Towards an Accessible Reference ODRL Policy Editor

Christopher Maidens[1], Paolo Pareti[1] and George Konstantinidis[1]

[1]University of Southampton, Southampton, UK

Abstract

The Open Digital Rights Language (ODRL) is a standardised machine-readable vocabulary and policy expression language, maintained and developed by W3C. It allows data providers and consumers to explicitly define and agree policies that can be used to automatically manage the permissions, prohibitions, and obligations controlling the usage of digital content and services. ODRL is being increasingly adopted to help automate compliance within ever more complex machine driven commercial ecosystems. However, the ODRL specification is complex for non-specialist users and there is a lack of comprehensive user-facing tooling to assist in the creation of the required policies. Therefore, the creation of machine processable policies remains difficult for business domain experts who may not be specialists in ODRL and associated semantic development standards (e.g. JSON-LD and RDF Turtle). Here we present the UoS Policy Editor (an ODRL authoring and validation tool) towards solving this problem. Demo attendees will interactively build example policies based on representative business examples to demonstrate how using this editor can simplify the creation of the required machine-readable policies. We hope that this work will underpin the creation of a reference editor capability to enhance the evolution of the full ODRL specification.

Keywords

ODRL, Policy Management, Usage Control, Semantic Web, Interactive Demonstration

1. Introduction

Decentralized data spaces (such as Gaia-X and IDSA) and other distributed data ecosystems are increasingly adopting ODRL. It is intended that this adoption should automate (or assist with human-in-the-loop partial automation) of processes related to the management of negotiation and subsequent compliance evaluation required for distributed data processing. The ODRL Information Model [1], Vocabulary [2] and Semantics [3] provide for a very expressive standard for policy definition. However, although expressive, it is also complex and business domain experts do not always have the knowledge of the logic and semantic standards required to build valid and precise policies. Therefore, the need for editing tools that can help more easily translate business and legal domain expertise into machine processable policies is growing. To this end we are developing an ODRL editor which is currently available as an open-source demonstration tool. We wish this editor to provide users with an easy to use graphical interface that allows them to more easily build and output policies aligning to the underlying ODRL Information model, standardized / user-supplied vocabularies and semantics .

Such an editor is actually not just a tool to create policies. It is also gateway for the broader community, including business domain experts, to embrace and benefit from easier access to ODRL and it's ability to assist in the automation of policy compliance within their organizations. We see such a reference editor sitting alongside the specifications and potentially other extended reference tooling to provide a user-focused entry point to the full and implicit capabilities.

2. The Open Digital Rights language

The Open Digital Rights Language (ODRL) is a W3C Recommendation that provides a standardized, machine-processable format for expressing policy statements about the usage of digital assets, content

ISWC 2026 Companion Volume, October 25–29, 2026, Bari, Italy

*Corresponding author.

†These authors contributed equally.

c.maidens@soton.ac.uk (C. Maidens); p.pareti@soton.ac.uk (P. Pareti); g.konstantinidis@soton.ac.uk (G. Konstantinidis)

0000-0002-4385-7202 (C. Maidens); 0000-0002-2502-0011 (P. Pareti); 0000-0002-3962-9303 (G. Konstantinidis)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

and services [1] [2] [3] [4]. ODRL establishes a flexible information model and vocabulary **to define** who can perform **what action** on **which resource**, and under **what conditions**. Policies are built using four key components. Firstly *Assets* defining the digital content or services being governed. Secondly *Parties* defining the stakeholders taking on roles such as assigners or assignees. Thirdly *Actions* defining the operations permitted, prohibited, or mandated (e.g., read, use, modify, distribute). Fourthly *Rules* defining the core logic governing the asset, these include Permissions (what is allowed), Prohibitions (what is strictly forbidden), and Duties/Obligations (what must be fulfilled, such as making a payment). Rules can also be tightly controlled using *Constraints* (such as temporal or spatial) and *Refinements*. Widely serialized using JSON-LD, XML, or RDF/Turtle, ODRL is extensively utilized across modern data spaces, AI governance frameworks, digital rights management, and open data platforms to automate secure and transparent policy compliance between systems.

