End-user identity in Solid: *the interoperability problem space*

Author: Ruben Verborgh, SolidLab, Ghent University – imec
Date created: 2022-12-06
Last updated: 2023-01-10
Version: v1.0.0

The Solid ecosystem uses a decentralized mechanism of WebIDs to identify agents and to manage their access control. As the number of participants in the ecosystem increases, the question of how to manage a multitude and variety of WebIDs becomes increasingly pressing. To this end, we performed an assessment of the current state of end-user identity and the demands going forward. This document examines the interoperability angle for personal identity within Solid, providing strict technical as well as looser interpretations of the WebID concept, building upon these to outline the problem space as well as directions for solutions. We discuss the necessity of a shared understanding, and describe challenges including anonymity and pseudonymity, extending the identifier space, and disambiguating different WebIDs and identity providers pertaining to the same end-users. We thereby provide a blueprint for the work needed to mature the Solid ecosystem with regard to identity.

Current WebIDs and their interpretation

Definition and possible technical interpretations

The overall working definition states that a **WebID is an HTTP(S) URL uniquely identifying an agent**. We observe that the meaning of "identifying", however, depends on the specific context of its usage. Different interpretations consider other, sometimes simultaneous and/or conflicting, angles:

- From a ***text*** angle, because HTTP URLs are text strings, a WebID is a string as well. Therefore, in the most narrow interpretation, the WebID for a certain agent should only be interpreted as the series of characters, not as the agent it points to, nor as the document that possibly results from passing the URL to an HTTP client.
    - Example:
      `https://sophie.example/#me`
      could be a WebID, and it consists of 26 characters.
    - It is impossible to tell given only the text string whether or not a given URL is a WebID; that is, there are no structural constraints (even the hash # is optional).
    - We cannot derive from the form of the URI what agent it points to; only exact text string equality may be used.
- From a ***name*** angle, a WebID being a URI means that it can be used to uniquely point to a specific concept. Specifically, pointing to an agent (such as a person or piece of software) is what differentiates a WebID from other URLs. Herein, "unique" means that each WebID corresponds to only one agent; but each agent can and will have multiple WebIDs as names, i.e., there exists a one-to-many relationship between agents and WebIDs.
    - Example:
      `https://sophie.example/#me`
      `https://abc.example/Pages/Other`
      could be names for a specific real-world person.
    - The current practice is to use WebIDs for *agents*, i.e., things that have agency. However, they might also become used for more passive entities such as cars and buildings, which are operated by an agent, i.e., *agent-operated* entities.
    - The process by which we establish the connection between a WebID (text string) and a real-world entity is purposely not uniformly defined or standardized. This parallels similar questions on other parts of the Web, for instance, the extent to which a given domain name or social media handle can be trusted to be an official representation of a certain real-world entity.
    - Anyone can mint HTTP URLs and use them as (unapproved) WebIDs for *any* agent, without their knowledge or consent. In fact, there exist an infinite number of WebIDs as names for any agent, only a small fraction of which they might be aware of, and an even smaller fraction of which they might consider authoritative.

- From a ***data*** angle, a WebID being a URI means that we can employ it within RDF triples and documents to express information about a real-world person or a software agent. The WebID's *name* angle provides disambiguation with regard to the topic of the data. Note that this usage of a WebID is not necessarily authoritative; i.e., it does not express provenance or truth.

    - Example:
      `<https://sophie.example/#me> ex:hasBirthDate "1981-02-03".`
      `<https://sophie.example/#me> ex:hasBirthDate "1984-05-06".`
      `<https://doc.example/1> ex:hasAuthor <https://sophie.example/#me>.`
      The deliberate inconsistency in birthdays shows that usage in data by itself does not constitute an authoritative reference, nor sufficient evidence that the HTTP URL is in fact a WebID or that the agent pointed to from the data is aware of or has agreed to any statements mentioning this WebID.

