Towards using the Solid Protocol for Data Transport in International Data Spaces (IDS)

Christoph H.-J. Braun[1,*], Yanxiu Wuwang[1], Zhi Wang[1], Xuyang Hou[1] and Tobias Käfer[1]

[1]*Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany*

Abstract

With this paper, we contribute to the discussion of the conceptual relationship between the International Data Space (IDS) protocol and the Solid protocol. We propose that the Solid protocol can complement the IDS protocol as its data plane implementation. In an IDS' data transport process, the actual data is served from Solid Pods under access control. We highlight four different modes of operations for exchanging datasets between two Solid Pods using respective IDS Connectors. Data exchange can be performed (a) through the IDS connectors or directly at the Solid Pods; and (b) by either pulling the data from the data provider or pushing the data to the data consumer. We explore, which access control mechanisms may apply, depending on the mode of operation. We demonstrate practicality with a proof-of-concept implementation.

Keywords

International Data Spaces (IDS), Solid Protocol

1. Introduction

A universal protocol to facilitate data exchange between data providers and consumers – that is what the International Data Space (IDS) protocol [1] aims to become. To this end, the IDS protocol includes a bundle of protocols covering data catalogs, contract negotiation and data transport, which is mainly geared towards organizations. To break open organizational data silos and to give users back control over their data, the Solid protocol [2] consists of a bundle of specifications covering agent identification, authentication, authorization and data interaction. A conceptual combination of IDS and Solid, called *Solid Data Spaces (SDS)*, has been proposed [3]. While outlining how Solid could underlie a data space, they did not present an architecture design for how these two protocols could practically be combined.

The two protocols exhibit major design differences: Solid is designed for users to manage users' data in a decentralized manner [4, 5], while IDS is designed for organizations to perform data exchange between organizations [6]. Both Solid and IDS look at facilitating data sharing, but their protocols focus on different aspects of the overall data sharing process. Solid focuses on actual data transport ("over the wire"), whereas IDS focuses on the abstract *process* protocols without considering the *actual* data transfer. We thus ask:

Q1 How can the IDS protocol and the Solid protocol be combined to provide data transfer in an IDS?

With this work, we contribute to solving the generic challenge[1] of clarifying the conceptual relation between the IDS protocol and the Solid protocol. We propose that the Solid protocol may complement the IDS protocol as its data plane implementation. Actual data transfer is in-fact deemed out-of-scope by the IDS protocol [1]. We thus combine the IDS' data transport process protocol and the Solid protocol for the actual sharing of data. We demonstrate how a dataset can be exchanged between two Solid Pods, i. e. web servers that adheres to the Solid protocol, using two respective IDS Connectors, web

The Third International Workshop on Semantics in Dataspaces, co-located with the Extended Semantic Web Conference, June 01, 2025, Portorož, Slovenia

*Corresponding author.

braun@kit.edu (C. H.-J. Braun); yanxiu.wuwang@student.kit.edu (Y. Wuwang); zhi.wang@student.kit.edu (Z. Wang); xuyang.hou@student.kit.edu (X. Hou); tobias.kaefer@kit.edu (T. Käfer)

0000-0002-5843-0316 (C. H.-J. Braun); 0000-0003-1810-0395 (Y. Wuwang); 0009-0006-0455-9356 (Z. Wang); 0009-0004-8538-2999 (X. Hou); 0000-0003-0576-7457 (T. Käfer)

[1]https://github.com/w3c-cg/dataspaces/issues/9

servers that adheres to the IDS protocol. Specifically: The data consumer's IDS connector establishes a connection with a data provider's IDS connector to initiate a data transfer process. Access to the data on the Solid Pod may be executed through the provider's IDS connector or directly without this level of indirection. Higher level protocols like contract negotiation and data discovery in catalogs are out-of-scope for this paper as the Solid protocol does not directly provide its own solutions to these problems. However, the Solid protocol might provide the technological foundation to implement solutions for these problems as well.

