

Enforcing Usage Control Policies in Solid using Rule-Based Web Agents

Wout Slabbinck^{1,*}, Julián Andrés Rojas¹, Beatriz Esteves¹, Ruben Verborgh¹ and Pieter Colpaert¹

¹IDLab, Departement of Electronics and Information Systems, Ghent University - imec, Belgium

Abstract

A core feature of the Solid ecosystem is enabling the sharing of data resources with other agents using access control policies. However, the decision of allowing access (or not) to a given data resource may not be final and might need to change over time. The Solid authorization specifications do not allow expressing and thus neither enforcing complex constraints (e.g., temporal) on access or more generally, usage policies. A policy language that does have the expressivity to declare permission rules, including temporal constraints, is the *Open Digital Rights Language* (ODRL) W3C standard. To support and enforce ODRL policies over Solid resources, we design and implement a Web agent-based solution where an agent (i) decomposes ODRL policies into actionable tasks (such as granting and retracting access to resources) using declarative condition-action rules and; (ii) takes care of executing such tasks. Usage control within Solid can be delegated to agents such that neither applications nor users within the ecosystem need to ensure that access permissions over their data resources are valid and up to date. We show how the expressivity limitations for usage control of the current Solid specifications could be addressed by supporting ODRL via long-running Web agents with the task of enforcement. Since currently all data-sharing actions within the Solid ecosystem are manually executed by the owner of the resource, future work includes automating policy-based negotiation processes among actors through Web agents.

Keywords

Solid, Access Control, Usage Control, Enforcement, Policy, Intelligent software web agents

1. Introduction

The Solid protocol¹ is a Personal Data Store Web technology that enables individuals to store and govern data on their data space, in Solid terminology also referred to as *pod*. A Solid server must support at least one of the two authorization specifications, namely the Web Access


2nd Solid Symposium, 2-3 May 2024, Leuven

*Corresponding author.

✉ wout.slabbinck@ugent.be (W. Slabbinck); julianandres.rojasmelendez@ugent.be (J. A. Rojas); beatriz.esteves@UGent.be (B. Esteves); ruben.verborgh@ugent.be (R. Verborgh); pieter.colpaert@ugent.be (P. Colpaert)

🌐 <https://woutslabbinck.com/> (W. Slabbinck); <https://julianrojas.org/> (J. A. Rojas); <https://besteves4.github.io/> (B. Esteves); <https://ruben.verborgh.org/> (R. Verborgh); <https://pietercolpaert.be/> (P. Colpaert)

🆔 0000-0002-3287-7312 (W. Slabbinck); 0000-0002-6645-1264 (J. A. Rojas); 0000-0003-0259-7560 (B. Esteves); 0000-0002-8596-222X (R. Verborgh); 0000-0001-6917-2167 (P. Colpaert)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

¹<https://solidproject.org/TR/protocol>

Control (WAC)² specification and/or the Access Control Policy (ACP) Language³, such that a user can declare access control rules over resources in their pod. To this date, however, these specifications do not support fine-grained access rules with temporal conditions. As a result, users must either for revocation re-configure the access control resources again after access was granted, or use a Solid application that can perform the granting and revoking access. However, neither option is infallible. In the first option, users must adjust rules to revoke access, yet human forgetfulness may allow unintended authorizations to persist. For the second option, due to the Web-based nature of the Solid protocol, the application must remain open and active for revocation to occur. This means that a Web client application is not sufficient. The first aspect to overcome the limitations of current specifications is to use another standard which supports fine-grained conditions to declare and enforce policies such as the Open Digital Rights Language (ODRL)⁴ W3C standard [1]. Since Solid only supports the aforementioned specifications, ODRL can not directly be enforced. However, it is possible to partially materialize ODRL to existing access rules that the Solid protocol can enforce. Furthermore, a continuously running Web service is required to overcome the manual and Web client revocation limitation.

In this paper, we extend an open-source, rule-based Web agent [2] to (i) perceive policies from an ODRL policy Knowledge Graph (KG), managed by the resource owner, (ii) interpret those ODRL policies and transform them to Access Control Resources (ACR) rules as defined by ACP, and (iii) perform long-running tasks to enforce temporal usage control policies.