- From a ***locator*** angle, a WebID being an HTTP URL means that it may (and in fact should) point to a *WebID profile document*, which an HTTP client might be able to retrieve over HTTP. This WebID profile document, given the Linked Data principles and the *data* angle of a WebID, is expected to contain RDF data involving and/or related to the agent identified by that WebID.

    - If an agent deems a particular WebID (URL) as authoritative for them, the expectation is that they have (presumably exclusive) write access to the corresponding WebID profile document that results from following this URL. However, there is no such guarantee.

    - Example: `https://sophie.example/people/sophie#me` might lead to a document `https://sophie.example/people/sophie` containing RDF triples, some of which have as a subject `https://sophie.example/people/sophie#me`.

    - Locators may irreparably break because of DNS reasons if the domain name ownership changes. Like any URL, a WebID is therefore a *leased* identifier for location purposes, given that relatively few people have or control their own domain name.

- From an ***authentication*** angle, a WebID can be an outcome of an authentication process. That is, an identity provider (IDP) generates a token by which an agent can support a claim towards a third-party service that they are the agent identified by the given WebID.

    - This mechanism requires establishing a certain degree of trust between the IDP and the service. A current mechanism involves the locator and data angles, whereby the WebID profile document resulting from following the WebID (URL) explicitly marks certain IDPs as trusted to make authentication claims about this particular WebID.

    - We also observe that people as human agents access Solid pods through applications, so in addition to authenticating the agent with a WebID, we also need to authenticate the application, which currently happens with a Client ID. Application identity is out of scope for this document.

We stress that these different angles for WebIDs are, in theory, largely independent. However, as some angles build upon others, we observe larger degrees of coupling in practice. Importantly, the existence

of these multiple angles means that any solutions for identity will probably need to consider or address combinations of multiple angles simultaneously.

Common interpretations today

Notwithstanding the above technical interpretation, current practitioners and systems will typically assume more meaning about a WebID than is strictly allowed. While such meaning might partially correspond to reality in specific situations, it cannot be generalized.

Let us take the WebID `https://sophie.solidcommunity.example/profile/card#me` as an example.

Technically permitted assumptions

- *Without any further knowledge about the above text string.*
  - ✅ It is an HTTP URL.
- *When the URL dereferences to a document containing RDF, making some statement that* `https://sophie.solidcommunity.example/profile/card#me` *is an agent.*
  - ✅It is a WebID.

Common assumptions (which might or might not be correct, so are not permitted in general)

- ❓It is a WebID.
- ❓It identifies a person.
- ❓It identifies Sophie.
- ❓It points to an RDF document.
- ❓Sophie is aware of this WebID.
- ❓Sophie is the one who created this WebID.
- ❓Sophie is the one who manages this WebID.
- ❓This is Sophie's WebID.
- ❓This is Sophie's only WebID.
- ❓This is Sophie's main WebID.
- ❓`solidcommunity.example` is an IDP for Sophie.
- ❓`solidcommunity.example` is the only IDP for Sophie.
- ❓`solidcommunity.example` is the main IDP for Sophie.
- ❓`https://sophie.solidcommunity.example/` is Sophie's storage.
- ❓`https://sophie.solidcommunity.example/` is Sophie's only storage.
- ❓`https://sophie.solidcommunity.example/` is Sophie's main storage.
- ❓Basic profile information (name etc.) will be included in the WebID profile document.
- ❓Sophie has read, write or control access to storages listed in the WebID profile document.
- ...

In particular, one common interpretation for a given WebID does not correspond to the notion of "this agent" (as the data angle would prescribe), but rather to the more specific "this agent identified in a given context" (currently not substantiated by any specification). For example, suppose we know a person named Ira who uses `https://ira.solidcommunity.example/profile/#me` as a WebID. Some parties consider an authentication with `https://ira.solidcommunity.example/profile/#me` to not indicate "Ira", but rather an ill-defined "solidcommunity.example version of Ira". These diverging interpretations highlight the current conflict between the data and authentication angles of WebIDs,

where the data angle would consider Ira a real-world person rather than a more abstract contextual identity. Furthermore, it confuses the trust in the name with the trust in a specific IDP this URL is assumed to relate to (namely solidcommunity.example), for which there is no strict technical basis.

