existing policy initiatives for management and enforcement in decentralised privacy-sensitive systems. The analysis is structured around the following evaluation dimensions: A) *Policy specification*: the type of model promoted by the different initiatives to express the policies (e.g., an ontology or an XML schema); B) *Policy management*: the initiative promotes an architecture, software or service oriented, to support policy management operations, i.e., CRUD operations (create, read, update, or delete); C) *Policy enforcement*: the initiative promotes an architecture, software or a service oriented, to evaluate policies taking into account the state of the world. The state of the world (SoTW) refers to the external contextual information used to evaluate the policy, such as time, identity, or environmental factors.; D) *Access control*: the initiative supports authorization mechanisms that enforce explicit allow/deny decisions over resource access, typically based on the identity of the requester and policy rules; E) *Monitoring*: the initiative promotes a software or service that supports usage control, that is, the ability to observe and track policy compliance over time; F) *Operations interface*: the type of interface exposed by the initiative to manage and evaluate policies. This may include RESTful APIs, SPARQL endpoints, command-line tools, or web-based dashboards. Table 1 shows the summary of the analysis performed in four well-known policy initiatives.

ODRL [7] is a W3C recommendation that provides a formal ontology to define access and usage control policies through permissions, prohibitions, and obligations. It supports policy specification using RDF and following the model of its ontology; which is aligned with Linked Data principles [3]. However,

[1]See https://auroral-project.eu/, Horizon 2020, Grant Agreement ID: 101016854.

Feature	ODRL[7]	XACML[8]	UCON[11]	LegalRuleML[12]
A) Policy specification	ontology	schema	conceptual model	schema + ontology linking
B) Policy management	✗	✓	✗	✗
C) Policy enforcement	✗	✓	✓	✗
D) Access control	✗	✓	✗	✗
E) Monitoring	✗	✓	∼	✗
F) Operations interface	✗	✓	✗	✗

Table 1
Comparison of selected policy languages.*Note:* ✓ indicates full support for the feature; ✗ indicates no support; ∼ denotes partial or conceptual support or limited interface support.

ODRL does not provide recommendations for policy enforcement, discovery, or policy management. Furthermore, the ODRL initiative does not provide any means to adopt policies in practical scenarios besides using them for descriptive purposes; the standard does not provide any service-based interface or component-based architecture [13]. Nevertheless, recent efforts, such as the ODRL profile for expressing consent in Solid environments [2] advocate its potential for granular policy specification in decentralised systems. However, these efforts remain limited in their ability to offer executable or integrated enforcement features.

XACML [8], developed by OASIS, defines both a policy language and a reference architecture for access control, based on the interaction between Policy Decision Points (PDPs), Policy Enforcement Points (PEPs) and Policy Action Point (PAP). It supports policy specification using XML and following a particular XML schema and provides real-time access decisions via service-based interfaces. In addition, XACML includes policy management operations or monitoring capabilities; however, it lacks support for ontology-based semantics or dynamic usage control. The XACML standard is suited for static access control scenarios in centralized environments.

Usage CONtrol (UCON) [11] is a conceptual model that allows to describe mutability of the attributes, representing those that may change during access, and allows to describe continuity, which enables policy evaluation before, during, and after access. It defines policies through a formal model and supports enforcement mechanisms across the access lifecycle. UCON provides conceptual support for monitoring, but lacks standardized implementations, semantic modeling, integration interfaces, and policy management functionality. It is primarily used in academic or prototype contexts requiring persistent and adaptive control [10].

LegalRuleML [12], also developed by OASIS, is an XML-based legal rule language designed to represent legal logic, rights, and obligations. It enables the specification of normative rules through a structured, formal model that supports advanced compliance reasoning. However, it does not support policy enforcement, runtime monitoring, or policy management operations. LegalRuleML also lacks support for access control (AC) mechanisms and does not provide service-level interfaces for integration in operational systems. As such, it is primarily suited for offline legal analysis, documentation, and regulatory alignment, rather than for executable policy enforcement.

As summarised in Table 1, none of the reviewed initiatives simultaneously supports semantic modelling based on ontologies, policy enforcement at runtime, and integration with operational infrastructures. In particular, the absence of monitoring capabilities and dynamic adaptability restricts their applicability in decentralized ecosystems where authorizations and contextual conditions evolve over time.

2.1. ODRL-Based Enforcement Frameworks

Since this article focuses on ODRL-based proposals and ODRL lacks recommendations in addition to the ontology, this subsection reviews recent frameworks and proposals that extend the W3C ODRL 2.2 specification with additional semantics or mechanisms not covered by the official standard. These proposals rely on the semantics and vocabulary of ODRL, but differ in how they are used to manage, discover, or enforce policies.

Because ODRL does not provide native support for policy execution [13], several initiatives have emerged to bridge the gap between policy specification and enforcement. Table 2 presents a comparative analysis of these proposals using the evaluation dimensions introduced earlier, including a new criterion G) Types of constraints: that each framework supports during policy evaluation (e.g., static vs. dynamic). This dimension highlights whether the frameworks rely solely on preconfigured values (static) or are capable of processing runtime values provided as context before the enforcement by the system or a third-party entity (dynamic).

ODRL Policy Modelling [14] explores how ODRL policies can be aligned with legal regulations, focusing on formal compliance checking. The framework provides reasoning mechanisms over policy expressions to detect inconsistencies and evaluate whether they fulfill regulatory requirements. However, it does not include any policy enforcement at runtime, monitoring, or API integration features.

DUC [15, 16] and IntentKeeper [17] are ODRL-based frameworks designed for specific application domains: industrial IoT and federated learning, respectively. Both provide RESTful APIs that allow external systems to interact with policy evaluation services at runtime, enabling practical policy enforcement. They incorporate basic enforcement mechanisms over data access and transmission, but do not include support for access control or runtime monitoring of policy compliance.

ODRL-PAP [18] is a policy administration component that enables the transformation of ODRL policies into executable Rego rules for enforcement via the Open Policy Agent (OPA). Policies are specified in ODRL and automatically compiled into enforcement-ready logic. The system provides a REST API for the creation, retrieval, and deletion of policies, supporting external integration. However, it does not include policy monitoring or support for explicit access control, as access decisions are handled by OPA using logic-based policies rather than static subject-permission mappings. Enforcement decisions are delegated to OPA, which evaluates runtime access conditions. While it lacks built-in compliance checking, the separation between specification and execution makes ODRL-PAP suitable for modular, interoperable environments.