3. Related Work

The ODRL-PAP (Policy Administration Point) policy editor is designed to manage Gaia-X ODRL policies and templates [5]. It helps users to create and manage usage policies and reusable policy templates without needing to write raw ODRL code manually. It can be run standalone or integrated as an embeddable web component. It provides Rego-equivalent policies to the Open Policy Agent. A cut down web based policy editor may be found at [6], providing a nice clean interface to creation of simple policies. A great deal of additional information about related ODRL initiatives may also be found at [7]. Although not specifically related to ODRL policy creation, future work to increase guided editing capabilities may take cues from examples such as the Creative Commons Licence Chooser [8]. Similarly, the free open source expert system Docassemble ”a free, open-source expert system for guided interviews and document assembly, based on Python, YAML, and Markdown” [9].

4. Core Features and System Architecture

The demonstration implements a full-stack, web-based editor for creating and validating ODRL policies. The application is split between a modular Node.js React application frontend and a Python FastAPI backend.

4.1. GUI Summary

As mentioned in the introduction we believe that a reference editor should complement the ODRL specifications and additional tooling to enhance ease of access to it’s capabilities for non-specialist users. In this development we are aiming to help bring ODRL ’to life’ through this ODRL authoring and validation tool to aid business domain leaders and effectively guide users through compliant use of the ODRL Information Model and associated Vocabularies in order to create ODRL compliant machine processable policies representing business requirements. A summary of the GUI capabilities described below is provided by the screenshot presented in figure 1

4.2. Key Features

A summary of the key features are summarised below:

Policy Metadata Management Configures core policy properties such as policy type (Agreement, Offer, Set), Unique Identifiers (UIDs), profiles, assigners, assignees, conflict strategies, and global target assets.

The purpose of the ODRL metadata section is to provide administrative, structural, and contextual information about the policy itself, rather than governing the individual rules of asset usage.

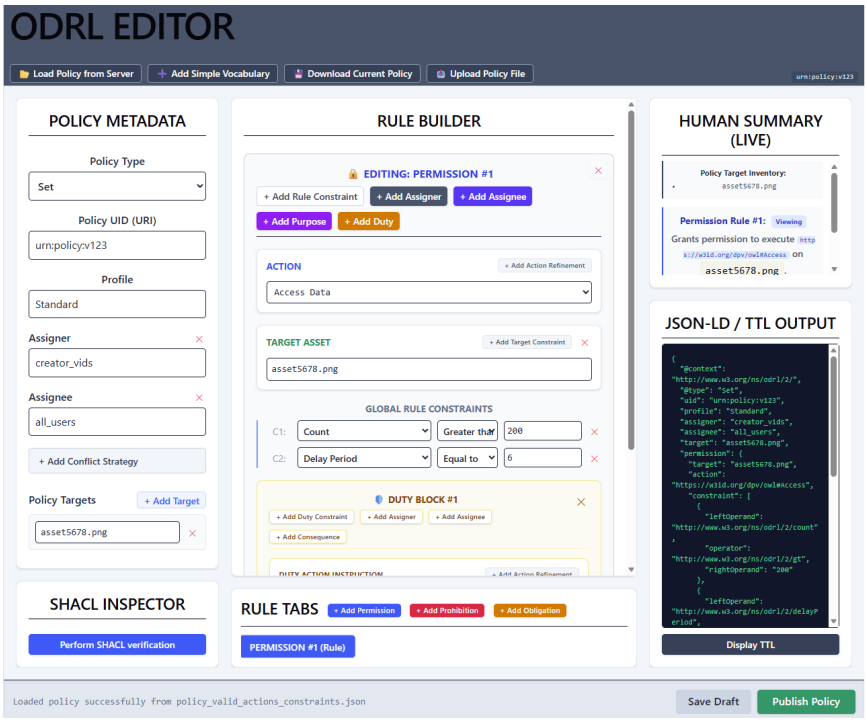


Figure 1: Illustrative screenshot of the policy editor

Interactive Rule Builder Allows users to add, edit, and remove granular Permission, Prohibition and Obligation blocks complete with actions, refinements, assigners, assignees, purposes, targets, and conditional Duties with associated consequences.

This is the core of the editor's capabilities. The optionality within the information model is complex. The rule builder and tabs panel allow a user to more simply introduce new rules within a policy. Structure within the rules is guided through the rule builder panel allowing the user to create simple and complex rule definitions that are ODRL specification compliant.