The paper is structured as follows: In Section 2, we cover foundations and related work. In Section 3, we highlight the architectural design space that a combination of the two protocols offers. In Section 4, we present our proof-of-concept implementation; code[2] and a short video[3] are available online. In Section 5, we provide an outlook to future work on further investigating the relation between the IDS protocol and the Solid protocol.

2. Preliminaries and Related Work

This section provides an overview of the IDS protocol and the Solid protocol. It also presents related work on Solid Data Spaces (SDS) which inspired our work.

2.1. International Data Space (IDS) Protocol

The International Data Space (IDS) protocol [1] aims to standardize data exchange within trusted ecosystems. The primary goal of the IDS protocol is to facilitate interoperable data sharing in a distributed data exchange ecosystem: Every data controller is allowed to specify its own policies on data consumption while using a unified data transfer process. In an IDS, participants interact through certified components which aims to ensure data security and to foster trust within the data space.

Figure 1(a) illustrates core components of an IDS relevant to the data transfer we consider in this work. Additional IDS components e. g. from the IDS reference architecture model (RAM) [6], are not explicitly listed in Figure 1(a), including a meta-data broker, clearing house, app stores, apps and vocabularies. An IDS revolves around a central identity provider that verifies and certifies data space participants and their connectors. Figure 1(a) also shows three different IDS sub-protocols [1]: the catalog protocol, the contract negotiation protocol and the data transfer process protocol.

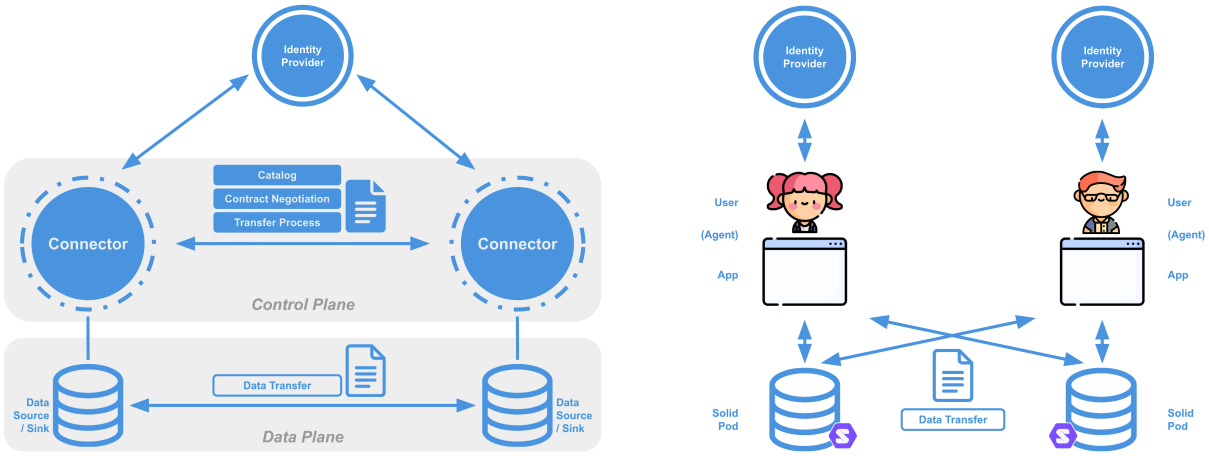
These three IDS (sub-)protocols belong to the *control plane*, a conceptual level on which messages are exchanged that control the data sharing *process*. The *actual* data transfer between a data source and data sink, deemed out-of-scope by the IDS protocol, belongs to the *data plane*.

The *IDS catalog protocol* specifies how a catalog is requested from a catalog service by a consumer using messages in an abstract message exchange format. Concrete message exchange wire formats may be defined in corresponding binding specifications.

The *IDS contract negotiation protocol* specifies a contract negotiation (CN) between a provider that offers one or more datasets under a usage contract to a consumer that requests datasets. A CN progresses through a series of states using messages between provider and consumer in an abstract message exchange format. Concrete message exchange wire formats may be defined in corresponding binding specifications.