The MOSAICrOWN Policy Engine [19] is an ODRL-based access control module developed in the context of privacy-preserving data analytics. It evaluates access requests using ODRL policies that define constraints over actions, purposes, data subjects, and contextual attributes. The engine supports complex rule hierarchies and expressive conditions such as attribute visibility and duty-based obligations. However, it does not support runtime policy management or monitoring, and it lacks integration with AC mechanisms. Its architecture is tailored to the needs of the MOSAICrOWN framework, and no general-purpose API is provided. Despite these limitations, it shows how ODRL can be effectively applied to control access in federated data pipelines.

Interoperable Usage Control [20] proposes a usage control framework based on ODRL for the context of European Data Spaces. It introduces support for dynamic constraints—such as temporal, contextual, or purpose-based conditions—making policy enforcement more adaptive. However, the framework does not provide integration APIs or monitoring capabilities, and its implementation is primarily conceptual at this stage.

The OTT Copyright Management System [21] extends ODRL for the automated governance of digital content rights in Over-the-Top (OTT) platforms. It supports policy specification using the ODRL 2.2 vocabulary to represent copyright transactions, ownership ratios, and usage permissions. The system includes automatic policy enforcement via smart contracts that verify agreement thresholds before executing copyright transfers. It incorporates mechanisms for policy management such as agreement recording and verification. While it does not integrate AC, it ensures secure control using digital signatures and zero-knowledge proofs. Monitoring is achieved through immutable blockchain logs that capture usage events and transactions. Although it does not expose a REST API, it offers a functional modular interface through Hyperledger Fabric components. It partially supports dynamic constraints related to ownership and user signatures but does not include compliance checking mechanisms.

ODRE [22] constitutes a significant contribution to ODRL-based enforcement by embedding a formal execution model directly within the policy structure. It supports evaluation of permissions, obligations, and access control, and allows compliance checking with contextual constraints. However, ODRE is code-based and lacks a service architecture or REST API, which limits its usage in a decentralized

environment. In addition, it does not provide policy management capabilities such as creation, update, or deletion of policies, nor does it support external monitoring or trust integration.

The ODRE-PDS, proposed in this paper, builds upon ODRE and extends it with formal semantics [23], third-party trust models, and dynamic contextual evaluation. It is the first framework to integrate AC enforcement, monitoring, and REST APIs within a fully extensible ODRL-based system. Similar to OWL-POLAR [24], which provides reasoning capabilities for policy enforcement, ODRL extensions aim to bridge semantic representation and runtime validation.

Framework	A)	B)	C)	D)	E)	F)	G)
ODRL Policy Modelling [25]	ODRL	✗	✗	✗	✗	✗	Static
DUC [15]	ODRL Ontology	✗	✓	✗	✗	✓	Static
IntentKeeper [17]	ODRL Ontology	✗	✓	✗	✗	✓	Dynamic
ODRL-PAP [18]	ODRL (compiled to Rego)	✓	✓	✗	✗	✓	Static
MOSAICrOWN Policy Engine [19]	ODRL Ontology	✗	✓	✗	✗	✗	Dynamic
Interoperable Usage Control [20]	ODRL+Ext. (Dyn. Constraints)	✗	✓	✗	✓	✗	Dynamic
OTT Copyright Management System [21]	ODRL+Ext. (Copyright Terms)	✓	✓	✗	✓	~	~
ODRE [22]	ODRL+Enf. Layer	✗	✓	✓	✗	✗	Dynamic
ODRE-PDS (This work, 2025)	ODRL+Ext. (Formal Semantics)	✓	✓	✓	✓	✓	Dynamic

Table 2
Comparison of ODRL-based Policy Management Frameworks. Evaluation criteria: A) Policy Specification; B) Policy Management; C) Policy Enforcement; D) Access Control; E) Monitoring; F) Interface; G) Types of Constraints. *Legend:* ✓= full support, ✗= no support, ~ = partial or implicit support.

2.2. Summary

While previous efforts have contributed important mechanisms—such as compliance checking [14], REST APIs [15, 17], and enforcement logic [22]—none offers a complete, extensible solution that supports real-time enforcement, access control and monitoring. ODRE-PDS addresses this gap by providing a structured API that supports dynamic policy enforcement and is designed for integration with external trust mechanisms—although trust validation is not yet implemented in the current version. This approach helps transform ODRL from a purely descriptive language into an operational framework for privacy policy management.

3. Proposed Approach

This section presents the architecture of the ODRE Policy Directory Service (ODRE-PDS), a modular and extensible Web service for managing, discovering, and enforcing policies defined using the Open Digital Rights Language (ODRL). ODRE-PDS is designed to be deployed in decentralised environments where external components, such as authentication services, monitoring systems, or biometric verifiers, can interact and exploit its features.

Figure 1 shows an overview of the ODRE-PDS architecture. The service consists of four main components accessible through dedicated interfaces: the Management component, the Enforcement component, the SPARQL component, and the Triplestore. These components are interconnected and operate over the policies that are stored as a Knowledge Graph. All operations are available via a RESTful interface, although the architecture can be easily adapted to support other transport protocols besides HTTP.

The Management component handles the lifecycle of ODRL policies, including their creation, update, retrieval, and deletion. Policies that are registered must comply with the ODRL 2.2 specification, as they are validated both syntactically—e.g., ensuring correct Turtle syntax—and semantically, by verifying that all RDF terms conform to the expected structure defined by the ODRL ontology and that

