ID4me Verified Identities

Version 1 – 30 September 2019
Author: Vittorio Bertola – vittorio.bertola@open-xchange.com

INDEX

SCOPE OF THIS DOCUMENT

This document describes the operational and technical architecture for providing interoperable “verified identities” within the ID4me framework. Readers are expected to be familiar with the basic concepts of the ID4me architecture before reading this document – if not, please refer to the introductory documents on the ID4me website.

THE CONCEPT OF VERIFIED IDENTITIES

The ID4me protocol was designed to allow the federated login and distribution of personal information (under the form of so-called "claims") across the Internet, without necessarily ensuring that the information provided by the claims is “true” (by any meaning of the word).

However, for many use cases it is necessary that the providers of the identity (the identity authority and the identity agent) guarantee to the relying parties that the information provided by the claims has been verified in some way and can be trusted. As federated interoperability on an Internet scale is one of the objectives of the ID4me project, some aspects of this verification need to be standardized.

An **ID4me verified identity** is thus defined as *an ID4me identity that includes additional information that proves that some or all of its claims have been verified by a party other than the user*.

Identity verification is thus *the process through which a party other than the user verifies the truthfulness of the information about the user stored in the identity*.

The objective of the standardized procedures and protocol extensions outlined in this document (and, if necessary, better specified one by one in other dedicated documents) is thus to provide relying parties with an understanding of the type and meaning of the additional information that they will receive as part of ID4me verified identities.

However, it will still be up to each relying party to decide whether they want to accept all ID4me identities, including the unverified ones, or whether they want to restrict access only to verified identities, or even only to those identities that have been verified in specific ways and/or by specific parties. This really depends on the use case and on its requirements; to help relying parties, some suggestions for specific use cases are included in the final part of this document.

PARAMETERS OF A VERIFIED IDENTITY

Multiple parameters can affect the trust that a relying party can attribute to an identity. First of all, each identity is associated to its **identity authority** and its **identity agent**. When verified, an identity is also associated to a **data authority**: a party that deals with verifying some or all of the claims.

The role of data authority can be performed by a specialized third party, or by either the authority or the agent. The agent, in particular, already deals with the collection and distribution of the user’s claims, and so it is well positioned to also perform their verification; in this case, it can just add the related information into the responses that it already provides to the relying party.

On the other hand, if the role of data authority is performed by a separate entity, this entity could provide signed **assertions** to the agent, which will then forward them to the relying parties as part of the *"userinfo response"*; an assertion is made by one or more couples of claim name and value, plus a digital signature and other information that testifies that the values of the claims have been verified, by whom, and how. Alternatively, the data authority could communicate the verified claims directly to each relying party, as the agent could delegate some claims to the data authority using the *"distributed claims"* mechanism of OpenID Connect within its own response to the relying party.

The verification of the identity – be it performed by a data authority, or by the identity agent, or by the identity authority – allows to attribute to it a **level of assurance**: a parameter that conveys in a standard way the thoroughness of the checks that were made when releasing the identity, and later during the authentication process. The lowest level of assurance (equivalent to no assurance at all) would involve self-declared data and simple authentication, while a higher level might require the user to appear in person in front of the verifying entity and show one or more documents to prove their identity, and then use multiple factor authentication when logging in.

To increase interoperability, levels of assurance are standardized; unfortunately, multiple competing standards exist, and so a further parameter is the **trust framework** that was followed in the identity verification process. As most of the early deployments of ID4me are in Europe, ID4me recommends the adoption of the eIDAS trust framework as the reference standard by all operators[1], but other frameworks are available (e.g. ISO 29115).

Optionally, depending on the trust framework, the agent could also provide additional information that might be useful to the relying party to determine and track the assurance, such as additional details on the verification procedure, or a unique id that would allow to track the specific authentication operation if necessary in the future.

In addition, a fourth party can come into play: an **auditing partner**, i.e. an entity that does not deal directly with the identity and with its owner, but that verifies and certifies independently that the other parties are who they say they are and follow reliable procedures. The auditing partner is similar to a *"conformity assessment body"* in the eIDAS framework, and, when implementing it, ID4me will explore opportunities for synergies between the two concepts.