The *IDS transfer process protocol* specifies a transfer process (TP) between a provider that offers one or more datasets under a usage policy and a consumer that requests datasets. A TP progresses through a series of states using messages between provider and consumer in an abstract message exchange format. Concrete message exchange formats may be defined in corresponding binding specifications for the transfer *process*.

[2] https://gitlab.kit.edu/uofvo/solid-ids-connector
[3] https://purl.archive.org/uvdsl/sds25



(a) An overview of the IDS protocol's core components and conceptual planes. Further IDS RAM components are omitted.

(b) An overview of the Solid protocol's conceptual components. There is no concept of "planes".

Figure 1: The conceptual components of the two protocols. The IDS protocol deems the data plane out-of-scope; the Solid protocol defines data access and transfer via a RESTful HTTP API of an access-controlled data storage.

The *actual* data transfer between data source and data sink is out-of-scope for the IDS transfer process protocol – only the abstract transfer *process* is being defined. For the actual data transfer, the transfer process protocol abstractly supports two modes of transfer process: *pull* transfers where the consumer retrieves data from the provider; and *push* transfers where the provider initiates data transmission to the consumer.

2.2. Solid Protocol

The Solid protocol [2] is a bundle of specifications defining the behavior of a RESTful [7] web server, the Solid Pod (Personal Online Datastore). To this end, the Solid protocol specifically covers agent identification, authentication, authorization, and data interaction. Agents can be persons, groups or organizations, or even software.

For agent identification, the Solid procotol relies on WebIDs [8]. A WebID is an HTTP URI that identifies an agent. For agent authentication, the Solid protocol relies on Solid-OIDC [9], a modified version of OpenID Connect [10]. For agent authorization, the Solid protocol relies on specifying access control rules, e. g. using Web Access Control (WAC) [11] or Access Control Policies (ACP) [12]. Finally for operations on web resources, the Solid protocol takes inspiration in the Linked Data Platform (LDP) [13]. A Solid Pod thus provides a standardized interface to store newly provided data, to access existing data for consumption, or to modify data at its source.

Figure 1(b) illustrates conceptual components in the Solid protocol: Users through applications, applications themselves, or – more general – agents consume data from and provide data to Solid Pods as their data storage. As the Solid protocol uses Solid-OIDC for single sign-on, there also exist (potentially multiple) identity providers (IDPs) in a Solid-based ecosystem. However, the Solid protocol does not provide a governance framework, certification procedure or negotiation protocol - it merely provides the technical means to build more complex data ecosystems.

2.3. Solid Data Spaces (SDS)

A Solid Data Space (SDS) [3] is a decentralized dataspace concept that builds on the Solid Protocol and Linked Data principles to enable self-sovereign data sharing. Compared to the IDS protocol that relies on certified components such as the central identity provider and connectors, the SDS approach aims to be lightweight, prioritizing ease of adoption and gradual integration into decentralized ecosystems.

The SDS approach avoids dependence on a central authority and emphasizes the flexibility offered by a decentralized approach.

Considering IDS connectors as a potential part of an SDS, [3] merely suggests that IDS connectors could be considered as Dataspace Service (DSS) apps for data exchange: An IDS connector uses a Solid Pod as its data source or sink (cf. Figure 1(a)). The idea is only outlined vaguely and remains a theoretical concept, thus providing a starting point for our work.

3. Architecture Design Space

The IDS protocol deliberately leaves the actual data transfer on the data plane out-of-scope to allow for flexibility in the underlying transfer technology. Looking at open standards that may fill this void, we apply the Solid protocol to create a unified data transfer protocol. Inspired by the SDS approach, we design an IDS connector that runs above a Solid Pod. The IDS connector exchanges process-related messages (on the control plane) as described in the IDS transfer protocol and accesses data from a Solid Pod (on the data plane). In this section, we present the architectural design space that such a combination of the IDS protocol and the Solid protocol provides.