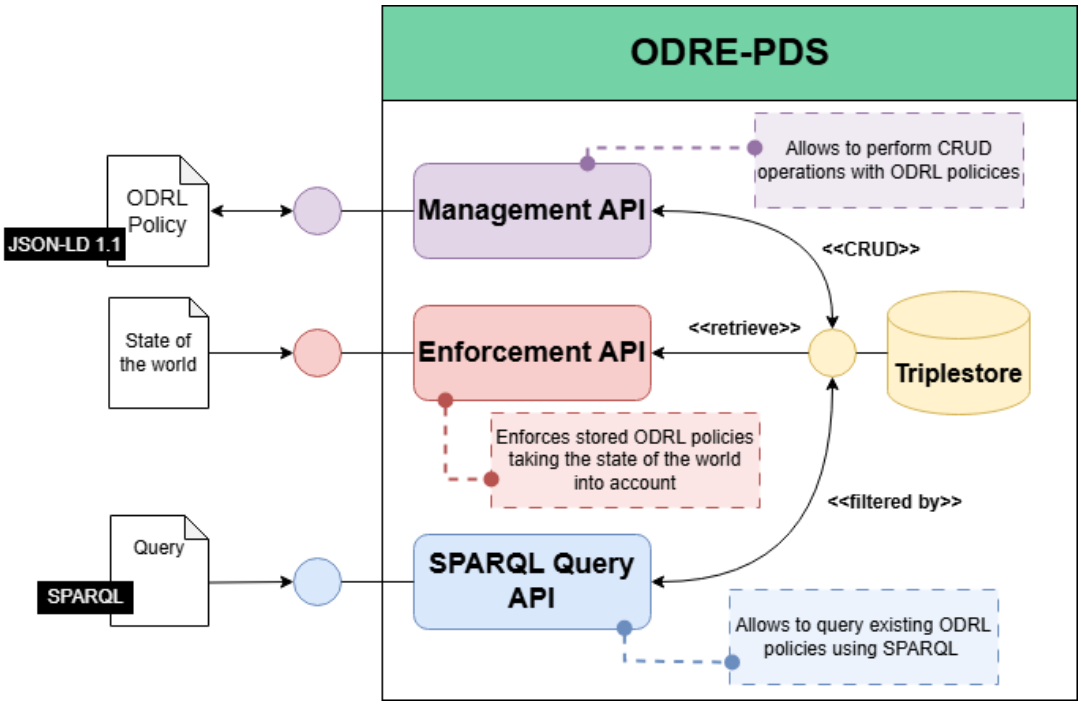

Figure 1: System architecture of the ODRE-PDS.

policy elements follow the vocabulary constraints (such as valid use of permissions, constraints, and actions) before being stored in the triple store. Policies can be submitted in RDF serialised as Turtle or JSON-LD 1.1; internally policies are stored in the Triplestore component using named graphs, due to this approach, each policy is stored individually, easing queries while maintaining provenance and traceability. The SPARQL component allows to perform queries for discovering or exploring policies stored in the ODRE-PDS (e.g., finding policies defined by a particular assigner or to target a specific asset).

Although it is not explicitly represented in the architectural diagram (Figure 1), the Evaluation Engine plays a central role in the enforcement process. This logical component is responsible for enforcing ODRL policies based on a state of the world that may include both internal and external information relative to the ODRE-PDS. This separation allows the architecture to remain modular and evaluation-agnostic, enabling the integration of alternative reasoning engines in future extensions.

When a request is received through the Enforcement API, relevant policies are first retrieved from the Knowledge Graph. In the envisioned architecture, policy relevance is determined by matching the target resource, the requested action, and, where applicable, contextual parameters derived from the state of the world. This allows the system to dynamically select only the applicable policies for a given access request.

Once the relevant policy is identified, contextual data—such as the current time, user identity, or device location—is gathered and injected into the evaluation process. The Evaluation Engine then assesses each rule defined in the policy individually, verifying whether its associated constraints—and, where applicable, refinements—are satisfied given the current state of the world. A rule is considered satisfied if all of its conditions hold. If one or more rules evaluate positively, the corresponding actions specified in those rules are either taken or executed by the ODRE-PDS or those actions are delegated to a third-party component or actor to be taken.

A key feature of the ODRE-PDS Enforcement API is its ability to operate in access control and monitoring scenarios. In the former, when a third party intends to take an action over a resource protected by a policy, the enforcement of that policy is triggered. The evaluation of the policy is performed based on a set of data from the state of the world that does not change during the enforcement process, such as a allow list for accessing a resource. In this case, the enforcement task finishes after

the evaluation, and the action is taken (either by the system or by a third party entity). In the latter scenario, the enforcement keeps evaluating the policy during the time window in which the third party keeps intending the action.

For example, let us assume that a document is protected by a policy. In an access control scenario, the policy may allow reading the document if valid credentials are provided. During enforcement, the credentials obtained from a third-party entity (i.e., the state of the world) are evaluated against those specified in the policy. If they match, the policy is considered satisfied, and the document is delivered as the result of the enforcement (i.e., the system executes the permitted action).

In a monitoring scenario, the policy may allow the document to be displayed only if an AI model detects a face associated with a unique identifier that is authorized to access it. In this case, enforcement requires continuous evaluation of the state of the world, which may change over time. If the AI-provided identifier no longer matches the one allowed in the policy, the system stops displaying the document. Note that the policy only performs the *odrl:display* action but it is not able to control if a practitioner is reading the document.

The different aforementioned features make ODRE-PDS particularly suitable for decentralised initiatives. On the one hand, it supports access control and monitoring scenarios, making ODRE-PDS suitable for a wide range of use cases. On the other hand, the decentralised nature of linked data (RDF) allow different ODRE-PDS instances to be deployed working in conjunction. For instance, the discovery based on SPARQL could be federated over multiple directories relying on the SERVICE statement of the SPARQL queries.

4. Implementation

To showcase the feasibility of the proposed architecture of the ODRE-PDS service, a Python-based implementation has been made publicly available in Git under an Apache 2.0 license[2] . The implemented service provides a RESTful interface built with *FastAPI* [3], allowing external applications to manage, evaluate and enforce ODRL policies in real time. The transport protocol used is HTTP since it is a W3C standard; however, the modular design allows future integrations with alternative communication protocols, such as CoAP or MQTT.

The Management and SPARQL components are developed using the *rdflib*[4] that handles different serialisations of RDF. This library is used to translate policies from JSON-LD 1.1 to Turtle serialisation or to perform SPARQL queries over a set of policies expressed in Turtle. Since the ODRL standard has not yet published a JSON-LD 1.1 frame, having a policy in JSON-LD 1.1 which has to be translated to Turtle and back to JSON-LD 1.1 obtaining the same identical policy as the original is complex and tricky; requiring multiple potential ad-hoc adjustments. Due to this reason, the ODRE-PDS implementation does not rely on a Triplestore but instead, this component stores directly the policies written in JSON-LD 1.1. Only when a SPARQL query is issued, the policies are translated to Turtle and the query performed. In addition to not having the frame, this implementation choice is motivated by the fact that it is expected to have more requests that need to retrieve policies rather than query requests; with this implementation, response times are optimised in these cases.