The ID4me association plans to strike deals with auditing partners so that they can provide digital certifications to ID4me operators; these certifications can increase the amount of trust that relying parties are willing to attribute to the ID4me identities that they manage. So, thanks to one or more auditing partners, a **level of operational trust** can be attributed to each identity authority[2], identity agent and data authority, validating who they are, their internal procedures, their quality assurance mechanisms, their data protection measures and so on. The level of operational trust of each of the involved operators can then be communicated to the relying parties together with the identity and its level of assurance. The level of

[1] This does not imply that the identity would necessarily be valid under eIDAS or that ID4me operators have to be accredited under an eIDAS-compatible scheme, as that would require significant additional effort by the operators to meet the requirements of their national eIDAS implementation. It just means that the same procedures and definitions adopted for levels of assurance in the eIDAS framework (i.e. the Commission regulation 2015/1502/EU and the related guidelines) are used by the operators to verify the identity. The additional level *"None"* can be used to specify explicitly that the claims have not been verified.

[2] Differently from the levels of assurance, the names and definitions of the levels of operational trust are peculiar to the ID4me framework and will be later defined in a specific document.

operational trust can also be retrieved by other ID4me participants that want to assess the operator's reliability.

This level of operational trust is securely described and communicated to the relying party through a **chain of trust**, i.e. a set of digital attestations each signed by the previous node in the chain, all the way up to a **trust anchor**. The trust anchor represents the ultimate origin of the trust in the entire federated identity system; for generic ID4me verified identities, the ID4me association will act as the trust anchor, selecting the auditing partners that in turn will certify the operators. The specific technology used to implement this chain of trust and to make it securely and automatically verifiable by relying parties will be defined separately.

Given the role of the trust anchor as an independent and original source of trust, each new trust anchor inherently defines a new **identity federation**. ID4me supports multiple identity federations at once, and each ID4me identity can be part of any number of identity federations, enabling ID4me users to reuse their single ID4me identity in multiple environments if they want.

The most basic, default identity federation in ID4me is the **ID4me federation**, stemming from the ID4me association as a trust anchor; unverified identities are also part of the ID4me federation. However, depending on the use case, any party may introduce an additional trust anchor and thus an additional identity federation for specialized usage; see the examples at the end of the document.

REQUIREMENTS FOR OPERATORS

ID4me operators that want to provide verified identities have to perform specific activities; here follows a brief description.

The identity agent, being the party that practically collects the information from the user, is also the originator of any data verification process. When creating or updating the identity, the identity agent can perform the verification directly: in this case, it is up to the agent to follow the procedures described in the selected trust framework, according to the level of assurance that it wants to provide to its customer. This might involve, for example, checking the customer's documents or, for high levels of assurance, having the customer show up in person in a physical point of presence, e.g. a shop. After performing such verification, the agent will add to the identity a set of additional claims that describe the level of assurance and the other parameters, turning it into a verified identity.

The agent might however also decide to delegate the verification to a specialized third party, i.e. a specialized data authority. In this case, the agent is still responsible to ensure that the data authority correctly performs the verification; there are however two different options to choose from, depending on the agent's commercial policies and on the use case.

One possibility is that the agent still keeps the "ownership" of the assertion on the level of assurance; in this case, the data authority will not be visible to the relying parties and will just act as a subcontractor of the agent. Another possibility is that the agent devolves that ownership to the data authority, which will then sign an assertion on the level of assurance directly; the relying party will then understand that the data authority, rather than the agent, is the source of trust in the identity's claims. It would even be possible to have multiple data authorities for different subsets of the identity's claims, or multiple assertions for the same claim (the technical feasibility of these features within the OpenID Connect standard is still being analyzed).