The IDS protocol explicitly allows out-of-band data transfer[4] to offload technical limitations to the applied systems instead of their connectors: A direct connection between the Data Provider's system acting as a data source, and a system on the consumer-side acting as the data sink can be established. By allowing to switch over to an unspecified environment, the IDS protocol thus allows arbitrary messages to be exchanged after control plane messages. Among others, this raises the question of how access control is handled during the actual data transfer, e. g. a data consumer should not be able to access data stored in the provider's data source without authentication and authorization. We therefore investigate potential *modes of operations* (Section 3.1) for data transfer and which access control mechanisms could be applied (Section 3.2).

3.1. Modes of Operation

To realize actual data transfer, the IDS protocol allows for in-band or out-of-band communication [1]. Therefore, two potential designs exists: We call the in-band communication the *proxy mode*, where all messages pass through the IDS connectors and each IDS connector only communicates with its corresponding Solid Pod. We call the out-of-band-communication the *direct mode*, where messages from the control plane are exchanged between the connectors while messages from the data plane are exchanged directly with the data source and sink where possible.

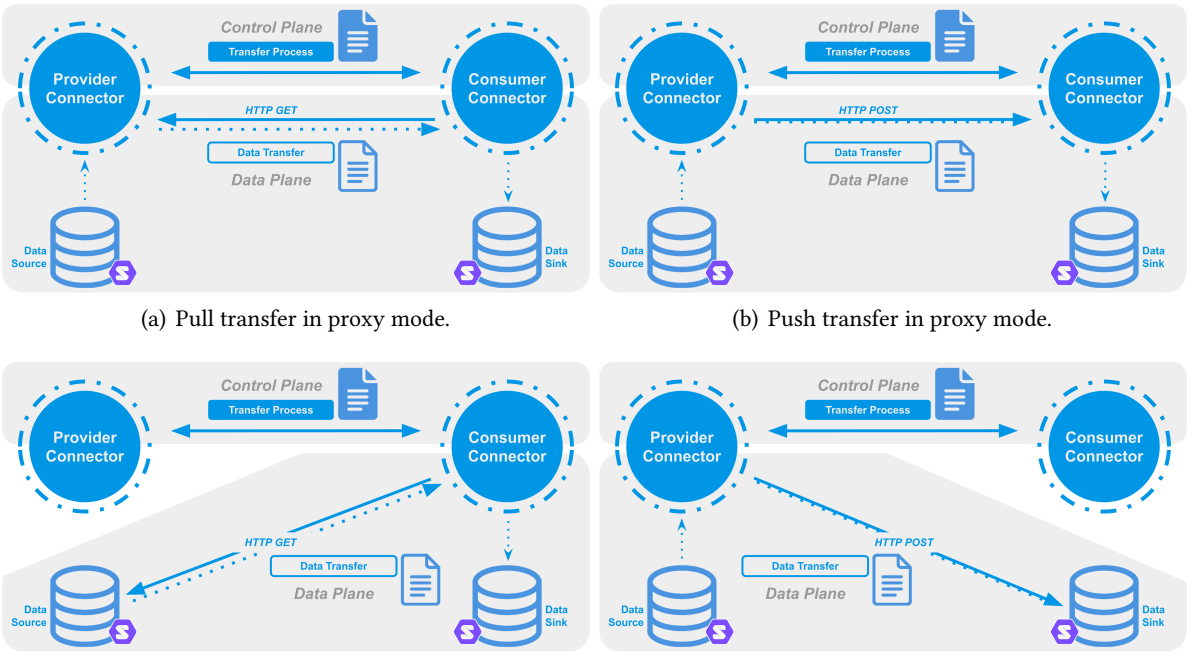
In that communication, the IDS data transfer protocol allows for pull and push transfer. Therefore, two potential flows exists considering that data source and data sink are Solid Pods: In the *pull transfer*, the consumer connector uses `HTTP GET` requests to actively obtain the data. In the *push transfer*, the provider connector uses `HTTP POST` requests to actively provide the data. Depending on the communication design, the `HTTP` request are send to the respective other IDS connector or to the corresponding Solid Pods directly.