Current usage

WebIDs for agents play a crucial role within Solid in different ways, making use of the aforementioned angles. We distinguish the following, wherein we note that this document focuses on WebIDs for end-users specifically.

- **WebIDs for end-users**
  - As part of **authentication**, WebIDs serve as an identifier for an end-user.
  - WebIDs allow associating **data** with end-users in RDF documents.
  - WebIDs allow associating **access control rules** with end-users. Given that Solid uses RDF documents to define access control, this case is a specialization of the previous. It hinges on the correct merging of the name, data, and authentication angles.
  - The WebID profile document associated with the WebID allows applications to **discover** information about the end-user, in particular their associated identity provider and storage setup.
- **WebIDs for software agents and applications**
  - Software agents are similar to end-users with regard to WebID usage, as they always act on behalf of an end-user. We distinguish **interactive agents** (operated in real-time by an end-user) and **autonomous agents** (operated asynchronously).
  - A key difference is the authorization process, which cannot always happen in an interactive way.
  - The analysis for the different angles of the WebID should be done for applications as well, given that there is currently **ambiguity** as to what their identification by a specific WebID means. It could be the software package of the agent, a specific version of that package, a specific deployment of a version, a specific deployment authorized for a specific end-user, or an undefined combination of any of those. Therefore, some concerns and solution directions for the disambiguation of end-user WebIDs also pertain to WebIDs for applications.

Current interactions between WebID actors

Core WebID actors

WebID profile document hosting provider

A WebID profile document *hosting provider* is the party offering access to a WebID profile document.

For instance, the *WebID* (a URL) `https://sophie.solidcommunity.example/profile/card#me` points to the *WebID profile document* `https://sophie.solidcommunity.example/profile/card`, which can be requested from the WebID profile document hosting provider *sophie.solidcommunity.example*. Another WebID could be `https://dani.example/id` and it could point to the *WebID profile document* `https://ids.example/dani/` through an HTTP redirect.

Identity provider

An identity provider (IDP) generates authentication tokens for agents. An agent can present this token to third-party services to support a claim that it has a certain WebID as a name. The token separately indicates the client application by a Client ID. The step-by-step reasoning for this process is as follows:

- A WebID is a string.
- The IDP is aware of a connection between such a string and an agent.
- The IDP offers a (non-standardized) way for the agent to indicate who they are.
  (For instance, the IDP might offer username/password or biometric authentication.)
- In response to a successful indication, the IDP issues a token to the agent.
- The agent uses this token to tell third-party services that this specific IDP vouches for the correspondence of the string to this specific agent.
  (For instance, the IDP might say "the bearer of this token is `https://sophie.example/#me`")
- The IDP might vouch for the stronger correspondence of this string to an actual entity.
  (For instance, the IDP might imply that "this agent is Sophie")

In contrast to the ***one-to-many* relationship between agents and WebIDs**, there exists a ***many-to-many* relationship between WebIDs (URLs) and IDPs**: a single WebID can be authenticated by many IDPs, and each IDP can authenticate many WebIDs. In practice, IDPs tend to restrict themselves to WebIDs (URLs) within a certain address range; for instance, those with a certain top-level domain.

Nonetheless, an infinite number of IDPs can exist for any given WebID; however, only a very small subset can be trusted. Concretely, a token stating that a specific IDP has authenticated a specific agent with a given WebID, by itself does not provide any more information beyond that statement. We must consider it as an unknown agent sharing an IDP-signed but otherwise unverified opinion ("I can prove that IDP *x* says that I am identified by WebID *y*"). The token can only gain a degree of credibility when we establish the IDP can indeed be trusted as a reliable authenticator *for that specific WebID*.

