Technical overview

Version 1.2.2 – 19 January 2018
Author: Vittorio Bertola – vittorio.bertola@open-xchange.com

INDEX

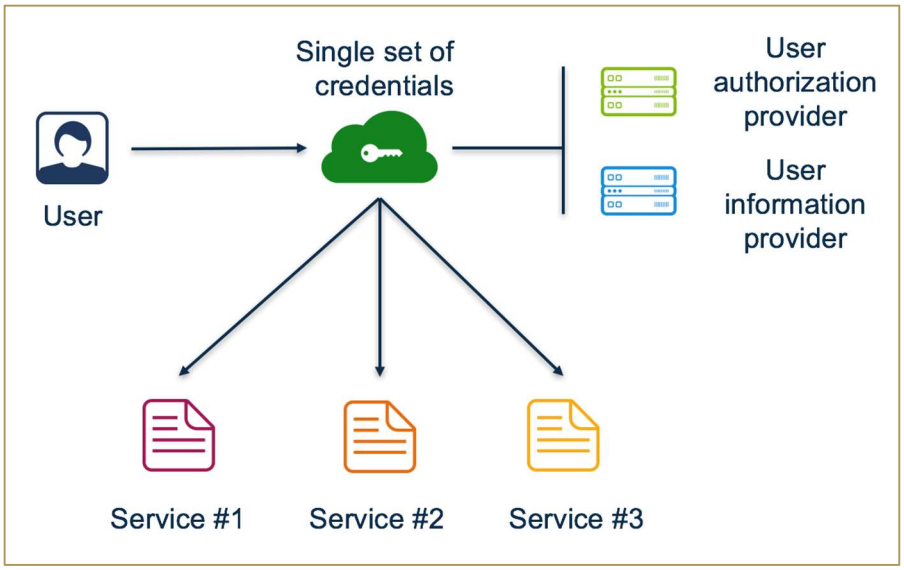
SCOPE OF THIS DOCUMENT

This document describes the basic technical architecture and features of the proposed id4me standard for federated online digital identity services. While this document does not represent a full technical reference, it can help technical experts to understand the id4me standard and implementation requirements in detail, including a point-by-point comparison with the existing OpenID Connect standard.

KEY FEATURES

id4me is a federated digital identity service that aims to provide two main functions:

- Authorization of a user for access to any third party accepting id4me identifiers ("single sign-on" on an Internet scale);
- Controlled communication of the user's personal information to the third parties accessed by the user.



In other words, a user that owns an id4me identifier can use it to log into any website or online service supporting id4me, even without prior registration; on first access to that service, the service can request access to the user's personal information as entered by him into his profile; if the user consents to this access, the requested information will be made available to the service, which can thus automatically create a local account or profile for the user, associated to his id4me identifier.

The service is federated, meaning that multiple interoperable providers of identifiers can exist, including personal providers self-hosted by their users, and that all of them are intrinsically supported by any online service implementing the id4me standard. Users are free to pick any provider and to move their identifier to a different one whenever they want.

id4me is, in itself, a "weak" identity standard; the purpose is to authorize the user, i.e. to ensure that the user of a given identifier is always the same that initially acquired that identifier at registration. Accordingly, there is no initial actual authentication of the user, and his identity and personal information are entirely self-declared, as it currently happens for most online registration systems. The standard may however be extended to support third-party validation of the user's personal information and thus provide stronger proof of the user's real world identity.

To make implementation easier, id4me builds over the existing OpenID Connect/OAuth 2 standard, but it expands it on a number of dimensions. Thus, all communications between the different parties happen via HTTPS, and make use of signed JSON web tokens (JWTs). This choice also allows to make the user experience easy to understand for users, as the flow is similar to currently existing OAuth-based authorization systems, and to reduce implementation costs, as existing OpenID Connect server and client implementations can be used as a starting point, and in some cases they only need some appropriate configuration and customization to work.

