YADIS – Yet Another Decentralized Identity Interoperability System

The YADIS Project
www.yadis.org

1 Background

In early 2005, NetMesh published the Light-Weight Digital Identity (LID) specification for decentralized, URL-based personal digital identities. Shortly thereafter, Six Apart published the OpenID specification for authenticated blog comments using blog URLs as identifiers. (see http://lid.netmesh.org/ and http://openid.net/) These two personal digital identity systems are currently being used by well over fifteen-million users world-wide.

The lead developers of these two initiatives (Brad Fitzpatrick and David Recordon of Livejournal/Six Apart, and Johannes Ernst of NetMesh) quickly realized the complementary nature of their technologies. Over a few weeks in summer 2005, they developed a design to make LID and OpenID interoperable, and to leverage each protocol's most compelling features with each other.

Working on this, it became clear very quickly that the resulting interoperability architecture was much more broadly applicable. In our view, it promises to be a good foundation for decentralized, bottom-up interoperability of a whole range of personal digital identity and related technologies, without requiring complex technology, such as SOAP or WS-*. Due to its simplicity and openness, we hope that it will be useful for many projects who need identification, authentication, authorization and related capabilities.

This document describes the base YADIS protocol, and outlines how to use it together with LID and OpenID. For how to get involved, see the last section of this document. This document is largely still a work in progress, proposing how different existing identity systems can work together; feedback is welcomed. The YADIS codename is also not designed to be user facing and is expected to be changed as this project further progresses.

2 Goals

The YADIS goals are:

- to further broaden the applicability and feature set of OpenID, LID, and of other personal digital identity technologies: not by creating more fully-featured stovepipe technologies, not by expecting the world to conform to specs under control of a single vendor, but by creating an interoperable foundation around which many can innovate.
- to reduce the fragmentation and to improve the interoperability of today's digital identity technologies; this includes to help reduce the number of passwords the typical Internet user has to manage today.
- to provide a unified experience for users who wish to assert their personal digital identity on the Internet, regardless of the underlying technical plumbing.
- to contribute a multi-vendor, multi-technology personal identity foundation for Web 2.0.
- to make identification and authentication easier on the Internet, without compromising privacy.
- to follow an open, meritocratic process for doing so, e.g. by following common open source development practices.

- to allow and foster innovation and competition within the personal digital identity market.
- to provide a foundation that current personal digital identity systems can build upon as to not discard their previous work or users.

YADIS' initial focus is to empower the individual user with user-centric personal digital identity, and not so much to serve the needs of enterprises for, say, enforcing compliance with government regulations. While there are successful uses of the described technologies in enterprises already, we realize that more work needs to be done to address additional enterprise requirements. If you have specific expertise in this area, we very much appreciate your input. We do however see the ability for corporations to integrate their existing authentication mechanisms with other YADIS enabled services providing their users with SSO abilities outside of their own architecture.

3 Architectural Assumptions

We have found it is easiest to understand an architecture if it explicitly lists its assumptions; so here you are.

3.1 Fully decentralized, and no one point of control

The Internet is a big place, in which centralized control of any kind is very difficult or impossible. While the technology is quite simple in case of ICANN, for example, its checkered history can serve as the proverbial Exhibit A for this conjecture. On the reverse, where certain companies have been successful in establishing a technical or organizational choke hold on the Internet, innovation these days tends to route around it.

We believe any digital identity architecture must take these lessons learned and not introduce any additional centralized bottleneck, whether of technical or of a governance nature, if there is a way of avoiding it. LID and OpenID have both demonstrated that this is possible.

3.2 Let many (interoperable) flowers bloom

We firmly believe that innovation is a good thing and want to enable people around the world to innovate upon this interoperable infrastructure, instead of declaring we have all the answers already and making ourselves the bottleneck for innovation.

We believe that digital identity technologies today are only the tip of the iceberg, and, while growing, the market is only embryonic today. For example, so far we have seen little public debate on the merits and issues of personal digital identity technologies; we can bet that such a debate will occur and that it will have substantial impact on what technologies will be broadly accepted and which won't. So we feel it is paramount to let people with good ideas plant new flowers and let those flowers bloom. While there may be a point that a single identity system reaches critical mass on its own, providing a foundation for interoperability will only decrease the amount of time before the general public understands and uses digital identity systems.