Therefore, whether or not a third-party service chooses to believe the agent's WebID claim, is solely based on trust in the IDP. For non-trusted IDPs (as is currently the majority of the network), trust might be moved to the WebID profile document host instead. The trust is then established by the standardized mention of that IDP in the WebID profile document, which is assumed to imply that the agent has write access to this document and hence is in control of the corresponding WebID.

Authentication-enabled storage and service providers

Authentication with a WebID provides a uniform identity layer to services, of which storage ("Solid pods") is an important class within the ecosystem. Rather than having to authenticate with every service in the network, the Solid-OIDC mechanism provides a single sign-on, wherein agents are identified by their WebID.

There exists a ***many-to-many* relationship between WebIDs (URLs) and services**; in particular, there exists a ***many-to-many* relationship between WebIDs and pods**. So every WebID can provide access to multiple pods, and each pod can grant access to multiple WebIDs. Again, there can exist an infinite number of pods and services that allow access to any given WebID.

It is common—but not mandatory—for the role of a WebID profile document host to be fulfilled by an authentication-enabled storage provider. This means that the WebID profile document for an agent is in practice often hosted on a Solid pod, which the agent would refer to as "(one of) their Solid pod(s)".

Common interactions today

Despite the many-to-many relationships between WebIDs, IDPs, and pods, the current ecosystem tends to combine them in specific ways, suggesting a more coupled interpretation than is prescribed by the underlying technologies.

Currently, people tend to have one WebID, which gives access to one pod, which also hosts the corresponding WebID profile document, and whose underlying server also acts as the IDP. For example, situations such as the following are common—but by no means standardized or mandatory:

- Sophie uses the WebID `https://sophie.solidcommunity.example/profile/card#me` as a name for herself, which she considers to be the only authoritative WebID for her.
- The WebID profile document `https://sophie.solidcommunity.example/profile/card` is stored on a storage service at `https://sophie.solidcommunity.example/`, which Sophie thinks of as her only pod, to which she has read and write access.
- The HTTP server at `https://sophie.solidcommunity.example/` also acts as Sophie's IDP, and she considers this IDP the only one that should be trusted to authenticate her with her WebID.

However, given that the actual relationships are many-to-many, there will in fact be multiple WebIDs for Sophie, each of which can be authenticated by multiple IDPs, and each of which gives access to multiple pods. Her WebID profile document might or might not be hosted on a pod.

Problem spaces and directions

Establishing an unambiguous understanding

An important challenge is to ensure that all parties in the ecosystem have a shared and correct understanding of what exactly a WebID is and is not, and what it means and does not mean.

On the one hand, this is about **correct application of the terms for a WebID and its associated actors**. For instance, "WebID" is currently used in different contexts to either mean a URL, the document identified by that URL, a person, a piece of software, or even the pod of which the agent identified by this WebID is deemed to be the owner. This impreciseness is not limited to colloquial usage, but is also present in technical discussions, where it clouds the understanding of issues and potential solutions.

On the other hand, it is about **agreeing on the exact semantics of a WebID** (URL). There exists an important conflict in which a WebID from the data angle points to a specific person or other agent, whereas from the authentication angle, the same WebID might be considered as an agent within a specific context. The former interpretation is that the WebIDs (URLs) *X*, *Y*, and *Z* could all point to the same person "*Sophie*" and can thus possibly be used interchangeably. The latter interpretation is that those WebIDs (URLs) actually identify Sophie as authenticated by a certain IDP in a specific context, such as *"work Sophie (a doctor)"* and *"private Sophie (my neighbor)"* and *"Sophie authenticated by this highly trusted IDP"*. This is akin to how a person might use different addresses for personal and professional contexts, even though the recipient is always that same person.