Moreover, id4me also builds over the Domain Name System (DNS), the most fundamental directory of the Internet. The DNS, with the use of DNSSEC, provides data authentication and integrity as servers exchange information with each other; the standard does not require any out-of-band, real world authentication of the various parties in advance, making it possible to scale easily and to build a decentralized federation of actors, similarly, to how email works. The DNS is also used to store public information about each id4me identifier, making it simply accessible throughout the entire Internet.

While the entire system is based on HTTPS, nothing prevents clients for other protocols (SSH, IMAP…) from implementing the id4me login flow, as long as they are able to ask their user for the identifier, for the authentication credentials and for consent on which claims should be shared (either interactively during the login process, or as secure configuration entries supplied once for all at installation) and to make programmatical REST-like HTTPS connections to perform the authorization.

ELEMENTS OF THE ARCHITECTURE

The id4me standard refers to the following four roles.

User: Any physical user of the Internet, but also any entity (company, organization…) or even software / service instance, that needs to authorize himself/itself on online services.

Relying party: An online service that uses id4me to validate user logins.

Identity authority: An online entity ensuring the correct authorization of the user at every login, by storing and verifying his authentication credentials, such as his password. It represents the “back-end” and trust source for the provision of the authorization service, but it generally does not deal directly with the user, and does not store or supply his personal information.

Identity agent: An online entity providing the id4me service to users. It represents the “front-end” for the provision of the authorization service, selling or supplying id4me identifiers to users and creating them at an identity authority. It also stores the user’s personal information, communicating it to the relying parties when the user consents, and can potentially provide the user with reports and statistics on the service.

The separation of roles between the authority and the agent is similar to the separation of roles between the registry and the registrar in the domain name industry.

The id4me standard allows users to identify themselves by an **identifier**. Any valid hostname in the Domain Name System can be used as a id4me identifier, even if it does not correspond to any existing host and it is not associated to any A/AAAA record. For the system to function, it is necessary to perform a DNS query and retrieve some data stored under that hostname. This implies that, to be able to use a hostname as his id4me identifier, the user needs to have access to its DNS zone and to be able to edit and publish it, either because he owns the domain, or because he has been allowed by the owner of the domain.

This allows the co-existence of many possible service scenarios – these are just a few examples:

- The user buys in a bundle a personal domain name, the id4me service and the DNS management service; the service provider registers the domain name, sets up the name servers, creates the identifier by an identity authority of their choice, and adds the DNS records that allow the identifier to work.
- The user already owns and manages a personal domain name, and buys only the id4me service; the user can then configure the appropriate DNS records so that they point to his id4me provider and allow the identifier to work.
- The user owns a personal domain name and runs the name servers on his own servers; he can install an id4me server application, add the appropriate DNS records and run his id4me identifier entirely on his own.
- The user acquires the id4me service from his ISP; the ISP gives him an id4me identifier in the ISP's domain name and sets up the appropriate DNS records, either running the id4me service on their own or relying on external identity agents and authorities; no new domain name is registered.

While the identifier is the user-friendly handle that people enter to identify themselves, identifiers should not be used internally by relying parties as the primary key identifying the user. There are privacy and security reasons, including the fact that the same identifier could be reused by a different person if the ownership of the domain name changes, to introduce a separate **identity handle**, specifically designed to be used as the primary key identifying a specific identity. There is no guarantee that the same identifier will always refer to the same online identity, but the same identity handle will.

The identity handle is returned by identity authorities to relying parties on authentication, under the form of the *"subject identifier" ("sub"* field in the identity token) defined in the OpenID Connect core specification. Identity authorities can decide whether to provide the same identity handle to all the relying parties having access to a specific identity, or whether to provide a different identity handle to each relying party.

The id4me identifier is associated to **authentication credentials**. In its simplest form, the credential for authentication is a password, but the system could support different types of credentials and multiple authentication methods, for example implementing two-factor authentication. A key element of the architecture is that only the user and the identity authority know the authentication credentials, while neither the identity agent nor the relying parties ever gain access to them.

