

Distributed Social Benefit Allocation using Reasoning over Personal Data in Solid

Jonni Hanski¹, Pieter Heyvaert¹, Ben De Meester¹, Ruben Taelman¹ and Ruben Verborgh¹

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

Abstract

When interacting with government institutions, citizens may often be asked to provide a number of documents to various officials, due to the way the data is being processed by the government, and regulation or guidelines that restrict sharing of that data between institutions. Occasionally, documents from third parties, such as the private sector, are involved, as the data, rules, regulations and individual private data may be controlled by different parties. Facilitating efficient flow of information in such cases is therefore important, while still respecting the ownership and privacy of that data. Addressing these types of use cases in data storage and sharing, the Solid initiative allows individuals, organisations and the public sector to store their data in personal online datastores. Solid has been previously applied in data storage within government contexts, so we decided to extend that work by adding data processing services on top of such data and including multiple parties such as citizen and the private sector. However, introducing multiple parties within the data processing flow may impose new challenges, and implementing such data processing services in practice on top of Solid might present opportunities for improvement from the perspective of the implementer of the services. Within this work, together with the City of Antwerp in Belgium, we have produced a proof-of-concept service implementation operating at the described intersection of public sector, citizens and private sector, to manage social benefit allocation in a distributed environment. The service operates on distributed Linked Data stored in multiple Solid pods in RDF, using Notation3 rules to process that data and SPARQL queries to access and modify it. This way, our implementation seeks to respect the design principles of Solid, while taking advantage of the related technologies for representing, processing and modifying Linked Data. This document will describe our chosen use case, service design and implementation, and our observations resulting from this experiment. Through the proof-of-concept implementation, we have established a preliminary understanding of the current challenges in implementing such a service using the chosen technologies. We have identified topics such as verification of data that should be addressed when using such an approach in practice, assumptions related to data locations and tight coupling between our logic between the rules and program code. Addressing these topics in future work should help further the adoption of Linked Data as a means to solve challenges around data sharing, processing and ownership such as with government processes involving multiple parties.

Keywords

Solid, Linked Data,


1st International Workshop on Data Management for Knowledge Graphs (DMKG 2023) co-located with ESWC 2023, May 29th, 2023, Hersonissos, Greece

✉ jonni.hanski@ugent.be (J. Hanski); pieter.heyvaert@ugent.be (P. Heyvaert); ben.demeester@ugent.be (B. De Meester); ruben.taelman@ugent.be (R. Taelman); ruben.verborgh@ugent.be (R. Verborgh)

🌐 <https://pieterheyvaert.com/> (P. Heyvaert); <https://ben.de-meester.org/> (B. De Meester); <https://www.rubensworks.net/> (R. Taelman); <https://ruben.verborgh.org/> (R. Verborgh)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

1. Introduction

When the government decides to offer social benefits to its citizens, it will need some information about those citizens to properly allocate the benefits. Furthermore, when the social benefit concerns services offered by the private sector, such as benefits that are aimed to help citizens with electricity, gas or water invoices, the private sector may need to be involved in the process to some extent. To efficiently manage the allocation of benefits, the necessary data and other information needs to be accessible without unnecessary bottlenecks, such as manual labour in sending and receiving the necessary information as traditional documents multiple times over.

Together with the city of Antwerp in Belgium, we have produced an example use case in social benefit allocation, where the government offers eligible citizens a subsidy on utility invoices, in the form of *social tariff*, by paying part of the eligible citizens' utility invoices to private sector service providers. To facilitate this social tariff use case, the government has provided a set of rules that determine a citizen's eligibility to this benefit. The citizen has personal information associated with them that is evaluated against the rules to determine eligibility. The private sector utility company possesses the information needed to produce invoices for the citizen.

Decentralisation initiatives such as Solid [1] seek to enable data storage and control via personal online datastores (pods), without tying pods strictly for personal use by individuals, making them a generic means of storing data also in organisational contexts. Previous work [2] has demonstrated the benefits of applying the Solid initiative in government contexts to improve the handling of data. We decided to apply Solid in our social tariff use case, and have thereby decided to extend the previous work by introducing the private sector into the data processing scenario and building services to process the data between the different parties.

Following this introduction, section 2 will illustrate our use case and the service design, section 3 will introduce the implementation decisions, and section 4 outlines the observations and future work discovered during the experiment, with conclusions in section 5.