2. Related Work

Sambra et al. [3] introduced Solid with the aim of letting users control their data on the Web by decoupling data from applications. To reach this goal, the protocol builds upon existing W3C standards and specifications to define a data storage, commonly referred to as Solid pod despite the lack of a precise definition [4]. The Linked Data Platform⁵ (LDP) W3C standard [5] is leveraged in Solid to read and write to this storage, which consists of a collection of resources. An agent is identified through a WebID⁶ and through Solid-OIDC⁷ the claim of their identity can be verified by an authoritative Identity Provider (IDP). The current access control protocols, WAC and ACP, allow users to define which party (or agent) can act on a given resource.

However, complying with the General Data Protection Regulation (GDPR) [6] is impossible since extra requirements such as the purpose of data use (Article 13.1(c) and Article 14.1(c)) and retention period (Article 13.2(a) and Article 14.2(a)) can neither be encoded in WAC nor ACP. To overcome these challenges, Esteves et al. [7] introduced the ODRL Profile for Access Control⁸ (OAC) which aims to align access and usage permissions with GDPR for Solid. This is achieved through modelling usage control policies with the Open Digital Rights Language

²<https://solidproject.org/TR/wac>

³<https://solidproject.org/TR/acp>

⁴<https://www.w3.org/TR/odrl-model/>

⁵<https://www.w3.org/TR/ldp/>

⁶<https://www.w3.org/2005/Incubator/webid/spec/identity/>

⁷<https://solidproject.org/TR/oidc>

⁸<https://w3id.org/oac>

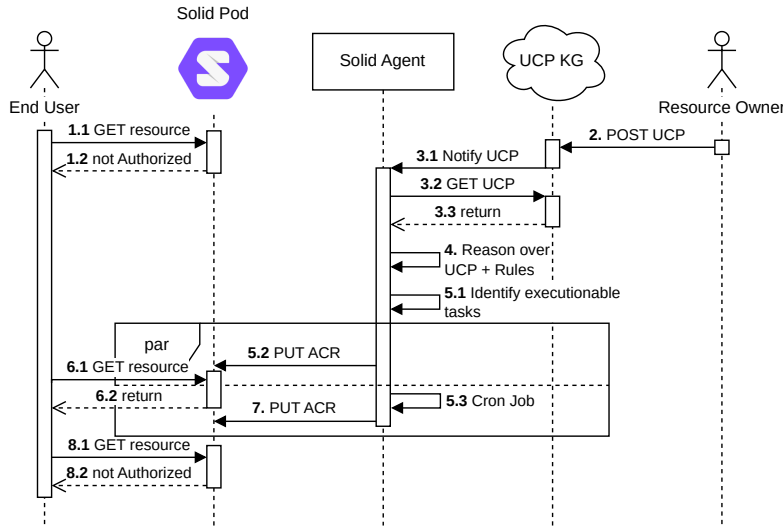


Figure 1: Enforcement flow showing (i) Bob not having access to X (1), (ii) Alice updating the UCP KG with a duration-restricted policy (2), (iii) the agent listening and fetching the policy (3), planning its tasks through reasoning (4) and executing them (5,7), and (iv) Bob having access for 30 seconds (6,8).

(ODRL) [1] and using the Data Privacy Vocabulary (DPV)⁹ [8] for representing data processing aspects. Usage control policies can model purpose and temporal constraints, where the latter could address the retention requirement. To the best of our knowledge, however, there is to this day no enforcement of ODRL policies and, by extension, no AOC enforcement mechanism. In general, Havur et al. [9] state that the enforcement of usage control in a decentralized setting is an open research challenge. Akaichi et al. contributed to this research problem by introducing a conceptual usage control framework to achieve continuous enforcement of usage control in a decentralized ecosystem [10].

An alternative approach for enforcing access control has been introduced by Grau et al. [11] through a demonstration of agent-based access control enforcement. However, this approach is limited by its reliance on ad-hoc rules tailored to their specific use case. Furthermore, it does not accommodate additional conditions such as the temporal domain.