Figure 2 summarizes the four modes of operations. Note that in our design of the direct mode, at least one connector is involved because access on a data source or sink is only granted IDS connectors as they are certified to be trustworthy. Separating an IDS connectors control and data plane components entirely is of course also possible.

3.2. Access Control

To secure the actual data transfer, authentication and authorization of consumer and provider are required. Depending on the mode of operation, how such access control is performed varies.

[4] https://docs.internationaldataspaces.org/ids-knowledgebase/ids-ram-4/layers-of-the-reference-architecture-model/3-layers-of-the-reference-architecture-model/3_4_process_layer/3_4_4_exchanging_data#data-transfer-via-another-infrastructure-or-protocol



(a) Pull transfer in proxy mode.

(b) Push transfer in proxy mode.

(c) Pull transfer in direct mode.

(d) Push transfer in direct mode.

Figure 2: The four modes of operation: *mode* ∈ { (*proxy*,*direct*) × (*push*,*pull*) }. Dotted lines indicate data flow. Solid lines indicate HTTP requests.

In *proxy mode*, authentication and authorization between the IDS connectors is covered by the IDS protocol. Authentication may be performed using e. g. the IDS connectors' certificates and dynamic attribute tokens. Authorization may be evaluated using e. g. a IDS connectors' policy engine based on the result of the contract negotiation (sub-)protocol.

In *direct mode*, the difference in authentication protocol between the IDS protocol and Solid protocol becomes an issue: They are not compatible out-of-the-box. However, because the IDS transfer process protocol allows for the out-of-band communication (which we use here), IDS authentication can simply be ignored at this point. Instead, the data provider and consumer follow the Solid protocol to access data on the Solid Pods; authentication via Solid-OIDC and authorization using WAC or ACP. In the messages on the control plane of the IDS data transfer process , the token fields remain therefore unused.

Such a design on its own is not against the IDS specification: Open data connectors, i. e. connectors without access control on the data they serve, can serve public data to any data consumer. And, what happens on the data plane afterwards is non of the IDS protocols concern.

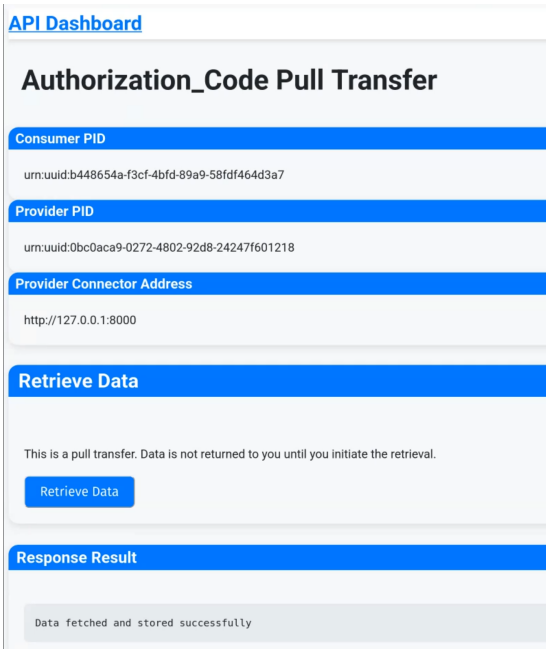
4. Proof-of-Concept

We implemented a proof-of-concept, available online[5]. To be able to properly demonstrate the four modes of operations to the workshop audience, we implemented a rough user interface (UI), see Figure 3. The UI resembles a user-facing website that allows users to store their own documents in Solid Pods and exchange data with others. To showcase, that we are really using Solid, we follow the Solid-OIDC Authentication Code grant [9], the standard flow for authenticating users, where the user actively logs in to their IDS connector with their WebID. The IDS connector then handles the data transfer process (`pull` or `push`) on the server-side. After the transfer process ended, the user is provided a summary of the IDS transfer process, generated from the exchanged messages.