2. Social Tariff Use Case

Preconditions: The government is offering a subsidy on utility invoices to citizens who meet a certain criteria. The subsidy is realised as a social tariff on invoices from utility company, where the government pays part of the invoice of citizens eligible for that subsidy. To facilitate such social tariff in practice, the government has defined a set of rules determining the eligibility of a citizen based on data about that citizen. The utility company has the invoice for citizen together with a set of rules to allocate that invoice between the citizen and the government based on the eligibility of citizen. Within this use case, the government, citizen and utility company have all chosen to use Solid pods to store their respective data and rules, and share that data with each other as deemed appropriate. The tasks of determining social tariff eligibility and allocating the invoice appropriately have been assigned to dedicated services. All five entities – the government, citizen, utility company and the two services – are identified by their own WebIDs [3], and the parties involved have assigned the appropriate permissions to the data in their pods. All use cases and services use a reasoner to process their data based on the rules provided. The use case has been illustrated in 2, and we have identified the following four use

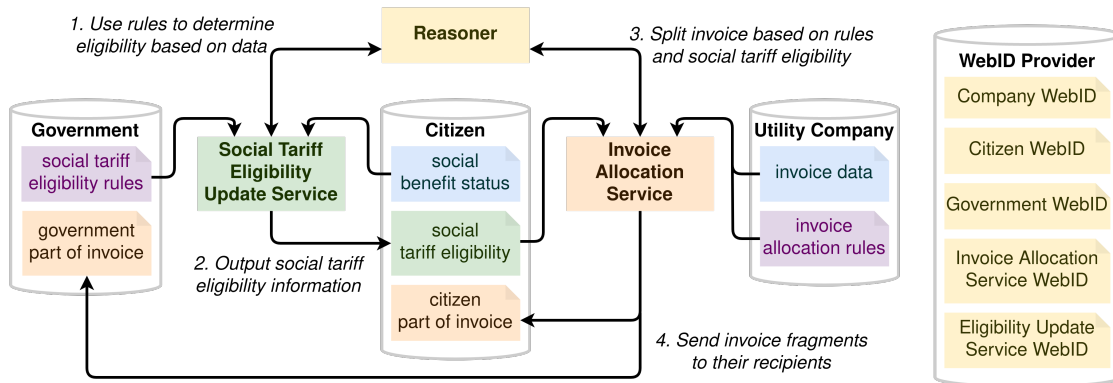


Figure 1: Illustration of the Solid pods of entities involved, data distribution between them, the services involved in manipulating that data and the WebIDs.

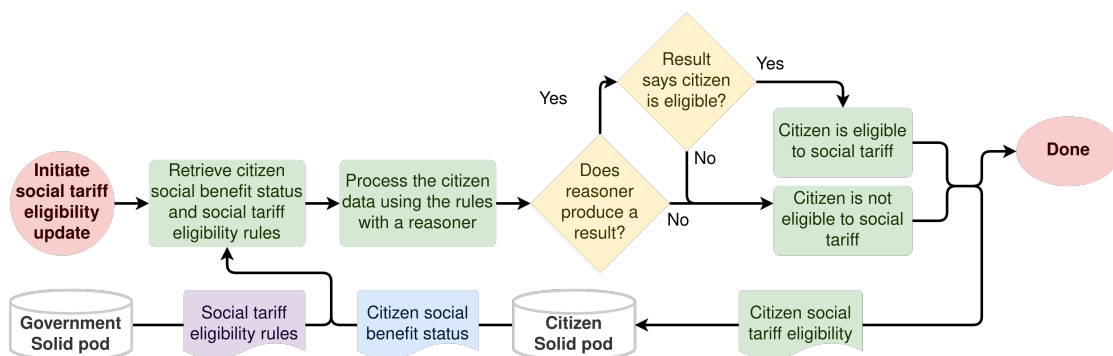


Figure 2: Illustration of the updating of social tariff eligibility for a citizen by the social tariff eligibility update service.

cases, of which 1A and 1B have been illustrated in 2:

Use case 1B: The data associated with the citizen makes them eligible for social tariff according to the government rules. The eligibility update service uses the rules to process the data and saves the status as eligible into the pod of the citizen.

Use case 1B: The data associated with the citizen does not make them eligible for social tariff according to the government rules *or* the data is not accessible or is unreadable, causing the system to fall back to not eligible. The eligibility update service saves the status as not eligible into the pod of the citizen.

Use case 2A: The user has been marked as eligible for social tariff. The invoice allocator service reads this data, reads the invoice data from the company, processes the invoice using the invoice splitting rules from the company, and then sends the invoice parts to the government and citizen accordingly.

Use case 2B: The user has been marked as not eligible for social tariff. The invoice allocator service reads this data, reads the invoice data and splitting rules from the company, processes the invoice using those rules and just sends the invoice to citizen without splitting it.

3. Prototype Implementation

The proof-of-concept implementation was built using existing established open-source software and libraries to the greatest extent possible at the time of implementation. Reducing the amount of custom code to the minimum was deemed important for appropriate evaluation of existing software available for implementing such a system. The system consists of three parts: Solid pods, reasoner and services. The technology choices are outlined below, and the code has been made available online¹.