The user's personal information is represented in id4me by a set of **claims**. A claim is a couple made by a standard claim name, identifying the information field (name, address…), and by its value, which are different for each user. The claims are provided by the user to his identity agent, and communicated by the agent to the relying parties, as a signed JSON web token, only after the user has consented, in front of the identity authority, to each single sharing of information with that specific party.

id4me introduces a dedicated **id4me DNS record format**, which is used to announce public information about an identifier. Initially, the three information elements that must appear are the protocol version used (*"v"*, with a standard value of *"OID1"*) and which identity authority (*"iss"*, issuer in OpenID terminology) and identity agent (*"clp"*, claims provider in OpenID terminology) are managing the identifier.

Similarly to other protocols using DNS-based information elements, such as SPF and DMARC, the information is stored in the DNS via a TXT record containing a string of name-value couples; the record is associated to a standard hostname obtained by prepending *"_openid."* to the identifier itself.

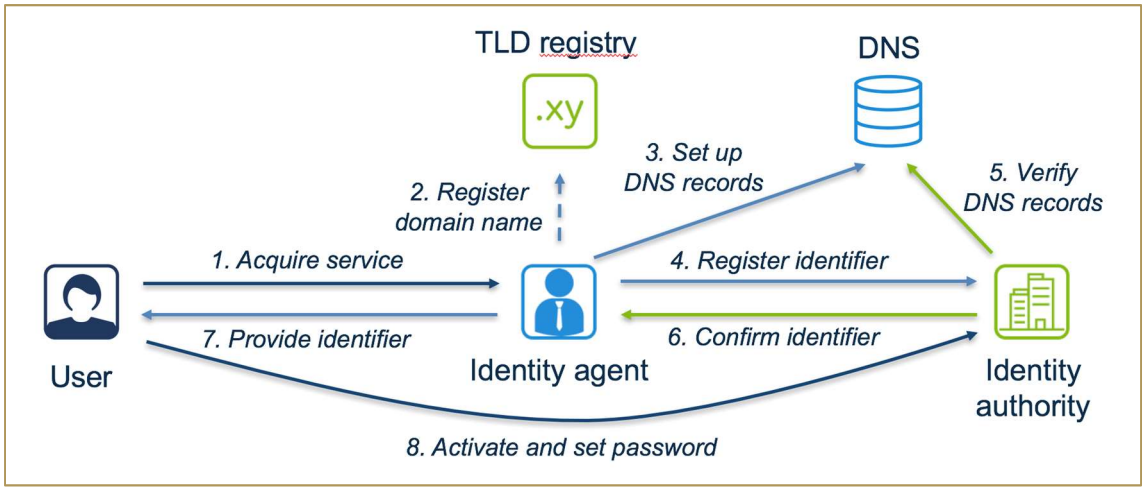
For example, for the identifier *"myname.example.org"*, the following record could be defined:

_openid.myname.example.org IN TXT "v=OID1; iss=my.idauthority.de; clp=service.idagent.com"

The existence of this record in the DNS also shows that the identifier has been correctly set up; if the zone is secured with DNSSEC, it also gives a good degree of certainty on the fact that the user either owns the *"example.org"* domain name or has been authorized by its owner, thus suggesting that the use of the hostname as identifier is presumably legitimate.

REGISTRATION OF A NEW IDENTIFIER

The registration of a new identifier can be described, in a high level summary, by the following diagram.