YADIS supports the introduction of new capabilities by anybody, while providing enough of a foundation to not break interoperability.

3.3 URLs as identifiers

It is a natural expectation of (non-technical) users that that they can employ search engines such as Google to find people, e.g. by searching for the first and last name of the person, company name etc.

Today, search engines most likely find a person's blog or home page (if they have one) first. Therefore, we believe using URLs (such as blog or homepage URLs) as identifiers for people is A Good Thing.

We believe it is possible to extend the YADIS architecture to work with non-URL identifiers as well; we may attempt this in the future to be able to integrate with other, non-URL-based personal digital identity technologies. We do however feel that reaching critical mass will be obtained first upon this assumption.

3.4 REST-ful and easy to use for developers

Digital identity technologies can only live up to their full potential if it is really easy for developers of all kinds – from hobbyists running, say, a PTA's discussion board over open source projects to large commercial protocols – to identity-enable their projects. Thus it is paramount to make and keep YADIS as simple as possible.

LID and OpenID implementations exist already in many common programming environments (e.g. PHP, Perl, Java) and can be incorporated easily into existing applications. The same will shortly be true for YADIS. For example, YADIS does not require SOAP or a web services stack at all.

4 User scenarios

Currently, YADIS defines only one scenario performed by the end user. Protocols that take advantage of YADIS support many additional scenarios that YADIS users can take advantage of by virtue of their integration with YADIS, but which do not need to be defined by YADIS.

4.1 Scenario: Authentication at website

1. The user encounters a website (called an "Identity Consumer", see terminology section below) that allows or requires the user to present a YADIS-enabled identifier (e.g. LID or OpenID URL). The user notices this because the website displays a text field titled "My URL" instead of the classical login box.
2. The user enters their YADIS-enabled URL into the "My URL" text field and clicks submit.
3. The identity consumer determines (using the YADIS Capability Discovery Protocol described below) which YADIS-compatible identity protocol to use with the given identifier, and authenticates the user according to the rules of that identity protocol.
4. Once the user is authenticated, the identity consumer may retrieve the user's profile, using the entered YADIS URL (e.g. by printing "Hi Mr. John P User!" on the top of the page)

5 YADIS Protocol

5.1 Capability Discovery Protocol

Once the user has entered their YADIS URL, such as a LID URL or OpenID URL (called **MYID** from here) into thye "My URL" text field at a website (called Identity Consumer), the Identity Consumer must first discover which capabilities the entered YADIS URL supports, such as whether it is a LID or OpenID URL, and which authentication methods it supports. To do that, the identity consumer performs the following HTTP operation:

```
GET MYID?meta=capabilities
```

This produces exactly one of the following responses:

1. A document of MIME type "application/x-meta-identity". This conveys to the identity consumer what identity protocols the YADIS URL supports (see below). This response is produced in case the entered YADIS URL is a LID URL, for example.
2. An HTML document (MIME type text/html) that contains the following links in the HTML HEAD section. This occurs in case the YADIS URL refers to a regular HTML web page.

- `<link rel="identity.server"   href="`**`IDENTITY.SERVER`**`” >  (mandatory)`
- `<link rel="identity.delegate" href=”`**`IDENTITY.DELEGATE`**`"> (optional)`

3. Any other response. This indicates that the entered URL does not support any identity protocol and is thus invalid from a YADIS perspective.

If the obtained response is #2, and **IDENTITY.SERVER** is a properly formatted URL, the identity consumer performs the following query:

```
GET IDENTITY.SERVER?meta=capabilities
```

This query must result in a document of MIME type "application/x-meta-identity".

The document of MIME type "application/x-meta-identity" lists the capabilities of the YADIS URL; the Identity Consumer then uses this information to initiate an authentication according to the identity protocol it chooses from the obtained choices.

For a description of the document format of MIME type "application/x-meta-identity”, see below.

5.2 Authentication

Authentication is delegated by YADIS to the respective identity systems. Which authentication methods are supported by a given YADIS URL can be determined through the YADIS Capability Discovery Protocol (see above).

5.3 Profile data exchange

LID defines a REST-ful protocol that allows the structured query of “profile” information independent of the schema in which that information is expressed. The returned information typically depends on the identity of the client that performed the query, thereby supporting a variety of privacy and security policies. (See http://lid.netmesh.org/wiki/LID_2.0_Traversal_Profile.) OpenID defines this subject as out of scope. The YADIS protocol for Profile data exchange is an adaptation of the LID protocol.