Alternatively, and maybe more in-line with how interaction with an IDS connector is envisioned, a headless application on a server such as a crawler or an analysis software could also use our connector to

[5] https://gitlab.kit.edu/uofvo/solid-ids-connector

(a) User interface to initiate data transfer process.



(b) Summary of transfer process after completion.

Figure 3: Our proof-of-concept demonstration. See also our short video online (cf. footnote 7).

consume data from a Solid Pod. In such a scenario, the connector performs the OIDC Client Credentials grant[6]. A pre-requisite is that the IDS connector has access to a valid `client_id` and `client_secret` from the user's/organization's Solid identity provider.

The different showcases are stored in their corresponding branches in the repository, namely: In `authorization_code` and `client_credentials` flows using the *proxy mode* for both *pull* and *push*; and in `solid_mode` using the *direct mode* for pull transfer. The not yet implemented designs are current work, to be adapted from the other implementations. To provide an impression of our demonstrator, a short video is available online[7].

5. To the Data Plane, and Beyond!

In this paper, we investigated and demonstrated how the Solid protocol may complement the IDS protocol as its data plane implementation. As a by-product, we recognized the following:

(1) The two protocols are not substitutes.
The Solid protocol complements the IDS transfer process protocol by filling the out-of-scope void of the data plane. In *proxy mode*, there are no Solid-based interactions between components belonging to different organizations. Identity and access management remains completely managed by the IDS connectors of each organization. There is no *necessity* to further merge the two protocols. However, aligning the two protocols by joining forces may result in a fully specified and open-standards based protocol which may foster adoption.

(2) Alignment of authentication protocols seems possible.
For authentication, the Solid protocol relies on an extension of OpenID Connect which is based on OAuth 2.0. The IDS protocol relies, among certificates, also on dynamic attribute tokens obtained using an OAuth 2.0 flow. We are intrigued to investigate the potential union of the two protocols approaches to identity and authentication protocols such that one unified authentication protocol may emerge.

[6]Note that the Client Credentials grant is not required for Solid conformance, but the Community Solid Server (https://github.com/CommunitySolidServer/CommunitySolidServer) does support this grant type.
[7]https://purl.archive.org/uvdsl/sds25

(3) User rights management seems out-of-scope[8] to the IDS protocol.
The IDS protocol is designed towards organizations exchanging data with other organizations; each organization has one IDS connector. The Solid protocol employs a more user-centric perspective: In an organization, different (human) users or (software) agent perform actions. In a Solid Data Space, delegation of rights to these actions may be implemented using a proxy pattern [14]: HTTP requests of users are checked and potentially forwarded via an organization's (central or federated) *rights delegation proxy*. In our investigation, we noticed that an organization's IDS connector acts in a similar manner as the presented delegation proxy; a user from an organization must use its IDS connector to exchange data with other organizations (via their IDS connector respectively). The conceptual difference: The delegation proxy holds policies to assess if a organization internal user is allowed to execute a certain action towards an external dataspace participant. The IDS connector holds policies to assess if an external dataspace participant is allowed to execute a certain action on an organisation internal dataset. Therefore, we are intrigued to apply Solid-based rights delegation [14] to implement user rights management in an IDS connector.

(4) Clarifying relations with other emerging or mature technologies.
In our investigation, we noticed that the marketing arguments for either of the two protocols do not reflect their technical details accurately: The IDS protocol "is geared towards" organizational data exchange whereas Solid "is geared towards" personal data sharing. But both protocols work with any data; the Solid protocol can be used in and across organizations while the IDS protocol can be used for data discovery, contract negotiation and transfer process on a privately owned IDS connector. Only when looking at the technical details, we were able to determine what the actual conceptual relation between the two protocols is, where they complement each other and where they intersect. Thus we wonder: What other technologies, emerging or mature, pose as substitutes or complements to parts of the IDS protocol? What do they do differently and is their way better or worse than IDS' status quo?