The Solid pods were set up using the Community Solid Server [4], a modular open-source Solid server implementation. Users were identified with their WebID [3] created for the purposes of the proof-of-concept and data access was controlled through Web Access Control (WAC) [5, 6]. The demo data was RDF in Turtle serialization with example in listing 1.

```
@prefix social: <https://example.org/social-benefit-vocabulary#>.

<https://example.org/people/citizen#me> social:
  getsIncomeGuaranteeForElderly false.
```

Listing 1: Sample of the social benefit status for a citizen, to be used with the rules.

The rules from the government and the utility company were expressed through Notation3 [7] logic with example in listing 2. The demo data and rules were stored in the Solid pods, highlighted in blue and purple respectively, in figure 2 earlier.

```
@prefix social: <https://example.org/social-benefit-vocabulary#>.

{ ?person social:getsIncomeGuaranteeForElderly true. }
=>
{ ?person social:isSocialTariffEligible true. }.
```

Listing 2: Sample of the social tariff eligibility rules used.

The services were implemented in TypeScript. Data access was abstracted through the use of Comunica [8], an open-source SPARQL query engine that supports link traversal query processing within the context of Solid [9]. Example of data access can be found in listing 3.

```
PREFIX social: <https://example.org/social-benefit-vocabulary#>

SELECT ?id ?eligible WHERE {
  ?id social:isSocialTariffEligible ?eligible
}
```

Listing 3: Sample SPARQL query used to abstract data access in Solid pods, taken from the prototype implementation.

¹<https://github.com/SolidLabResearch/a-solid-proof-of-concept>

Authentication was implemented using an existing library [10]. The open-source EYE reasoner [11] was used to process data based on rules in Notation3 format. While the use of Notation3 rules and the reasoner was not necessary for this trivial use case, we chose this method with a general-purpose approach in mind, where both the data and logic could be expressed in RDF. We believe the potential decoupling of application logic from specific implementations in program code is worth exploring.

The observations from this proof-of-concept implementation are discussed in section 4. While the specific use case here tackled a social tariff problem, the concept of building services running on Solid and Linked Data could be generalised to other government services such as healthcare or education eligibility, or be applied in business-to-business or other scenarios.

4. Discussion

The main purpose of this proof-of-concept prototype was to demonstrate the feasibility of implementing such a system between multiple parties to serve a real-life use case that takes advantage of the decentralised aspects of Solid to ensure all parties maintain full control of their own data, as well as identify potential shortcomings that could impact the implementation of other similar systems in practice. We have identified a number of shortcomings with our approach, outlined below, that when addressed, should help the implementation of Solid and RDF-based services in practice.

- **Verification of information:** The prototype operates on data spread across pods and assumes it be accurate and up-to-date. The current approach allows citizens to make themselves eligible for benefits if they know which data to modify in their pods, regardless of whether they would be eligible in practice. This issue could be addressed through a number of approaches, for example by digitally signing the relevant parts of the RDF graph stored in the pod [12], or associating meta knowledge with RDF statements [13] such as credibility or provenance, together with Verifiable Credentials Data Model [14].
- **Hardcoded data location assumptions:** The locations of data written into Solid pods was hardcoded for the prototype. In practice [15], there needs to be a dynamic means of detecting the location of such data, especially when no previous data of the kind exists.
- **Use case-specific implementation of orchestration logic:** All use cases performed similar tasks by reading data, reading rules, sending them to a reasoner, and writing the output somewhere, with potential splitting or other processing of that data. This resulted in two service implementations for performing similar tasks. Therefore, a reusable general-purpose orchestrator to handle data processing workflows within the context of Solid, Linked Data and Notation3-based services should be of interest, where only the truly unique parts would require custom code. Such an orchestrator could allow for simple workflow definitions of input/output and the use of custom filters or other custom components as needed.
- **Tight coupling between rules, data and service code:** Despite the service code, data and rules being separated, the service code was dependent on the output defined by the rules, and the output was dependent on the data processed by those rules. Different

types of logic output had to be parsed in the service code, and processed accordingly. For example, after splitting the invoice using the rules, the service code was responsible for parsing that split data and saving it in the correct locations. Therefore, we identify the true decoupling of data, rules and code in actual service implementations to be of interest.

- **Spreading of implementation between Notation3 logic and program code:** While the logic to process data was implemented in Notation3, the program code responsible for authenticating for access control purposes, fetching the data, sending it to the reasoner with the rules, and then parsing the output and performing the SPARQL queries was not written in Notation3. This resulted in a mix of languages and a tight coupling between the logic and the program code. Therefore, facilitating the implementation of the entire service logic in Notation3 or other form should be of interest.
- **Scalability, performance, adaptation and complexity concerns:** As noted by a reviewer, we have not explored non-trivial use cases that would better represent real world scenarios. For example, scalability or performance issues could emerge when social benefits need to be applied to a subset of the population, and this subset needs to be calculated based on the entire population. Furthermore, application of our approach to additional subsets such as households also remains an open question, where benefits would be allocated on a household level rather than for individual citizens directly.