These differences are not technical purisms, but on the contrary have **practical relevance, especially regarding trust**. For instance, some services might be tempted to only allow access to specific kinds of WebIDs, similar to how some services require specific email addresses to log on (such as a company or institutional email address). Concretely, services might ask end-users to authenticate with a WebID of specific form (such as `https://trusted.gov.example/{hash}`). When interpreting this request along strictly technical lines, it is hard to understand the rationale, given that WebIDs are strings and that no string is more special or secure than any other. However, in practice, the request is to be understood as a requirement for the authentication process to meet certain security criteria, which arbitrary IDPs will not satisfy. As such, the request is phrased in terms of a mandatory address space for WebIDs, which so happen to share the characteristic that they can all be authenticated via a specific IDP with a certain security level. However, that security resides in the IDP, not the identifier; in fact, many other IDPs also provide a (less secure) authorization for the same WebID, which the service would need to distrust. Furthermore, the trusted IDP could be extended with WebIDs of different forms, similar to how people can register a personal (unsecure) email address to receive notifications from more trusted services, voiding the argument that the form of the WebID serves as a reliable predictor for trustworthiness.

The bottomline is that **different systems today treat WebIDs as different things, thereby hindering interoperability**. Whereas some data systems treat WebIDs as a name for a person, other services treat them as a combination of identity plus an indication of the trustworthiness of the authentication. One possible solution is to be more explicit in the response from an identity provider, by not only returning the WebID but also a proof of how the correspondence of the agent to the WebID was established (for instance, which specific multi-factor authentication was used to initiate the current session, after a previous passport-based identity verification).

Clear and accurate communication

After aligning on a shared understanding of WebIDs, we need to find ways to communicate this understanding to wider audiences with minimal loss of meaning. An angle to consider is the usage of **appropriate metaphors** for WebIDs and Solid in general, and in particular understanding what the limitations of these metaphors are in different contexts. Below is a non-exhaustive list of examples:

- The ***identity card*** metaphor partly works, in the sense that a WebID profile document tends to contain some basic attributes about the agent it describes. However, WebID profile documents are not issued by an authority such as a government, meaning that they do not carry the same identifying attributes (such as a photo and biometric attributes) nor the same trustworthiness of the mentioned data (such as name and date of birth). Furthermore, it does not sufficiently recognize the fact that people have multiple WebIDs (whereas relatively few people have multiple identity cards), and that there can be multiple identity providers (whereas there is typically one authoritative regional body for an identity card).

- The ***email address*** metaphor is partially helpful, in the sense that it acknowledges that people use different email addresses for different occasions. This can for example be done for *routing* purposes (professional and personal emails are checked at different times), or for *anonymity* or *pseudonymity* (through so-called burner accounts). People can analogously have multiple WebIDs, each of them pointing to the same person (similar to how a person eventually receives the email sent to their different addresses). However, the comparison breaks at the moment we consider the relationship between WebIDs and pods, since one WebID gives access to multiple pods, whereas each email address for a person tends to correspond to one mailbox.

- The ***access badge*** **or** ***key*** metaphor might help, given that different companies can issue access badges, and they might provide access to different buildings and rooms (and differently for different people). However, people tend to also carry different access badges and keys, whereas a WebID would be able to act as a more uniform kind of key; so there would be no necessity to mint a new WebID merely to access a new service.

Improved trust mechanisms

As discussed above, the current authentication method assumes trust in either the IDP, or in the WebID profile document host (which expresses its trust in the IDP). The absence of a trust authority is due to the **decentralized nature of authentication**. A risk is that, in order for a service to establish a higher degree of trust in the WebID/agent correspondence, the service establishes closer trust relationships with specific IDPs only, and hence possibly specific WebIDs only. This would lead to siloization, where certain services require people to choose certain IDPs, leading to an undesired proliferation of authoritative WebIDs and the associated management thereof.