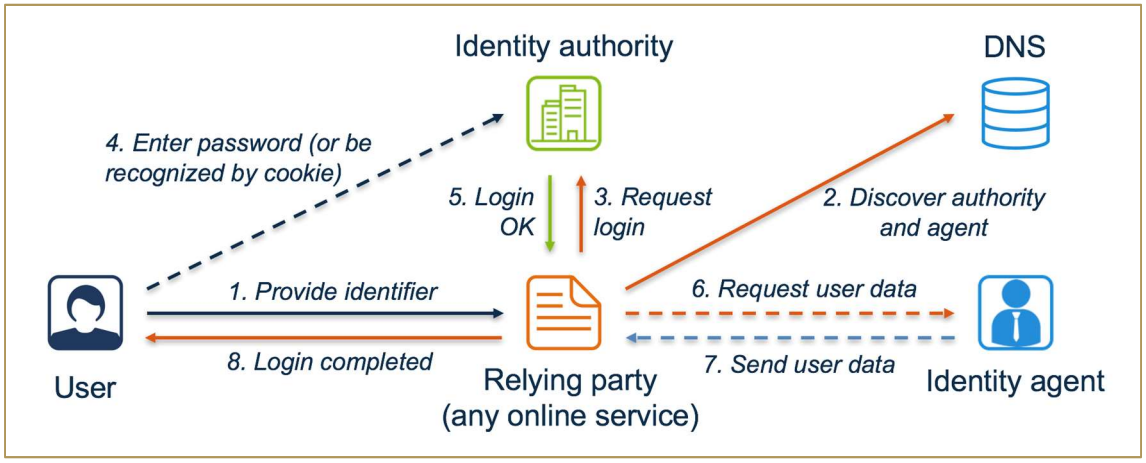
The process starts by the user acquiring the service and requesting the creation of a new identifier to an identity agent. If the identifier belongs to a new domain name, then the agent will first of all register it from the appropriate TLD registry. The agent will then set up the identifier on its internal service platform; if it manages the name servers for the domain name, it will also set up the id4me DNS record described in the previous section (if the name servers are directly managed by the user, the agent will rather provide the user with the appropriate record, which the user will have to add to the zone on its own).

After these preliminary actions have been completed, the agent will contact an identity authority to actually create the identifier. To have the proof that the agent and/or the user are really in control of that domain name, the authority will run a challenge to get a domain name proof-of-possession; in the prototype, a DNS-based challenge under the ACME protocol is used, but other types of challenges could be added in the future.

After the validation of the DNS configuration is successful, the authority will confirm the creation of the identifier to the agent, which will in turn confirm it to the user. It will also redirect the user to a one-time URL hosted by the identity authority, that will allow the user to set the password or any other authentication credentials, and confirm the final activation of the identifier.

USE OF AN IDENTIFIER FOR LOGIN

Whenever a user wants to use an identifier for authorization in front of a relying party, the following high-level diagram will be followed; it is a standard OpenID Connect Authorization Code Flow, with the id4me DNS-based discovery procedure in front of it.



The process starts when the user gets to the login page on the relying party's website, or to the equivalent interactive feature on non-Web-based services. There, they are given the option to pick *"Login with id4me"* and to provide their identifier (not their authentication credentials).

The relying party, possibly using a standard and public API which implements its side of the protocol, will first of all discover which identity authority and agent are managing the identifier that was provided by the user, by making a DNS query for the id4me DNS record described above (step 2 in the diagram).

Then, if the relying party discovers an unknown authority, and only for the first time, it has to perform an automatic configuration and registration operation (not shown in the diagram): as per the OpenID Connect Discovery standard, it will append *"/.well-known/openid-configuration"* to the authority's hostname as retrieved from the DNS record, and it will perform an HTTPS connection there, receiving back a JSON object that contains all the configuration parameters for the OpenID/OAuth service. One of these parameters is the URL of the authority's *"registration endpoint"*; the relying party will then perform a second HTTPS connection to this endpoint, following the OpenID Connect Dynamic Client Registration protocol, and will receive back a couple of credentials (*"client_id"* and *"client_secret"*) that will be used to authenticate the relying party in front of that specific identity authority in any future operation. The relying party can then store locally these credentials for reuse, so that this setup operation does not need to be repeated every time; it will need to repeat this operation only if these credentials expire. As this operation is entirely automated and does not require any out-of-band interaction (identity authorities must not require any pre-existing authorization for access to the registration endpoint), this model is fully scalable and allows easy support for any number of relying parties and identity authorities.

At this point, the actual login procedure begins, following a standard OpenID Connect flow; any existing library implementing OpenID Connect should be able to perform the rest of the procedure out of the box. The rest of the user experience will thus basically be the same as for the existing OpenID Connect/OAuth based authorization systems, such as *"login with Google"* and *"login with Facebook"*.