Live Human-Readable Policy and Translation (JSON-LD and TTL) Automatically compiles the user's visual inputs into standard ODRL JSON-LD documents in real-time. A human-readable live summary panel is also provided.

Allowing for human-readable summaries of the policies being defined through the rule sets aids users in understanding the wider context of policies being created. Live translation to and display of machine-readable versions of the policies also aids better integration of business and semantic domain dialogue during policy design process.

SHACL Compliance Verification Integration with a Python validation engine to evaluate policies against ODRL SHACL shapes, displaying conformity status and detailed error reports.

In addition to dynamic compliance guidance and validation during creation of policies the user may also perform SHACL based validation of the live policy to further assure structural compliance

Vocabulary Extension Tools Supports a vocabulary manager for extending dropdown vocabularies within the editor. Users may load their own vocabularies or create simple vocabularies through an editor dialogue.

Use of profiles and additional vocabularies is important to enhance domain specific expressibilty. For simple vocabularies the user can extend dropdowns through an edit interface.

A usage scenario for the editor is described in figure 2

4.3. Architecture Overview

There are two major application architecture components:

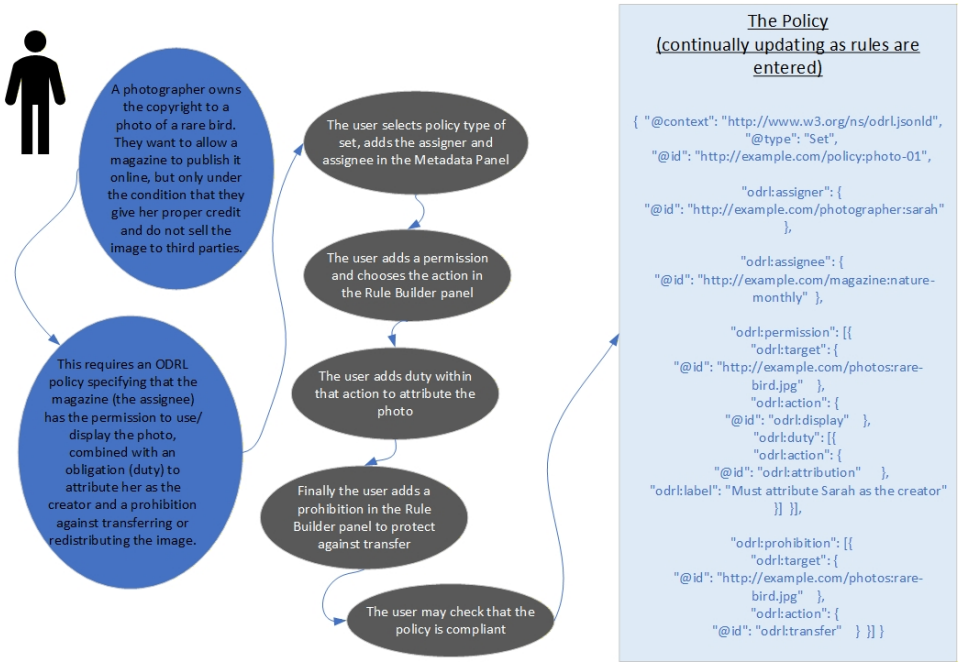


Figure 2: Illustrative summary of editing process from requirements to policy

Frontend (React)

The core UI component provides a workspace layout consisting of a policy metadata panel, policy validation panel, a rule builder canvas, a rule overview panel, a live human-readable summary view and a live policy preview console (JSON-LD or TTL format). The policy metadata panel, rule builder and rule overview panels aid the user in creating policies that structurally comply with the ODRL Information Model. Dropdowns within these elements are populated with the standard ODRL and DPV vocabularies, however importantly these vocabularies may also be augmented by user supplied vocabulary definitions (e.g. domain specific vocabularies and ODRL Profiles).

Backend (FastAPI / RDFLib / PySHACL)

Vocabulary and SPARQL Engine Loads semantic vocabularies (Turtle/RDF) into memory and exposes SPARQL query endpoints to dynamically populate dropdown selections within the editor GUI.

Validation Pipeline Manages and executes structural SHACL compliance checks using ‘pyshacl‘.