3. Demo

To demonstrate the enforcement of a temporal ODRL policy, we elaborate the use case of a Solid pod owner, Alice, who wants to share resource X for a limited amount of time with another actor Bob. It is assumed she has a policy Knowledge Graph (KG) to which her Web agent is subscribed and that the Web agent has `ac1:Control` permission to the resources in Alice her pod.

To start sharing resource X, Alice adds the data usage ODRL policy shown in Listing 1 to the

⁹<https://w3id.org/dpv/>

KG: **Alice** gives **read** access to **Bob** for the duration of 30 seconds. Next, the agent is notified that the policy has been added to the Usage Control Policy (UCP) KG and the agent fetches this policy. The duration policy is passed to a reasoner, which results in an action plan with two concrete actions to be performed: (i) The authorization resource of X must be updated such that Bob has read access and (ii) In 30 seconds, the authorization resource of X must be updated such that Bob has no longer access. The agent performs (i) and starts a CronJob for 30 seconds to remove Bob's access to X. Finally, after 30 seconds have passed, the agent sends another request to the authorization resource of X such that Bob has no more access. A screencast in the open-source Solid Agent GitHub repository demonstrates the enforcement flow¹⁰ described at the beginning of this section and illustrated in 1.

```
ex:policy a odrl:Offer;
  odrl:permission ex:temporalPermission.

ex:temporalPermission a odrl:Permission ;
  odrl:action odrl:read ;
  odrl:constraint ex:durationConstraint ;
  odrl:target ex:resourceX ;
  odrl:assignee ex:Bob .

ex:durationConstraint
  odrl:leftOperand odrl:elapsedTime ;
  odrl:operator odrl:eq ;
  odrl:rightOperand "PT30S"^^xsd:duration .
```

Listing 1: A duration-restricted ODRL policy issued by Alice stating that Bob has read access to resource X for a duration of 30 seconds.

3.1. Materialization of the ODRL policy by the Solid Agent

The Solid Agent operates in the environment of the UCP KG and the Solid pod of Alice. Where in the former, the agent listens to changes by being notified by the KG, it actuates in the latter through HTTPS requests to Access Control Resources.

The actuations are driven by the input through condition-action rules. In the Solid Agent, these are implemented through Koreografeye [12], which separates the planning from the execution. In Koreografeye, reasoning over Notation3¹¹ (N3) rules is performed to generate a plan consisting of multiple tasks. In a second phase, these tasks are then executed through Koreografeye plugins¹² that execute Typescript¹³ code. Since Koreografeye is domain-agnostic, it neither understands ODRL nor Solid, dedicated N3 rules and plugins are written to materialize a temporal ODRL policy to concrete Access Control Resource at the pod level. For brevity and due to the fact everything is open-source¹⁴, we chose to use a shorthand notation to make clear

¹⁰<https://github.com/SolidLabResearch/Solid-Agent/tree/feat/sosy/documentation/sosy>

¹¹<https://w3c.github.io/N3/spec/>

¹²<https://github.com/eyereasoner/Koreografeye-Plugin>

¹³<https://www.typescriptlang.org/>

¹⁴<https://github.com/SolidLabResearch/Solid-Agent/tree/feat/sosy/>

how the materialization process works and why rather than explain this via code, RDF and Notation3 rules.

- Deontic concept *D*: It is only possible for this to be instantiated as Permission or Prohibition since there is no concept of Duty or Dispensation in ACP. In ODRL this is represented by the class of a rule.
- Subject *S*: The WebID of the party that is subject to the rule.
- Action *A*: The allowed action of a rule that a subject can execute on a resource.
- Resource *R*: The target resource to which the subject of a rule is allowed to perform a given action.
- Duration-restricted constraint *p*: The duration of how long the rule is active. This period is represented using an RDF literal of type `xsd:duration`.
- Time *t*: A representation of time of type `xsd:dateTime`.
- A Koreografeye action *Action*: A Koreografeye Plugin¹² that performs a dedicated execution task.