Finally, the Enforcement Evaluator is implemented based on the ODRE enforcement framework [22], in particular with the enforcement algorithm implemented in Python. This framework allows to enforce a policy providing a state of the world modelled as a JSON set of values. The ODRE-PDS implementation enhances the construction of the state of the world that can be used to enforce a policy by injecting values provided in a request to the Enforcement API sent as parameters in the URL. Following this approach, a policy can take into account the information that a requester may provide using URL parameters.

The implemented Enforcement API raises the question of trust in the values provided in the URL

[2]https://github.com/ODRE-Framework/policy-directory-service
[3]https://fastapi.tiangolo.com/
[4]https://rdflib.readthedocs.io/en/stable/

parameters that could be used to enforce policies. It would be interesting to implement a mechanism to trust who provides such values. For instance, a token-based system like JWT could be used so only authorized and authenticated entities could provide information to be taken into account. However, due to the academic nature of the current implementation, and the fact that the authors aim at providing a proof of concept for the components described, this feature will be further analysed and implemented in the future. This trust becomes particularly relevant and crucial in use cases like the one based on AI explained in the following subsections that relies on biometric recognition.

In order to facilitate its adoption and deployment in real-world scenarios, the ODRE-PDS implementation has been containerised. This simplifies its integration into cloud-based or on-premises infrastructures. In addition, all REST endpoints are documented with Swagger (Open API specification [5]) to ensure that developers can integrate ODRE-PDS with other services with minimal effort, delegating policy-related decisions to the directory while maintaining their existing infrastructures.

As a final remark, the ODRE-PDS implementation has been developed to support concurrent requests through asynchronous processing, enabling horizontal scalability across distributed instances. Stateless policy evaluation ensures low response times and high availability under load. As a result, the implementation presented validates the operational feasibility of the proposed architecture, offering a suitable building block for decentralised initiatives.

5. Practical Use cases

This section introduces two real-world use cases derived from research projects. The former use case belongs to the AURORAL European project, where the ODRE-PDS was used to access certain documents under certain temporal restrictions. The latter use case belongs to the GUIA project[6], where ODRE-PDS is used to allow the reading of confidential documents using biometric-based access. The usage of ODRE-PDS in these projects validates its adoption in real scenarios.

For the sake of privacy, the use cases described in the following subsections do not use the same resources as those they protect in the projects. In addition, to showcase these use cases, public endpoints have been enabled to see how they work. To this end, a public instance of the ODRE-PDS service has been deployed [7]. The *Time-Restricted Access Policies* use case can be accessed directly [8], whereas the *AI-Driven Access Control* use case has been made available through a third-party service [9], which usage is described in a video in the Zenodo repository[10].

5.1. Time-Restricted Access Policies

This use case illustrates how ODRE-PDS protects the access to a specific document based on time, ensuring that only requests within a valid time window are granted. The policy used in this scenario is publicly available in the Zenodo repository under the file *time base access policy.json*.

In this use case, a user attempts to access the European Union's Artificial Intelligence Act document. The access condition is defined using an ODRL policy where a constraint restricts the read action to requests made before 23:59:00 to 00:00:00 considering the time zone of the server where the service is deployed (CEST). Note that this restriction has been set for the sake of simplicity and reviewers' demo. The policy is expressed using RDF in JSON-LD format and stored in the directory, where it can be retrieved and evaluated using the RESTful interface exposed by the system. The constraint is encoded using the left operand *time:time*, a custom extension aligned with the ODRL ontology and published

[5] https://swagger.io/specification/

[6] See https://github.com/guia-project, Madrid Government Multiannual Agreement 2023-2026, Emerging PhD researchers, M230020126A-AJCA

[7] https://odrldirectory.linkeddata.es/docs

[8] https://odrldirectory.linkeddata.es/api/policy/evaluate/5000

[9] https://aifacerecognition.linkeddata.es/

[10] https://doi.org/10.5281/zenodo.15106825

by Cimmino et al. [22], and the evaluation is performed using the current system time computed at runtime.

When a user wants to access the document, such user must make a request to the ODRE-PDS service, in particular, a request to the Enforcement API. Then, the ODRE-PDS directory tries to find the relevant policy for such request based on the policy identifier; if no policy is found, the directory provides an empty response. Otherwise, the system updates its representation of the state of the world extracting potential URL parameters and formatting them accordingly. In this case, the current time is taken from the system and no parameters are extracted nor provided by the URL.

The retrieved policy and the state of the world are then passed to the ODRE framework, which evaluates the time constraint using its internal logic to process ODRL constraints. If the current time is before the allowed limit, the condition is satisfied, and the ODRE framework proceeds to retrieve the requested document from the storage layer (Document Store). The document is then returned to the user, completing the access control flow. In case the condition is not satisfied, the system returns an empty response with a denied access status.

This aforementioned workflow is illustrated by Figure 2, which depicts the sequence of interactions between the user, the ODRE-PDS, and the document store. The diagram outlines the enforcement flow, showing how policy retrieval, evaluation, and document retrieval are orchestrated.

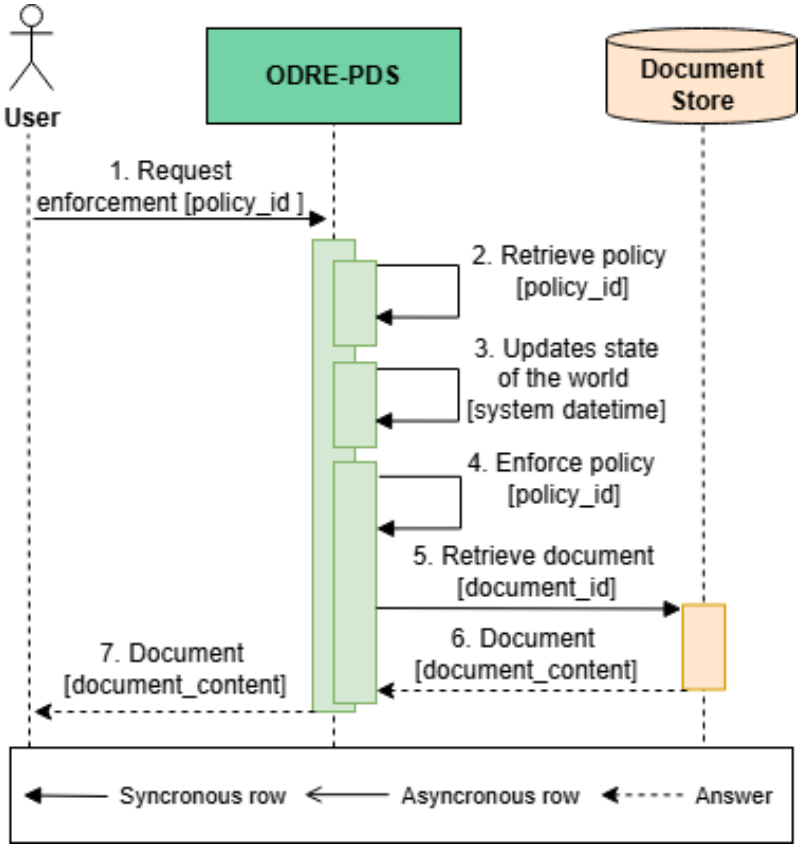

Figure 2: Sequence diagram of the time-restricted access control process.