The identity authority is also involved in determining the level of assurance that can be attached to an identity. Levels of assurance generally include requirements on the authentication phase; thus, whoever attaches a level of assurance to a set of claims must also be sure that the authority meets those requirements and has enabled the appropriate features (e.g. two-factor authentication) for that identity.

More generally, this shows an issue which is still open for discussion, and which will possibly only be solved through the experience deriving from early implementations. To guarantee a level of assurance for a given ID4me identity, a cooperation of all the operators – the identity agent, the identity authority and (if any) the data authorities – is required; all of them must be aligned and apply the appropriate procedures to the identity, depending on the level of assurance that they want to guarantee. Thus, there needs to be some kind of coordinating entity that, on request by the user, tells all the operators to adopt a given level of assurance and checks that they actually did.

As the customer-facing entity in the provision of ID4me identities is the identity agent, one could imagine that this role is assumed by the agent. However, depending on the commercial and organizational model, it could also be the authority, as the central "registry" in a network of "registrars", to do this coordination.

Generally speaking, this framework does not address the issue of commercial models behind the provision of verified identities. At least two business models have been tried in similar cases:

1. The user pays for the cost of the verification, because a verified identity is required for access to services that he/she needs;
2. The relying party pays for the use of verified identities, for example through a micropayment to the identity agent or to the data authority whenever the verified claims are accessed, because this saves the relying party the cost of verifying the identities of its users directly.

Both (and more) are possible within this framework, though the second business model would possibly require the establishment of standard interfaces and/or a clearinghouse for the payments. This will be addressed if demand arises.

GUIDELINES FOR THE RELYING PARTIES

When implementing support for ID4me identities, relying parties have to decide whether they need trust in the identity, and up to which level.

For ordinary, everyday logins, relying parties might just decide that they can accept any information that they receive about the identity, even if it has not been verified, or that they will perform any out-of-band verification themselves, outside of the ID4me system, through traditional *"know your customer"* procedures. In this case, they can just accept any identity, even if unverified or self-hosted; this will also maximize the potential user base that they can reach.

On the other hand, if the relying party offers services that require stronger authentication of the user, then they could require a verified identity of a certain kind – or, as an intermediate step to avoid rejecting users, they could accept any identity but only make some functions available to users that are logging in with a verified identity or that have been independently verified by the relying party.

However, it is important to understand that not all verified identities are the same – exactly as, in real life, the trust you can place in someone's information is different depending on which document(s) they show and who released them. Thus, when deciding that they need some kind of verified identity, relying parties also need to ask themselves:

- Which level of assurance of the identity do I need, and under which trust framework?
- Which level of operational trust do I need in the operators that manage the identity?
- Which identity federation(s) am I going to accept, and thus, which trust anchors am I going to rely upon?

The easiest, more straightforward way is to consider only the ID4me association as a viable trust anchor, use eIDAS as the trust framework for identity assurance, and pick a minimum level of assurance and a minimum level of operational trust according to the best compromise between certainty on the identity and ease of use/broadness of user base (this is really an evaluation that only each relying party can do for itself). However, several further options are available depending on the use case – in small identity federations, for example, relying parties could just hardcode a list of trustable identity agents and/or identity authorities.

The objective of the ID4me framework is to automate all the technical operations that a relying party has to do to gather, authenticate and analyze all this information, so that each relying party can just make an API call and retrieve simple indicators of the overall trustability of the identity.

However, depending on the technology that will be used to implement the chains of trust, there might be the need for relying parties to do some initial manual installation – for example, they might have to pick the trust anchors that they want to trust and retrieve some cryptographic information that will allow them to verify later communications. This will be clarified and described in future specifications.

Moreover, future specifications will also address the issue of how can the relying parties be informed when the trust in an identity or in an operator should be updated, for example following an unverified update in the data or the de-accreditation of an operator.

SAMPLE USE CASES

This section describes possible deployment models for common use cases in which some degree of trust is required. Some of these use cases envisage the creation of an additional, specialized identity federation on top of ID4me.