Using the shorthand, a general duration-restricted ODRL Rule is $D(S, A, R, p)$ and `Permission(ex:Bob, odr1:read, ex:ResourceX, "PT30S"^^xsd:duration)` is the instantiated example at Listing 1. The two actions the agent can perform are *ChangeAuthorization*($D(S, A, R)$) and *CronJob*($t, Action$). The former changes the authorization resource of *R* using the deontic concept, subject and action. The latter executes another action at time *t*.

On a policy update at time t_1 , the agent thus performs the condition-action rule:

```

IF
    Permission(S, A, R, p)
     $t_1$ 
THEN
    ChangeAuthorization(Permission(S, A, R))
    CronJob( $t_1 + p$ , ChangeAuthorization(Prohibition(S, A, R)))

```

Instantiated with Listing 1 at time `"2024-06-05T12:00:00"^^xsd:dateTime`, two actions need to be executed:

- ChangeAuthorization*(`Permission(ex:Bob, acl:read, ex:resourceX)`), and
- CronJob*(`"2024-06-05T12:00:30"^^xsd:dateTime, ChangeAuthorization(Prohibition(ex:Bob, acl:read, ex:resourceX))`)¹⁵. This results in *ChangeAuthorization* immediately adding Listing 2 to the Access Control Resource accompanying resource X and after 30 seconds removing Listing 2 again.

```

<#acr> a acp:AccessControlResource ;
  acp:resource ex:resourceX ;
  acp:accessControl <#bobReadAccess> .

<#bobReadAccess> a acp:AccessControl ;

```

¹⁵The condition-action rule also maps ODRL Actions to ACP Granted Access Modes.

```

acp:apply [
  a acp:Policy ;
  acp:allow acl:Read ;
  acp:anyOf [
    a acp:Matcher ;
    acp:agent ex:Bob
  ]
].

```

Listing 2: The materialization as an Access Control Resource as defined by ACP of the duration ODRL Policy stating Bob has read access to resource X.

4. Discussion

The demonstration shows an agent-based method of enforcing temporal usage control policies on Solid pods using ODRL without altering the Solid Protocol. This is achieved through materializing the ODRL to Solid authorization resources and demonstrated with ACP as it would also allow restricting the client and issuer of the assignee. For this, OAC has to be used as policy language as core ODRL does not define solid client and issuer restrictions.

However, the Solid Agent solution for enforcing UCP policies comes with some limitations. A first limitation is the necessity that the agent has complete control over all the resources owned by the resource owner in order to alter the authorization resources. Resource owners must trust that the agent will not misuse its control authority. However, there are no guarantees that the agent would never modify authorization resources without the consent of the owner and refrain from accessing the contents of the resources themselves. Certification of agents could potentially resolve this issue. The UCP KG introduces a challenge where the agent executes all newly added policies directly, even those that conflict with existing policies, causing inconsistencies in authorization resources. To resolve this, approaches such as those suggested in GUCON [13] are crucial to prohibit conflicting policies from entering the UCP KG.

Finally, full core ODRL will never be possible to be supported using our presented enforcement mechanism. There are several ODRL constraint left operands such as cardinality (`odr1:count`) or purpose (`odr1:purpose`) that cannot be expressed in Solid. As a result, next to materialization, additional approaches need to be explored to support full ODRL enforcement over Solid resources.

5. Conclusion

In this paper, we demonstrate how temporal usage control policies can be enforced in the Solid ecosystem by employing ODRL and a Web agent. Adhering strictly to the Solid protocol reveals that only a limited amount of usage control policies can be enforced. Enforcing a larger set of policies will require changes to the way authorization is granted within Solid. To enforce ODRL in Solid, we decided how to interpret ODRL to materialize this in access control resources due to a lack of ODRL formalization. However, such formalization is essential for verifying Web

agents and as a consequence ensuring trust. Future work may include negotiation over usage control policies with Web agents such that other actors can initiate asking permission.