On the one hand, we could **establish independent trust in IDPs** via *standardized certification processes* (similar to how TLS certificates) to prove a certain level of security, or as part of a *federated trust process* (such as OpenID Federation) to prove mutual trust. Because of their complexity, certification processes might remain limited to niche use cases, such as government identity. On the other hand, the IDP could provide **additional security guarantees in the token** it issues to agents (such as the `acr` claim in OIDC). As discussed above, this also removes implicit contextual meaning from the WebID.

URIs and their resolvers

The name/locator duality of a WebID is based on the fact that it is an HTTP URL, which is both an *identifier* and *locator*. Namely, each HTTP URL identifies a certain resource, and also contains the instructions to locate a representation of that resource through the HTTP protocol. Therefore, if you know a person's WebID, then you can *dereference* that WebID by looking it up using the HTTP protocol, which should lead to a document describing the corresponding agent.

On the one hand, the WebID mechanism conceptually has **few dependencies on HTTP**. All that matters is that there exists a ***lookup function*** that leads from the (textual) WebID to a document describing the agent that has this WebID as a name. However, even for HTTP URLs, we could imagine the creation of different lookup functions beyond the standard `GET`-based lookup afforded by the HTTP protocol. For instance, there might be a service accepting an HTTP URL and returning an associated document generated using other means.

On the other hand, we could **imagine authenticating with a WebID *without* any lookup functionality**, if we leverage alternative IDP trust mechanisms as discussed previously. A main purpose of the lookup is indeed to establish an ad-hoc trust relationship for a particular WebID between the service and the IDP, but only in absence of a more sustainable trust relationship. Ultimately, an IDP's task is to sign a claim relating a certain identifier to a certain agent; whether or not this identifier is a WebID, another type of IRI, or another string altogether, is not a concern for the IDP itself. Lookup of that identifier is not a necessity if the service has another way of assessing the trust in the IDP's claim.

Both observations clearly pave the way for extending the URI space of WebIDs from HTTP URLs to **decentralized identifiers (DIDs)**. Similar to how HTTP URLs have HTTP as a resolver, each DID method has its own type of resolver. A classic problem with building DID systems is the decision of which DID methods to support, given that each requires its own implementation—and the list of methods is still being extended. As such, the required number resolvers might be large, although they might be run on federated infrastructure. However, such resolvers are only necessary when lookups are needed. If, in contrast, trust between a service and an IDP can be established in different ways, then DIDs can be treated as simple text strings as far as authorization is concerned.

Using DIDs could alleviate some **problems inherent to the management of HTTP URLs** (and thus WebIDs). One such problem is that people could lose control over the domain name of their WebID, and hence over the resolver function and thus the contents of the corresponding WebID profile document. Another such problem is that, when deliberately moving domain names, a new WebID would need to be minted. Certain DID methods can minimize these risks as one of their purposes is to provide different means of control over the resolver function, such that they are not tied to DNS.

Anonymity and pseudonymity

The high-level idea of Solid with WebIDs is the usage of **identity as a token** for being granted access. Simply said, a person tells a service who they are using a WebID as a name, and then this service decides to grant or deny access to them.

This contrasts with approaches such as Verifiable Credentials, where an agent will share one or more data attributes in order to indicate its eligibility for a certain purpose, such as being granted access to a certain resource in a storage location. Credentials might indicate, for instance, whether the agent is older than 18, or whether they have a certain diploma or currently work for a certain employer. On an abstract level, we could consider WebIDs as a specific kind of credential: the attribute an agent is sharing, is their WebID. In that sense, the difference between Verificable Credentials and WebIDs is gradient rather than a contrast. **WebIDs are typically more granular** as authorization (because they point to a specific agent), compared to the broader Verifiable Credentials that tend to identify a certain class of agents (e.g., all people over 18 years old).

The current practice is indeed that the majority of WebIDs lead to a uniquely identifiable agent, by design. This might lead to the impression that Solid applications "always know who you are", or that Solid would signify the end of anonymity on the Internet. Such a conclusion would be counter to some of the Solid aims, as Solid strives to enable data sharing in a privacy-friendly way—and mandatorily sharing a highly granular identifier would introduce a contradiction.

