Personal Data Ownership: An Agent-Based Approach

Erman Ayday[1], Youngjin Yoo[1], Mike Fisher[2], Ryan Tatton[1], and Benjamin Mucha[1]

[1] Case Western Reserve University, Cleveland OH 44106, USA
{exa208,yxy23,rdt46,bbm38}@cwru.edu
[2]

Abstract. Whitepaper summary goes here.

Keywords: Personal data · Self-sovereign identity

1 Introduction

2 Data Access Layer

2.1 Summary

The innermost layer of the Universal Data Agent (UDA) is the data access layer, which is where the ingestion of all personal data occurs. All data that passes through the UDA must come from an integrated "data source", which is an abstract term to describe any persistent or non-persistent supplier of data. Once data is retrieved from a data source, the UDA must perform some combination of either storing that data and/or pushing it to "subscribed" target agents. In other words, accessed data may be stored depending on the UDA's data source settings, and it also may be sent to target agents depending on the UDA's access control settings. Each data source can be integrated through its own "data driver", which is specialized code that actually translates access instructions from the UDA to the respective data source that it interacts with. [diagram here]

2.2 Data Sources and Data Drivers

We assume that data sources are not exclusively available to the UDA, such that other actors may actively change data which the

UDA accesses. Data sources can potentially manifest as "data stores", such as databases or file systems. A data source can even be another agent, through which data is transferred via DIDComm. A straightforward example of a data source is an external HTTP API, wherein the UDA can make HTTP requests such that data is received in respective response bodies. Of course, data sources are only meaningful as long as they can be reasonably integrated via a respective data driver. The responsibility of the data driver is to handle on-demand access requests from the UDA, and turn them into real requests that actually reach the data source. Once data is received from the data source, the data driver is then responsible for transforming that data such that it is reasonably structured for storage in the UDA.

2.3 UDA Data Storage

The UDA may store data at the discretion of the owning user in order to maintain the long term availability of such data. This is helpful as it guarantees that data targets may have access to a repository of accumulated long term data, rather than otherwise having to trust that each individual data source persistently and availably stores such data themselves. We believe that the most reasonable way to implement data storage in the UDA is to use a single database. While this of course incurs tight coupling between the queries that access data and the database itself, it also allows for more sophisticated and granular querying. Otherwise, we fear that allowing data storage on different types of repositories could very well result in compatibility issues, with advanced queries potentially not being supported on some implementations. Furthermore, since the UDA's stored data is only meaningful as a component of the UDA itself, and is not externally used, we therefore believe that there is no significant drawback to sticking with a single database. As for what that database is, we are currently leaning towards using MongoDB, where hereogenous data resources could be stored directly as JSON documents. This is especially convenient because JSON is an extremely common data format on the web, wherein the data from data sources may already be in JSON, allowing for simple translation and management within the UDA.

2.4 Access Queries

{I assume we should have a light abstraction layer for queries on stored data, just so that data targets dont have to issue actual direct database queries on literal tables/collections}

3 Access Control Layer

4 UI Interface

5 Conclusion

References