Acknowledgments

Supported by SolidLab Vlaanderen (Flemish Government, EWI and RRF project VV023/10) and SYTADEL (SYnchromodal proTotype for Data Sharing and PLanning), c-SBO project(VIL, imec, UAntwerpen and Vlerick), funded by VLAIO. The authors would like to thank Ruben Dedecker, Patrick Hochstenbach and Jos De Roo for giving their insights and feedback regarding this work.

References

- [1] W3C Working Group, The open digital rights language (odrl), <https://www.w3.org/TR/odrl-model/>, 2018.
- [2] W. Slabbinck, R. Dedecker, J. A. Rojas Meléndez, R. Verborgh, A rule-based software agent on top of personal data stores, in: Proceedings of the 22nd International Semantic Web Conference: Posters, Demos, and Industry Tracks, 2023.
- [3] A. V. Sambra, E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulmaga, T. Berners-Lee, Solid: A Platform for Decentralized Social Applications Based on Linked Data (2016) 16.
- [4] R. Dedecker, W. Slabbinck, J. Wright, P. Hochstenbach, P. Colpaert, R. Verborgh, What's in a Pod? – A knowledge graph interpretation for the Solid ecosystem, in: M. Saleem, A.-C. Ngonga Ngomo (Eds.), Proceedings of the 6th Workshop on Storing, Querying and Benchmarking Knowledge Graphs, volume 3279 of *CEUR Workshop Proceedings*, 2022, pp. 81–96. URL: <https://solidlabresearch.github.io/WhatsInAPod/>, iSSN: 1613-0073.
- [5] S. Speicher, J. Arwe, A. Malhotra, Linked Data Platform 1.0, 2015. URL: <https://www.w3.org/TR/ldp/>.
- [6] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation), 2018. URL: <https://eur-lex.europa.eu/eli/reg/2016/679/oj>.
- [7] B. Esteves, V. Rodríguez-Doncel, H. J. Pandit, N. Mondada, P. McBennett, Using the ODRL profile for access control for solid pod resource governance, in: European Semantic Web Conference, Springer, 2022, pp. 16–20.
- [8] H. J. Pandit, B. Esteves, G. P. Krog, P. Ryan, D. Golpayegani, J. Flake, Data Privacy Vocabulary (DPV) – Version 2, 2024. URL: <http://arxiv.org/abs/2404.13426>. doi:10.48550/arXiv.2404.13426, arXiv:2404.13426 [cs].
- [9] G. Havur, M. Vander Sande, S. Kirrane, Greater control and transparency in personal data processing, in: Proceedings of the 6th International Conference on Information Systems Security and Privacy, SciTePress, 2020, pp. 655–662.
- [10] 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), volume 3705 of *CEUR Workshop Proceedings*, CEUR, Hersonissos, Greece, 2024. URL: <https://ceur-ws.org/Vol-3705/#paper10>, iISSN: 1613-0073.
- [11] J. Grau, S. Mayer, J. Strecker, K. Garcia, K. Bektas, Gaze-based Opportunistic Privacy-preserving Human-Agent Collaboration, in: Extended Abstracts of the 2024 CHI Conference on Human Factors in Computing Systems, CHI EA '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 1–6. URL: <https://doi.org/10.1145/3613905.3651066>. doi:10.1145/3613905.3651066.
 - [12] P. Hochstenbach, R. Verborgh, H. V. d. Sompel, Using Event Notifications, Solid and Orchestration for Decentralizing and Decoupling Scholarly Communication, *The Code4Lib Journal* (2023). URL: <https://journal.code4lib.org/articles/17823>.
 - [13] I. Akaichi, G. Flouris, I. Fundulaki, S. Kirrane, GUCON: A Generic Graph Pattern Based Policy Framework for Usage Control Enforcement, in: A. Fensel, A. Ozaki, D. Roman, A. Soylu (Eds.), *Rules and Reasoning*, volume 14244, Springer Nature Switzerland, Cham, 2023, pp. 34–53. URL: https://link.springer.com/10.1007/978-3-031-45072-3_3. doi:10.1007/978-3-031-45072-3_3, series Title: Lecture Notes in Computer Science.