Use cases can be broadly classified under one of two models: a **closed** model, in which only ID4me identities that have been specifically created within that environment can be accepted for login; and an **open** model, in which any ID4me identity can be reused within that environment. The choice between the closed and the open model is a matter of policy and needs to be made by the relying party (or by the identity federation, representing a cooperative set of relying parties) for each use case; however, while the closed model allows for more control over the system, the open model is the most convenient one for the users and the one that fully exploits the potential of the ID4me architecture, and thus is recommended whenever possible.

A first set of possible use cases follows (please note that they sometimes refer to the use of cryptographic keys and certificates, but different solutions could be devised instead). Many more are possible; the examples below are meant to illustrate the flexibility of the ID4me architecture, and at the same time provide guidance for those who want to use ID4me for one of these situations.

I. Independent relying party needing trusted information (open)

Description: An online service wants to accept ID4me identities for login but requires a certain level of assurance of the data (or at least a part of them).

Solution: The service acts as a relying party which, after installing the ID4me federation's public key, will only accept identities that meet specific minimum levels of assurance and of operational trust. The level of assurance can be verified either by looking at the related claim in the identity token supplied by the agent (in which case, the agent is responsible for its reliability), or by looking for the assertion of a level of assurance signed by a data authority. In both cases, the relying party should also verify that the agent or the data authority can claim a certain level of operational trust, through an additional assertion that uses a chain of trust that gets back to the ID4me federation's public key; otherwise, any malicious party could just set up a malicious agent or data authority and declare fake levels of assurance. The relying party should also verify that the authority has a similar level of operational trust.

II. Corporate/organizational usage (closed)

Description: A corporation or organization wants to use ID4me for single sign-on by its members (employees…) and collaborators (consultants, suppliers…) across all its services; it also plans to provide identities to all the users of its system, and only wants to accept these identities.

Solution: The organization establishes (or gets as-a-service from a supplier) an internal identity agent; it can also run the identity authority, or it can outsource it to a third party as preferred. All identities that are to be valid in the system are created on that identity agent, where any access control parameters (e.g. permissions) are set, using a custom claim or any standard OpenID Connect mechanism. At login, each relying party in the system verifies that the identity is managed by the internal identity agent and refuses to accept any identity which is run by any other agent.

III. Community usage (closed)

Description: A community (a broader group of organizations, a city administration…) wants to use ID4me for single sign-on by its members, and only wants to accept identities within its community, but wants to allow multiple entities in the community to provide identities. *[The difference with case II is in the presence of multiple identity agents rather than just one.]*

Solution: The community establishes a trust anchor person or entity, which will have the role of maintaining the list of active and trusted identity agents within the community, also applying any policies and checks if necessary. All accepted identity agents receive a certificate signed by the trust anchor, that testifies that they are part of the system. At login, each relying party in the system verifies that the identity includes a trust chain of certificates that goes back up to the trust anchor's key and refuses the identity otherwise. Optionally, the trust anchor can define or adopt levels of assurance, which the agents will implement and the relying parties will check when necessary.

IV. Corporate/organizational/community usage (open)

Description: A corporation, organization or community wants to use ID4me for single sign-on by its members (employees…) and collaborators (consultants, suppliers…) across all its services (potentially a significant number of services, making per-service account provisioning impractical), but wants to accept any ID4me identity for login, run by any entity, provided that it has been vetted in advance (e.g. by the user contacting internal IT and getting their rights of access validated). The organization may or may not supply some of the identities directly; it could also strike a deal with external agents and authorities and get identities from there, or just tell its users to get an ID4me identity anywhere they want.

Solution: The organization establishes itself as the trust anchor of a new federation (in the case of a community, the role of trust anchor has to be attributed to someone trusted and willing to run it). When a new user needs to gain access to the system, the trust anchor will digitally give the user an assertion, cryptographically signed with the federation's private key, that testifies its right of access, including any permissions; the user will store this assertion within a claim in their identity, entering it through their agent's interface. At login, each relying party in the system requests that claim and verifies that it includes a valid assertion signed with the proper key (relying parties need to install/retrieve the federation's public key).