While for simplicity the high-level diagram shows a direct login request by the relying party to the identity authority (step 3), in reality this request is mediated through the user's browser; the relying party, after receiving the identifier as input, discovering the authority via DNS and performing the setup operation if necessary, will return to the user agent an HTTPS redirection, pointing it to the authority's *"authorization endpoint"* URL as retrieved among the configuration parameters, and including its *"client_id"* and a list of the claims (information fields) that it would like to know.

Thus, the user's browser will perform a second HTTPS request (step 4), this time towards the identity authority. If the authority implements some kind of permanent session management that allows it to recognize the user automatically and securely, then the authority could actually skip the following steps and

just provide a valid *"authorization code"* to the user's browser, redirecting it back to the relying party and making the login process entirely transparent; the user would not even see the authority's website. By providing an authorization code to the user, the identity authority is effectively authorizing the login (step 5, again mediated in practice through the user's browser).

However, there are two separate checks that needs to be made by the authority between steps 4 and 5. First of all, it must perform the actual authorization check; if it cannot recognize the user via an existing session, it must ask for the user's authentication credentials and verify them against its locally stored database. Then, if this is the first time that the user logs into that specific relying party, or if the list of claims requested by that relying party has changed, the authority has to show the user a list of the claims that would be shared with the relying party, and to ask for consent; according to European laws, the user has to be able to provide or deny consent separately for each claim. Additionally, id4me provides mechanisms for parties to communicate to the users the reason what any claims are actually needed for. These two requests, for authentication and for consent, could appear on two separate intermediate screens, or even combined on a single one, hosted by the identity authority.

Whenever the authority decides to authorize the login, it should also perform an asynchronous callback to the identity agent that manages the identifier (not shown in the diagram), notifying the login. This step is not strictly required, but allows the agent to maintain and show a history of the logins, and to track potential improper uses.

After the user receives an authorization code and gets redirected from the authority to the relying party, the relying party has to validate the authorization code by establishing a direct connection (not shown in the diagram) to the identity authority's *"token endpoint"* URL, as provided in the authority's configuration. There, the relying party authorizes itself by presenting its *"client_secret"*, and exchanges the authorization code (if valid) for a couple of tokens – the *"identity token"* and the *"access token"*.

The *"identity token"* is the actual proof of login; it includes the user's identifier, as well as information on the authentication process and an expiration date, and it is digitally signed by the identity authority; it can be directly used by the relying party to identify the user throughout its entire platform. The *"access token"* can be optionally used whenever the relying party does not just want to get the user authorized, but also wants to know the content of some of his claims; it includes a list of claims whose sharing was agreed by the user for that specific relying party, and thus authorizes the relying party to retrieve them from the identity agent.

After receiving and validating the identity token, the relying party should also perform a callback to the identity agent that manages the identifier (not shown in the diagram), notifying the login. This allows the agent to match the two callbacks and verify that everything is correct, or take notice of an issue and of a potential security breach if the two callbacks do not match.

In general, if this is the first time that the specific relying party encounters the specific identity agent, it first has to discover his configuration and endpoint URLs by applying the already-mentioned OpenID Discovery standard process.

In the optional following phase (step 6), the relying party, after verifying the validity and the signature of the identity token received by the authority, will connect to the identity agent's *"userinfo endpoint"* URL. To be more precise, as per the OpenID Connect standard, first the client has to connect with the identity authority's *"userinfo endpoint"* URL (not shown in the diagram), which will use the *"distributed claims"* mechanism described in the standard to redirect the relying party to the *"userinfo endpoint"* of the appropriate identity agent.

The connection to the userinfo endpoint does not require the relying party to be previously known or registered at the identity agent, because the *"access token"* is a self-describing bearer token, and the simple

possession of it entitles the bearer to access the claims. However, the identity agent must verify that the token is correctly signed by the appropriate identity authority, and to this extent it will use the authority's public keys (retrieving them from the location pointed out in the authority's well known configuration file, if necessary) and then use them to verify the signature of the token.

In response (step 7), the relying party will receive the content of the requested claims, which could then be stored in the local profile for the user, creating it if this is the first login ever by that identifier into that relying party.

In line of principle, all DNS queries must be secured with DNSSEC, and HTTPS connections must use DANE to validate the certificate used by the web service that responds to them, requiring all identity authorities and agents to publish and maintain TLSA records for the involved hostnames. However, this requirement may be initially skipped for experimental deployments.