5.2. AI-Driven Access Control

This use case illustrates how ODRE-PDS protects the access to a specific document based on AI-driven authentication mechanisms, such as facial recognition. This use case demonstrates how access to a restricted document is granted only to users who have been authenticated via a pre-trained AI model. By combining identity verification with policy enforcement, the system ensures that only authorised individuals can retrieve and consume a specific document. The policy used in this use case and a

video showcasing it are publicly available in the Zenodo repository under the file named *IA base access policy.json*.

An external system named the AI Service operates by performing facial recognition and linking its outputs to ODRL policies. When a user wants to read a document in the AI Service, this system captures and processes their facial features, generating a unique Universally Unique Identifier (UUID). The service then makes a request to the Enforcement API providing the UUID as a parameter in the URL. The ODRE-PDS receives the request and tries to determine whether a relevant policy exists or not. In the case it exists, the ODRE-PDS enforces the policy using ODRE and passing the UUID provided in the request as part of the state of the world. In the case the enforcement is positive, the ODRE-PDS provides to the AI Service the protected document. Take into consideration that the policy definition explicitly links permitted UUIDs with access conditions, ensuring that only pre-registered individuals can retrieve the document.

Note that the AI Service performs continuous recognition and, therefore, the ODRE-PDS is continuously enforcing the relevant policies. In the moment the AI Service stops providing a valid UUID the ODRE-PDS stops providing the document. As a result, the AI service can no longer display the document to the user. Note that revoking the document can only be achieved by stopping displaying it and having mechanism in the AI service to prevent copying or leak anyhow the document. The enforcement process of this use case follows the sequence diagram shown in Figure 3.

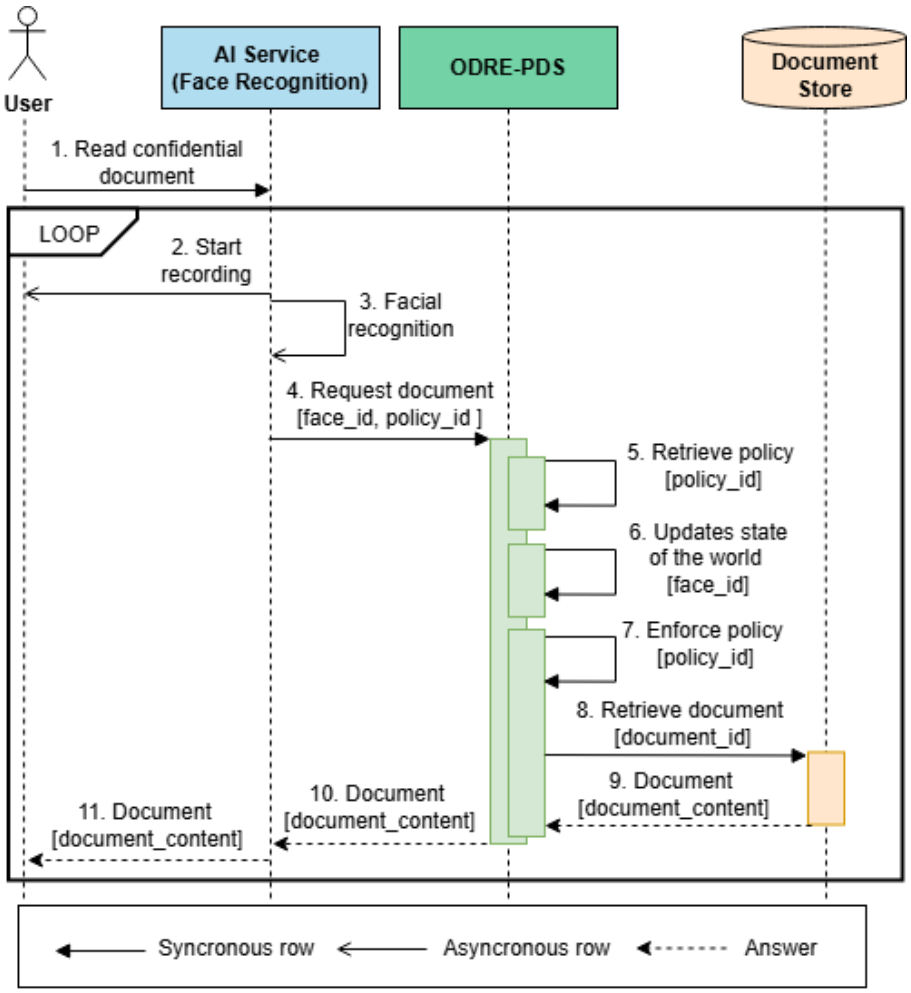

Figure 3: Sequence diagram of the AI-based access control process. 10)

The AI-driven access control provides several advantages over traditional access mechanisms. By relying on facial recognition and ODRL policies, it eliminates the need for password-based authentication, reducing the potential security risks associated with credential leakage. However, it entails a

technological challenge since it requires continuous enforcing of the relevant policies.

As a final remark, this use case shows how the ODRE-PDS is an out of the box solution for many complex scenarios. Since the policy enforcement logic is decoupled from the authentication mechanism, organizations can integrate different AI-based verification services without modifying the enforcement pipeline. This flexibility makes the ODRE-PDS well-suited for privacy-sensitive environments where identity validation must be performed in compliance with strict regulatory standards.

6. Evaluation

This section presents the evaluation performed for the ODRE-PDS implementation, which focuses on analysing its efficiency and scalability. The evaluation has been carried out by performing the following experiments: (i) policy enforcement performance, (ii) system scalability under concurrent requests, and (iii) overhead introduced by ODRE-PDS in comparison to the ODRE framework. All experiments were performed using the deployed ODRE-PDS and the *pyodre*[11] implementation of ODRE.