We feel inspired and look forward to tackle future work.

Acknowledgments

This work was supported in part by the German ministry BMBF under grant 16DTM107B (*MANDAT*) and in part by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project number 459291153 – Research Unit 5339. Icons of user and app in Figure 1(b) were created by Freepik.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] IDSA, Dataspace Protocol 2024-1, Technical Report, International Data Space Association (IDSA), 2024. URL: https://docs.internationaldataspaces.org/ids-knowledgebase/dataspace-protocol, accessed 2025-03-13.
[2] W3C Solid Community Group, Solid Protocol, 2024. URL: https://solidproject.org/TR/protocol.
[3] S. Meckler, R. Dorsch, D. Henselmann, A. Harth, The web and linked data as a solid foundation for dataspaces, 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. 1440–1446. URL: https://doi.org/10.1145/3543873.3587616. doi:10.1145/3543873.3587616.

[8] https://github.com/International-Data-Spaces-Association/IDS-RAM_4_0/discussions/60#discussioncomment-1813033

[4] A. V. Sambra, A. Guy, S. Capadisli, N. Greco, Building decentralized applications for the social web, in: J. Bourdeau, J. Hendler, R. Nkambou, I. Horrocks, B. Y. Zhao (Eds.), Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume, ACM, 2016, pp. 1033–1034. URL: https://doi.org/10.1145/2872518.2891060. doi:10.1145/2872518.2891060.

[5] E. Mansour, A. V. Sambra, S. Hawke, M. Zereba, S. Capadisli, A. Ghanem, A. Aboulnaga, T. Berners-Lee, A demonstration of the solid platform for social web applications, in: J. Bourdeau, J. Hendler, R. Nkambou, I. Horrocks, B. Y. Zhao (Eds.), Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume, ACM, 2016, pp. 223–226. URL: https://doi.org/10.1145/2872518.2890529. doi:10.1145/2872518.2890529.

[6] International Data Space Association (IDSA), IDS Reference Architecture Modal (IDS-RAM), 2023. URL: https://docs.internationaldataspaces.org/ids-knowledgebase/ids-ram-4/introduction/1_1_goals_of_the_international_data_spaces.

[7] R. T. Fielding, Architectural Styles and the Design of Network-based Software Architectures, Ph.D. thesis, 2000.

[8] A. Sambra, H. Story, T. Berners-Lee, WebID 1.0 - Web Identity and Discovery, W3C Editor's Draft, W3C, 2014. URL: https://www.w3.org/2005/Incubator/webid/spec/identity/.

[9] A. Coburn, E. Pavlik, D. Zagidulin, Solid-OIDC, 2022. https://solidproject.org/TR/oidc.

[10] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, C. Mortimore, OpenID Connect Core 1.0, Final Specification, 2014. URL: https://openid.net/specs/openid-connect-core-1_0.html.

[11] S. Capadisli, Web Access Control, Editor's Draft, W3C Solid Community Group, 2022. URL: https://solid.github.io/web-access-control-spec/.

[12] M. Bosquet, Access Control Policy (ACP), Editor's Draft, W3C Solid Community Group, 2022. URL: https://solid.github.io/authorization-panel/acp-specification/.

[13] S. Speicher, J. Arwe, A. Malhotra, Linked Data Platform 1.0, W3C Recommendation, W3C, 2015. URL: https://www.w3.org/TR/ldp/.

[14] S. Schmid, D. Schraudner, A. Harth, The Rights Delegation Proxy: An Approach for Delegations in the Solid Dataspace, in: Proceedings of the Second International Workshop on Semantics in Dataspaces (SDS 2024) co-located with the 21st Extended Semantic Web Conference (ESWC 2024), 2024. URL: https://ceur-ws.org/Vol-3705/paper02.pdf.