5. Demo Structure

During the ISWC poster and demo session, we will : 1) Discuss a number of representative use cases to illustrate how business requirements may be represented as ODRL policies; 2) the examples we will use will be drawn (in an anonymised form) from real projects where we have developed ODRL policy expressions being used in enterprise ecosystems; 3) we will then demonstrate how these may be implemented through the editor and how these align to the underlying specifications.

In this way we will demonstrate how the editor eases the creation of the machine-readable policy for the business domain experts.

6. Discussion and Future Directions

We believe that a freely available ’reference’ ODRL editor would complement the comprehensive specification resources further helping adoption by the business domain and process community. In this demonstration version of such an editor we have created a working model to illustrate creation of compliant policies. We have tried to create a comprehensive set of initial capabilities that can

already be used to create rich and compliant policies, but we also see opportunities for interesting areas for future consideration. We summarise some of these here. **AI-Assisted Policy Generation** Developing support for the automatic creation of draft policy rules, constraints, and refinements directly from initial human language input. And further implementing capabilities to create initial ODRL policies from business, contractual or legal text as well as improving natural language expression of existing raw ODRL policies. **Improving flexibility in modularity of the design** Developing flexibility of the application architecture to aid easier embedding into varied user workflows, this is motivated by consideration of reuse of the editor (or subsets of) capabilities within end to end business architectural implementations. **Improving empirical evaluation** This is currently in active and ongoing development. We would like to carry out studies and related assessments to understand better how this approach (and additional developments) can actively assist in making the journey from business objectives to ODRL policy creation easier. **Extensions to graphical capabilities** Consideration of enhancements to how policies may be described to users (such as a more graphical 'policy as a canvas' display of a policy and its rule hierarchies). Consideration of styling customizations. **Addition of a simulation pane** An area where a user can mock up a request and receive feedback on whether the request is allowed. For example a user may ask the question "If Party X attempts to print Asset Y at a particular time and location" and then receive feedback on whether the policy will allow such an action. **Handling Unknowns ('undefined')** Fully supporting this ODRL capability. **Policy Inheritance** Fully supporting this ODRL capability. **Profile Conformance ('profile')** Increasing support for this ODRL capability. **Administrative Tracking (Dublin Core Terms)** Fully supporting this ODRL capability. **Enhanced logical rule conflict validation** Enhancing current SHACL support to consider logical rule conflicts (e.g. [10])

7. Demo resources

An installable version of the latest development version of the tool may be found at https://github.com/DIPS-Tools/ODRL_Editor_Demo_v2. This includes additional supporting documentation. A running demo service may also be accessed at https://dips.soton.ac.uk/odrl-editor-demo/

Acknowledgments

This work was partially funded by the Horizon Europe projects DATAPACT (101189771) and RAISE Suite (101188337).

Declaration on Generative AI

During the preparation of this work, the author(s) used Gemini in order to: Grammar and spelling check, Paraphrase and reword. The authors also made use of Gemini in order to help create the demo software. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] W3C, Odrl information model 2.2, 2018. URL: https://www.w3.org/TR/odrl-model/.
[2] W3C, Odrl vocabulary and expression 2.2, 2018. URL: https://www.w3.org/TR/odrl-vocab/.
[3] W3C, Odrl formal semantics, 2018. URL: https://w3c.github.io/odrl/formal-semantics/.
[4] W3C, Odrl best practices, 2018. URL: https://w3c.github.io/poe/bp/.
[5] SEAMWARE, odrl-pap: Policy administration point to handle odrl policies and provide their rego-equivalent to the open policy agent, GitHub, 2026. URL: https://github.com/SEAMWARE/odrl-pap.
[6] CODATA, Odrl policy builder, Web, 2026. URL: https://odrl.dev.codata.org/policies.

[7] Wout Slabbinck, Odrl landscape, Web, 2026. URL: https://w3c.github.io/odrl/landscape/.
[8] Creative Commons, Creative commons licence chooser: Choose a license for your work, Web, 2026. URL: https://creativecommons.org/chooser/.
[9] Jonathan Pyle, Docassemble: A free, open-source expert system for guided interviews and document assembly, Web, 2026. URL: https://docassemble.org/.
[10] J. Salas, P. Pareti, G. Konstantinidis, Odrl policy comparison through normalisation, The Semantic Web. ESWC 2026. Lecture Notes in Computer Science 16549 (2026). doi:10.1007/978-3-032-25156-5_17.