6.1. Policy Enforcement Performance

To evaluate the efficiency of the Enforcement API, multiple access requests were executed using three policies. The first policy, taken from the ODRL standard[12], encodes a restriction to access a resource based on date and time, the second policy used is the one from the *Time-Restricted Access Policies* use case, and the third policy is the one used in the *AI-Driven Access Control* use case; the second and third policies can be found in the Zenodo repository.

In this experiment, for each policy, 30 requests were made and their response times were recorded. Then, all these values were averaged using the arithmetic mean. Figure 4 shows the results of the experiment. It can be seen that the response time increases proportionally depending on how the directory and the enforcement component evaluate them. While date-based policies are processed quickly, the rest introduce additional latency. In any case, the results show that ODRE-PDS is able to fulfil the requests in less than a second for the three policies.

6.2. Scalability and Concurrency Analysis

To evaluate the scalability of the Enforcement API in this experiment, concurrent requests to such API are simulated using 10, 50, and 100 parallel requests and recording their response time. These requests enforce the policy containing the date-time constraint.

Figure 5 shows the results obtained in this experiment. It can be observed that the average latency grows linearly with the number of concurrent requests. This behaviour, and the way it grows, indicates that ODRE-PDS can be deployed in multi-user environments with increasing load, while maintaining reasonable response times under stress.

6.3. Overhead Comparison with ODRE

To evaluate the overhead introduced by the REST-based infrastructure of ODRE-PDS, in this experiment, the response times to enforce the same policy (with a simple date-time constraint) using the *pyodre* engine and the Enforcement API of ODRE-PDS.

In this experiment, the policy was enforced 30 times using both API and *pyodre* and the time they required to finish recorded. Then, all these values were averaged using the arithmetic mean. Figure 6 shows the results of this experiment. It can be observed that ODRE-PDS introduces additional latency (0.0617 s on average versus 0.0335 s *pyodre*), the overhead remains acceptable for real-time scenarios. Most of the added latency comes from request processing and serialisation overhead.

[11] https://github.com/ODRE-Framework/odre-python
[12] https://www.w3.org/TR/odrl-model#constraint-rule

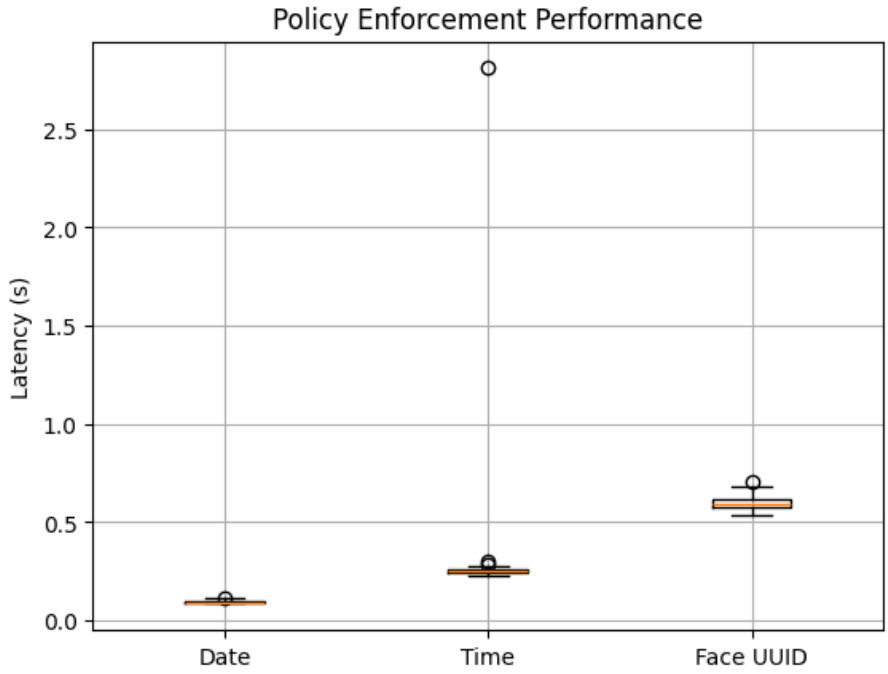

Figure 4: Results of the *Policy Enforcement Performance* experiment.

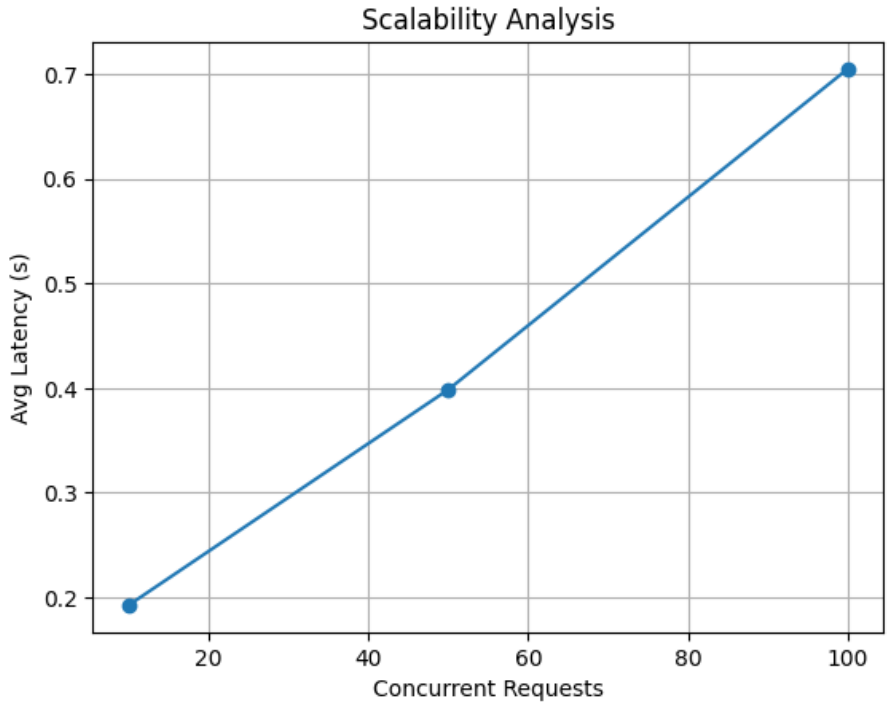

Figure 5: Results of the *Scalability and Concurrency Analysis* experiment.