As a technical reference, at the end of the document you will find a more complex, two-parts diagram representing in detail all the steps described above. Please note that the diagram describes two separate requests by the user to provide first the password and secondly the consent on claims sharing, but they could be conflated into a single one.

A COMPARISON WITH OPENID CONNECT

As described above, the entire login flow, starting from step 3 in the diagram, is compliant with OpenID Connect; id4me only adds the DNS-based discovery mechanism in front of it.

However, there are several other extensions and specializations that the id4me standard introduces in respect to OpenID Connect; we list them below for convenience.

1. OpenID Connect only supports a single identity provider (centralized source of identifiers and server for authorizations); multiple parties can deploy the standard, but the identifiers released by one of them will not interoperate with other identity providers; a relying party must add separate support for each OpenID Connect deployment, leading to cumbersome user experiences in which the user first has to pick whether he will log in with Google or with Facebook or with another OpenID provider, and then can proceed. id4me, on the other hand, supports multiple identity providers transparently; given an identifier, it is possible to discover which identity authority is handling it through a DNS query, and thus contact the correct server for authorization. All id4me identifiers, no matter which authority released them, can be used as part of a single federated identity system.

2. The flexibility of OpenID Connect has been leveraged to run the different standard endpoints of the identity provider at different entities: the *"identity authority"* and the *"identity agent"*, which also naturally mirror the traditional subdivision of roles in the domain name industry. More precisely, the identity authority fulfills all the responsibilities that the OpenID Connect standard attributes to the identity provider, except managing the actual user claim values; to delegate this role to the agent while keeping compatibility with OpenID Connect, id4me makes use of the OpenID Connect mechanism of so-called *"distributed claims"* and the identity agent plays the role of a claims provider.

3. As id4me is a federated system, it becomes much more important to establish from the start a common ontology and standard naming and meanings for all the claims that could be attributed to a user. To this purpose, id4me aims to define a much longer list of standard claim names, covering many possible attributes that can be useful to different industries and in different use cases, ranging from flight seat preferences to social network handles. However, identity authority and agents are only required to support the basic set of claims from OpenID Connect, and users are not required to

provide any claim – an id4me identifier could be associated to no claims whatsoever, and just be used as an empty and anonymous login handle. If the users and actors want to exchange further claims, though, they are required to use the additional claim names that id4me will standardize.

4. id4me introduces the option of callbacks from the identity authority and from the relying party to the identity agent. If supported, they allow the agent to show a history of logins and to track potential improper uses.

5. id4me adds an explicit mechanism to handle the request and provision of granular consent by the user before any information is shared with a relying party, complying with the principles of privacy laws in Europe and in many other countries. Also, requesting parties can (and should) provide the reasons why they need some personal data, so that the user can perform a real informed consent. While OpenID Connect also mentions that user consent should be obtained, there is no explicit mechanism in the standard to transmit this consent to other parties.

6. The OpenID Connect standard mandates the use of HTTPS, but does not mandate the use of DNSSEC and DANE to secure that connections cannot be intercepted and diverted via DNS-based attacks – this is an addition in id4me.