Though our schedule prevented the investigation of solutions to these challenges, we believe that addressing combinations of them will provide an interesting opportunity for future work. We acknowledge the shortcomings of this paper with regards to insufficient efforts towards developing solutions to these challenges and the associated reduction in the significance of this paper's contributions and the completeness of the work.

5. Conclusions

We conclude that addressing the shortcomings in section 4 through future work should help the adoption of Linked Data in practice, to solve real world challenges such as with GDPR but also general efficiency of passing data around within government and non-government processes. Services that need to process data should be possible to build on Linked Data and initiatives such as Solid, without the developers of those services having to make unnecessary assumptions or workarounds. Some of this work has been outlined previously in work on refining the definitions of a Solid pod [15], but other points such as verification of data, orchestration frameworks with reasoning and authentication, or scalability and adaptation to large-scale use over multiple overlapping sets of citizens appear in need of investigation.

Acknowledgments

The described research activities were supported by SolidLab Vlaanderen (Flemish Government, EWI and RRF project VV023/10). The use case was created in partnership with, and co-funded by, the City of Antwerp. Ruben Taelman is a postdoctoral fellow of the Research Foundation – Flanders (FWO) (1274521N).

References

- [1] R. Verborgh, Re-decentralizing the Web, for good this time, in: O. Seneviratne, J. Hendler (Eds.), *Linking the World's Information: A Collection of Essays on the Work of Sir Tim Berners-Lee*, ACM, 2022. URL: <https://ruben.verborgh.org/articles/redecentralizing-the-web/>.
- [2] R. Buyle, R. Taelman, K. Mostaert, G. Joris, E. Mannens, R. Verborgh, T. Berners-Lee, Streamlining governmental processes by putting citizens in control of their personal data (2019). URL: https://drive.verborgh.org/publications/buyle_egose_2019.pdf.
- [3] V. Balseiro, T. Turdean, J. Zucker, S. Capadisli, T. Berners-Lee, Solid webid profile, 2022. URL: <https://solid.github.io/webid-profile/>.
- [4] J. Van Herwegen, R. Verborgh, Communitysolidserver/communitysolidserver: v5.1.0, 2022. URL: <https://doi.org/10.5281/zenodo.7595117>. doi:10.5281/zenodo.7595117.
- [5] J. Hollenbach, J. Presbrey, T. Berners-Lee, Using rdf metadata to enable access control on the social semantic web, in: *Proceedings of the Workshop on Collaborative Construction, Management and Linking of Structured Knowledge (CK2009)*, volume 514, 2009, p. 167.
- [6] S. Capadisli, T. Berners-Lee, Web access control, 2022. URL: <https://solidproject.org/TR/wac>.
- [7] W. Van Woensel, D. Arndt, D. Tomaszuk, G. Kellogg, Notation3, W3C Community Group Draft Report, W3C, 2023. URL: <https://www.w3.org/N3/>.
- [8] R. Taelman, J. Van Herwegen, M. Vander Sande, R. Verborgh, Comunica: a modular sparql query engine for the web, in: *ISWC 2018, Proceedings*, 2018. URL: <https://comunica.github.io/Article-ISWC2018-Resource/>.
- [9] R. Taelman, R. Verborgh, Evaluation of link traversal query execution over decentralized environments with structural assumptions, 2023. URL: <https://arxiv.org/abs/2302.06933>. doi:10.48550/ARXIV.2302.06933.
- [10] Inrupt, Inc., Solid javascript authentication - solid-client-authn, 2023. URL: <https://github.com/inrupt/solid-client-authn-js>.
- [11] J. De Roo, Eye - euler yet another proof engine, 2023. URL: <https://github.com/eyereasoner/eye>.
- [12] G. Tummarello, C. Morbidoni, P. Puliti, F. Piazza, Signing individual fragments of an rdf graph, in: *Special interest tracks and posters of the 14th international conference on World Wide Web*, 2005, pp. 1020–1021.
- [13] R. Dividino, S. Sizov, S. Staab, B. Schueler, Querying for provenance, trust, uncertainty and other meta knowledge in rdf, *Journal of Web Semantics* 7 (2009) 204–219.
- [14] M. Sporny, G. Noble, D. Longley, D. C. Burnett, B. Zundel, K. Den Hartog, D. Chadwick, Verifiable Credentials Data Model v1.1, W3C Recommendation, W3C, 2022. URL: <https://www.w3.org/TR/vc-data-model/>.
- [15] 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, 2022.