7. Conclusions

This article has presented the ODRE Policy Directory Service (ODRE-PDS), a novel Web-based architecture designed to address current limitations of ODRL when adopted in practical scenarios or decentralised initiatives. The ODRE-PDS provides three main features related to policies, namely: management, discovery, and enforcement. Furthermore, the article presents an implementation of this

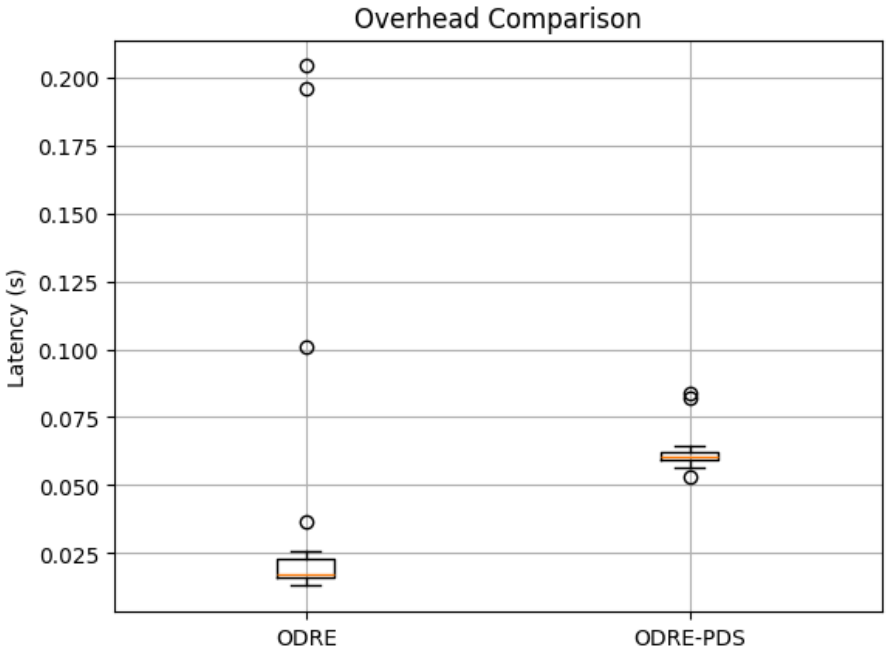

Figure 6: Results of the *Overhead Comparison with ODRE* experiment.

proposal coded in Python and publicly available.

Beyond providing operational implementation, ODRE-PDS moves ODRL a step closer to being fully usable in real-world systems, extending its scope from policy specification to practical enforcement, real-time evaluation, and service integration. In this sense, ODRE-PDS transforms ODRL from a purely descriptive language into a functional framework capable of supporting decentralised, privacy-aware ecosystems. Its modular and flexible design makes it easier to plug into real-world systems, helping bridge the gap between what is written in policies and what is actually enforced.

In order to evaluate the ODRE-PDS implementation in terms of performance and scalability, the article presents three experiments. The results advocate that the ODRE-PDS implementation is a suitable out-of-the-box solution in real world scenarios. However, some limitations should be taken into account, namely: the lack of trust mechanisms to verify the information provided by third-party entities for policy enforcement, i.e., those provided via URL parameters. This limitation is especially relevant in scenarios involving enforcement based on attributes provided by decentralised systems, where malicious or unverified input may compromise the enforcement decision. In addition, the system currently lacks built-in compliance logging mechanisms to support auditable traceability. Finally, The current validation mechanisms and SPARQL interface could both be improved to align with more characteristics of SPARQL 1.1. In the future, the authors aim to address the previous limitations. In addition, the authors plan to explore the integration of ODRE-PDS into cross-domain infrastructures.

Acknowledgements

This work has been partially supported by: the Madrid Government (Comunidad de Madrid-Spain) under the Multiannual Agreement 2023-2026 with UPM in Line A, Emerging PhD researchers through the project GUIA (M230020126A-AJCA); the European Union's Horizon 2020 Research and Innovation Programme of the European Union through the AURORAL project (101016854); and the Next Generation EU through the STICS project (09I02-03-V01).

8. Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT, Grammarly in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] I. D. S. Association, Technical Agreements, in: IDSA Rulebook, 2024. URL: https://docs.internationaldataspaces.org/ids-knowledgebase/idsa-rulebook/idsa-rulebook/4_technical_agreements.

[2] B. Esteves, H. J. Pandit, V. Rodríguez-Doncel, ODRL Profile for Expressing Consent through Granular Access Control Policies in Solid, in: IEEE European Symposium on Security and Privacy Workshops, EuroS&P 2021, Vienna, Austria, September 6-10, 2021, IEEE, 2021, pp. 298–306. URL: https://doi.org/10.1109/EuroSPW54576.2021.00038.

[3] S. Steyskal, A. Polleres, Defining expressive access policies for linked data using the ODRL ontology 2.0, in: H. Sack, A. Filipowska, J. Lehmann, S. Hellmann (Eds.), Proceedings of the 10th International Conference on Semantic Systems, SEMANTiCS 2014, Leipzig, Germany, September 4-5, 2014, ACM, 2014, pp. 20–23. URL: https://doi.org/10.1145/2660517.2660530.

[4] Z. Maamar, A. Benna, H. Kechaoui, ODRL-Based Provisioning of Thing Artifacts for IoT Applications, in: H. Kaindl, M. Mannion, L. A. Maciaszek (Eds.), Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE 2024, Angers, France, April 28-29, 2024, SCITEPRESS, 2024, pp. 168–178. URL: https://doi.org/10.5220/0012718600003687.

[5] R. Cimmino, Andrea and Cano-Benito, Juan and García Castro, The AURORAL Privacy Approach for Smart Communities Based on ODRL, in: International Summit on the Global Internet of Things and Edge Computing, Springer, 2024, pp. 89–100.

[6] D. Golpayegani, B. Esteves, H. J. Pandit, D. Lewis, AIUP: an ODRL Profile for Expressing AI Use Policies to Support the EU AI Act, in: D. Garijo, A. L. Gentile, A. Kurteva, A. Mannocci, F. Osborne, S. Vahdati (Eds.), Joint Proceedings of Posters, Demos, Workshops, and Tutorials of the 20th International Conference on Semantic Systems co-located with 20th International Conference on Semantic Systems (SEMANTiCS 2024), Amsterdam, The Netherlands, September 17-19, 2024, volume 3759 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3759/paper17.pdf.