DRAFT PUBLIC STANDARDS

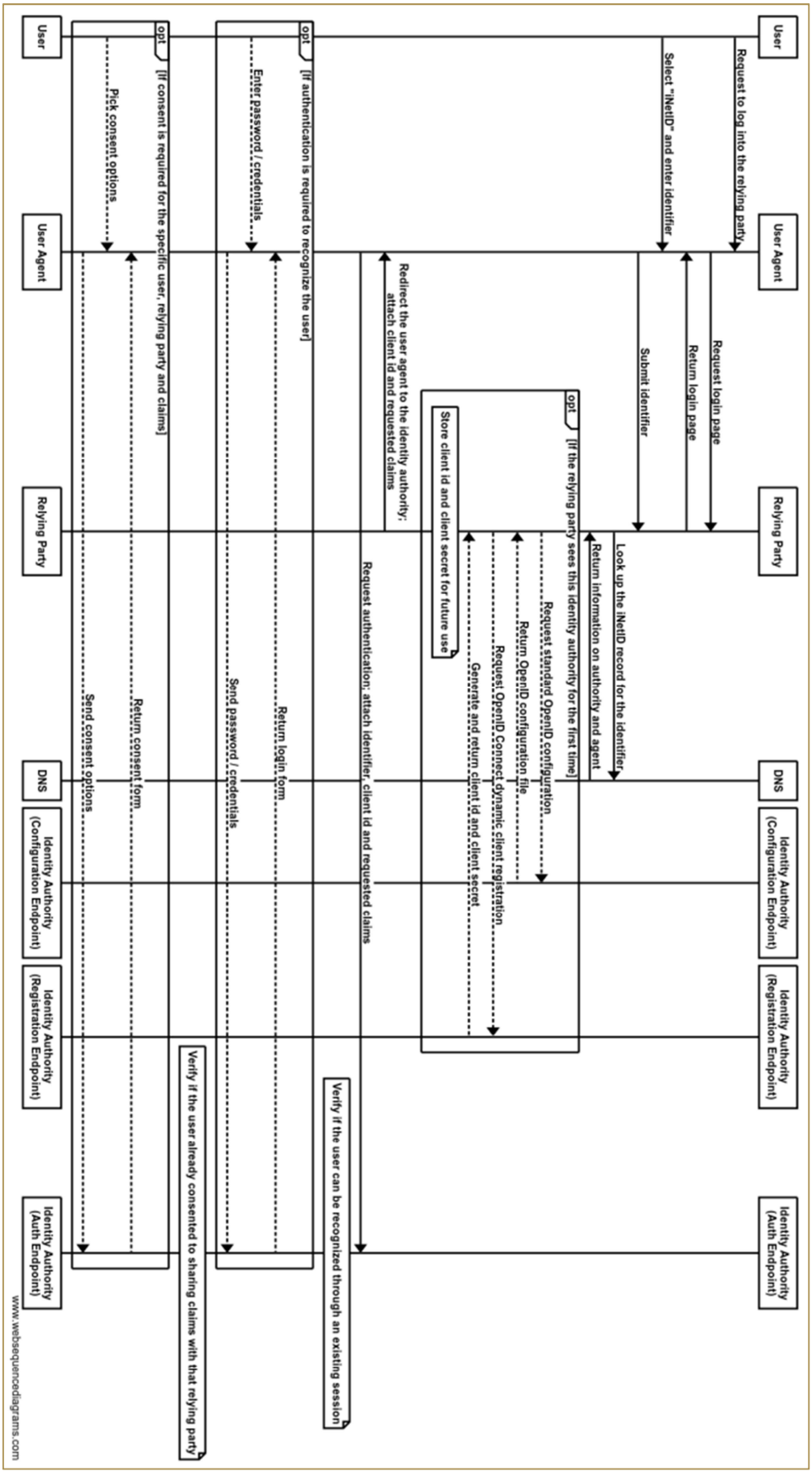
The intent of the id4me project is to provide a standard for a "public identity infrastructure" that can be widely used over the Internet, without entry barriers and intellectual property restrictions. We believe that the Internet needs an identity management system which is open, federated and not controlled by a single company – so all our specifications are public.

For the time being, we have submitted two independent Internet drafts to the IETF, so that they are archived and publicly accessible:

1. A general description of the architecture of the system: "An architecture for a public identity infrastructure based on DNS and OpenID Connect", https://datatracker.ietf.org/doc/draft-bertola-dns-openid-pidi-architecture/?include_text=1
2. A specific standard for the newly created alternative discovery process based on the DNS: "OpenID Connect DNS-based discovery", https://datatracker.ietf.org/doc/draft-sanz-openid-dns-discovery/?include_text=1

We welcome discussion and contributions to these drafts, and we are ready to work within the IETF, the OpenID Foundation and other relevant standardization organizations to ensure that id4me specifications are as public and shared as possible.

Complete id4me login flow – Part 1



Complete id4me login flow – Part 2