Apart from using Verifiable Credentials—which should be investigated as a broader class of partial identity sharing through attributes—we can leverage the one-to-many relationship of agents to their WebID. Basically, an agent can use **different WebIDs for different interactions**, not unlike how password managers support people in using different logins for every website.

Some of these WebIDs could serve the purpose of an **anonymous identity**, e.g., they could be burner accounts with the intention of never being traced back to a more widely known name of the agent. Other WebIDs could serve as a **pseudonymous identity**, e.g., they could be accounts with the intention of a more widely known name only being revealed at a later point in time and/or to a smaller audience. An anonymous identity becomes pseudonymous or public when the agent carrying the anonymous WebID communicates the association with a more widely known WebID.

The management of anonymous and pseudonymous WebIDs can be taken into consideration with the management of multiple WebIDs in general.

Disambiguation of multiple WebIDs

Finally, given that agents will have multiple WebIDs, we need a strategy of dealing with this reality such that agents can still have a unified experience across their WebIDs. The first step is the aforementioned **unambiguous understanding** of what a WebID is, and whether it indeed consistently identifies an agent (as opposed to an agent within a specific authorization context). Furthermore, any disambiguation needs to **consider all of the angles** of the technical interpretation, for each of which we provide initial guidance below:

- **Text:** On a textual level, we might want to establish whether certain identifiers are equivalent. For instance, URLs might differ in protocol (`http:` or `https:`), or might have encoding differences (`%2f` versus `%2F`); we need an explicit strategy to decide whether they are consistently different or the same. The current consensus is to only allow textual equality.

- **Name:** Of key importance is that equivalence of two or more WebIDs cannot be established only unilaterally. Otherwise, we risk believing a malicious agent falsely claiming that one of its WebIDs is equivalent to a WebID of another agent. In the case of anonymity and pseudonymity, a bidirectional declaration of some degree of equivalence might compromise the confidential nature of that relationship. Furthermore, we need to carefully consider in what cases different identifiers for the same agent can be used interchangeably, or when systems need to treat equivalent identifiers as if corresponding to different agents.

- **Data:** When different WebIDs (URLs) are used as names for the same agent in one or more data documents, we want to establish the conditions under which they can be treated as the same. On the one hand, this involves a strict understanding of whether the different WebIDs are exactly the same, or only contextually the same. For instance, the same person might be an employee of one institution, but a student at another; the claims that the person is a student or employee might or might not be simultaneously true in a given context. On the other hand, we need to take into account anonymity and pseudonymity, in cases where the agent does not want attributes from different contexts to be carried over, even if they describe the same agent.

- **Locator:** As far as the lookup function is concerned, we might want to investigate whether different WebIDs could or should resolve to the same WebID profile document, or to different documents that are somehow interlinked with each other. A solution path described in a previous section is to reduce the reliance on resolvers and lookup functions altogether, which also eases the transition to other kinds of identifiers beyond HTTP URLs.

- **Authentication:** Different IDPs might have restrictions as to what WebIDs they are willing to provide authentication for, but might be willing to accommodate alternative WebIDs for the same agent. Separately, some *authorization* systems might refer to the agent by one WebID and expect only that WebID as proof of identity, whereas other systems might be inclined to accept any WebID that corresponds to the agent in question. In any case, the symmetry conditions for interchangeability must be satisfied before even attempting such resolutions.

In designing solutions, we must consider the applicability across interpretation angles. For instance, it might become difficult if two WebIDs are considered equal as far as data is concerned, but separate for authentication; or perhaps this might be considered an advantage or necessity in other contexts.

Acknowledgements

This work is supported by SolidLab Vlaanderen (Flemish Government, EWI and RRF project V023/10).
The author wishes to thank Digita (Tom Haegemans, Wouter Termont) and Digital Flanders (Laurens Debackere) for reviews and discussions.