[7] Monegraph, Renato Iannella and Villata, Serena, ODRL Information Model 2.2, in: W3C Recommendation, 2018.

[8] O. Standard, eXtensible Access Control Markup Language (XACML) Version 3.0, A:(22 January 2013). URl: http://docs. oasis-open. org/xacml/3.0/xacml-3.0-core-spec-os-en. html (2013).

[9] R. Falcão, A. Hosseinzadeh, Towards a Decentralized Data Privacy Protocol for Self-Sovereignty in the Digital World, in: J. Araújo, J. L. de la Vara, N. Condori-Fernández, J. Bruel, M. Y. Santos, S. Assar, K. D. Moor, M. Gharib, T. Li, J. P. Barros, I. S. Brito, I. Machado, D. Karagiannis, T. P. Sales, C. Salinesi (Eds.), Joint Proceedings of RCIS 2024 Workshops and Research Projects Track co-located with the 18th International Conferecence on Research Challenges in Information Science (RCIS 2024), Guimarães, Portugal, May 14-17, 2024, volume 3674 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3674/ASPIRING-paper1.pdf.

[10] I. Akaichi, S. Kirrane, A comprehensive review of usage control frameworks, Comput. Sci. Rev. 56 (2025) 100698. URL: https://doi.org/10.1016/j.cosrev.2024.100698.

[11] J. Park, R. S. Sandhu, The UCON_{ABC} usage control model, ACM Trans. Inf. Syst. Secur. 7 (2004) 128–174. URL: https://doi.org/10.1145/984334.984339.

[12] M. Palmirani, G. Governatori, T. Athan, H. Boley, A. Paschke, A. Wyner, LegalRuleML Core Specification Version 1.0, 2021. URL: https://docs.oasis-open.org/legalruleml/legalruleml-core-spec/

v1.0/os/legalruleml-core-spec-v1.0-os.html, latest stage: https://docs.oasis-open.org/legalruleml/legalruleml-core-spec/v1.0/legalruleml-core-spec-v1.0.html.

[13] A. Cimmino, J. Cano-Benito, R. García-Castro, Practical challenges of ODRL and potential courses of action, in: Y. Ding, J. Tang, J. F. Sequeda, L. Aroyo, C. Castillo, G. Houben (Eds.), Companion Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023, ACM, 2023, pp. 1428–1431. URL: https://doi.org/10.1145/3543873.3587628.

[14] M. D. Vos, S. Kirrane, J. A. Padget, K. Satoh, ODRL Policy Modelling and Compliance Checking, in: P. Fodor, M. Montali, D. Calvanese, D. Roman (Eds.), Rules and Reasoning - Third International Joint Conference, RuleML+RR 2019, Bolzano, Italy, September 16-19, 2019, Proceedings, volume 11784 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 36–51. URL: https://doi.org/10.1007/978-3-030-31095-0_3.

[15] A. Munoz-Arcentales, S. López-Pernas, A. Pozo, Álvaro Alonso, J. Salvachúa, G. Huecas, An Architecture for Providing Data Usage and Access Control in Data Sharing Ecosystems, Procedia Computer Science 160 (2019) 590–597. URL: https://www.sciencedirect.com/science/article/pii/S1877050919317429, the 10th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2019) / The 9th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2019) / Affiliated Workshops.

[16] A. Munoz-Arcentales, S. López-Pernas, A. Pozo, Á. Alonso, J. Salvachúa, G. Huecas, Data Usage and Access Control in Industrial Data Spaces: Implementation Using FIWARE. Sustainability 12, 9 (2020), 38–85, 2020.

[17] F. Cirillo, B. Cheng, R. Porcellana, M. Russo, G. Solmaz, H. Sakamoto, S. P. Romano, IntentKeeper: Intent-oriented Data Usage Control for Federated Data Analytics, in: 2020 IEEE 45th Conference on Local Computer Networks (LCN), 2020, pp. 204–215. doi:10.1109/LCN48667.2020.9314823.

[18] S. Wiesner, ODRL-PAP: Policy Administration Point to handle ODRL policies, https://github.com/wistefan/odrl-pap, 2021.

[19] A. O'Mahony, A. Barnett, M. Globin, Using automotive property graph-based data models in a knowledge graph, 2021. URL: https://api.semanticscholar.org/CorpusID:250165724.

[20] I. Akaichi, W. Slabbinck, J. A. Rojas, C. V. Gheluwe, G. Bozzi, P. Colpaert, R. Verborgh, S. Kirrane, Interoperable and Continuous Usage Control Enforcement in Dataspaces, in: J. Theissen-Lipp, P. Colpaert, S. K. Sowe, E. Curry, S. Decker (Eds.), Proceedings of the Second International Workshop on Semantics in Dataspaces (SDS 2024) co-located with the 21st Extended Semantic Web Conference (ESWC 2024), Hersonissos, Greece, May 26, 2024, volume 3705 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3705/paper10.pdf.

[21] W. Son, S. Kwon, S. Oh, J.-H. Lee, Automated Over-the-Top Service Copyright Distribution Management System Using the Open Digital Rights Language, Electronics 13 (2024). URL: https://www.mdpi.com/2079-9292/13/2/336.

[22] Andrea Cimmino and Juan Cano-Benito and Raúl García-Castro, Open Digital Rights Enforcement framework (ODRE): From descriptive to enforceable policies, Computers & Security 150 (2025) 104282. doi:https://doi.org/10.1016/j.cose.2024.104282.

[23] N. Fornara, V. Rodríguez-Doncel, B. Esteves, S. Steyskal, B. W. Smith, ODRL Formal Semantics, Draft Community Group Report, 2025. URL: https://w3c.github.io/odrl/formal-semantics/.

[24] M. Sensoy, T. J. Norman, W. W. Vasconcelos, K. P. Sycara, OWL-POLAR: A framework for semantic policy representation and reasoning, J. Web Semant. 12 (2012) 148–160. URL: https://doi.org/10.1016/j.websem.2011.11.005.

[25] M. D. Vos, S. Kirrane, J. A. Padget, K. Satoh, ODRL policy modelling and compliance checking, in: P. Fodor, M. Montali, D. Calvanese, D. Roman (Eds.), Rules and Reasoning - Third International Joint Conference, RuleML+RR 2019, Bolzano, Italy, September 16-19, 2019, Proceedings, volume 11784 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 36–51. URL: https://doi.org/10.1007/978-3-030-31095-0_3.