Basis for YADIS profile data exchange is a REST-ful URL (called “Constructed LID” from here) that supports the query of information. Depending on the capabilities of the YADIS URL (that identifies the person whose profile shall be queried), the Constructed LID is determined as follows:

- If the YADIS URL is a LID URL (i.e. it understands the "meta=capabilities" query without indirection, and indicates that it supports LID in the response), the YADIS URL already is the Constructed LID.
- If the YADIS URL is an OpenID URL (i.e. it responds to the "meta=capabilities" query with HTML that contains a valid **IDENTITY.SERVER** field), the Constructed LID is

```
IDENTITY.SERVER?id=MYID
```

- where **MYID** is the field **IDENTITY.DELEGATE** if given, or the YADIS URL otherwise. According to the rules of URL syntax, the question mark must be replaced by an ampersand if **IDENTITY.SERVER** contains a question market already.

    Note: Alternatively, we could use a %s syntax (“printf”) or a {MYURL} syntax (a la A9 queries). I think the above is easier to understand, however.

Given this algorithm for determining the Constructed LID, the profile query is performed as follows: If Alice with Constructed LID **AliceURL** wishes to obtain profile information from Bob with Constructed LID **BobURL**, she performs the following operation:

```
BobURL?xpath=XPATH&lid=AliceURL
```

where **XPATH** is an XPath according to the LID specification. If **BobURL** already includes a question market, **XPATH** must be appended with an ampersand instead of a question mark. If **AliceURL** contains a question mark, this question market must be properly escaped, and NOT converted to an ampersand.

As in case of LID, The lid parameter is optional. If the lid parameter is not given, Bob has no way of identifying Alice, and thus will return the response for anonymous users.

If the id parameter is given, **BobURL** will usually check whether the request indeed came from **AliceURL**. Depending on the capabilities of **AliceURL** (determined according to the Capability Discovery protocol above) and **BobURL**'s own capabilities, **BobURL** will use either OpenID or LID authentication.

6 Formats

6.1 application/x-meta-identity

Note: we considered providing both a text and an XML-based format, in order to make it as easy as possible for clients to use capability information available through YADIS URLs. However, we came to the conclusion that requiring both a text and an XML-based format provides only small additional value, so we only require the text-based format at this time.

Here is an example response:

```
capability: http://lid.netmesh.org/
version: 1.0,2.0

capability: http://lid.netmesh.org/sso
version: 1.0,2.0

capability: http://openid.net
version: 1.0

capability: http://example.com/new/innovative/capability
version: 1.0beta2,1.0,2.3,9.0-12
foo: bar
...
```

The response consists of a sequence of capability descriptions that are separated from each other by empty lines. The sequence of these capability descriptions is not significant.

Each capability description consists of two mandatory lines, and an unlimited number of optional lines (we encourage implementors to be brief). Each line is a name-value pair, which is separated by a colon and a single space after the colon.

The two mandatory lines are, in this sequence:

```
capability: CCC
version: VVV,VVV,VVV
```

The capability line identifies, as the value (called **CCC** above) the name of the capability. The name of a capability must be a valid URL that, through the contained host name, must guaranteed to be collision-free by the creator of the capability name.

The version line lists one or more version identifiers (called **VVV** above), separated by a comma. The version identifier is an uninterpreted string that must not contain a comma. For the purposes of YADIS, version identifiers are not interpreted. There is no assumption whatsoever that if, for example, a version 5.2.1 is supported, a version 5.2 is also supported without being listed. All versions of the capability that are supported must be listed individually.

All other lines in a capability descriptor are undefined by YADIS and can be defined and used by a capability descriptor as they wish. This allows a capability to indicate things that are specific to the capability. (For example, an iris authentication capability may want to specify that it requires a camera of at least such and such of a resolution)

7 Impact on LID and OpenID

- LID will need to support the YADIS Capability Discovery Protocol as part of MinimumLID.
- OpenID will need to support the YADIS Capability Discovery Protocol.
- OpenID will use the profile data exchange protocol defined in this document
- The LID protocol will be updated to support the profile data exchange protocol defined in this document, and some rules with respect to question marks in LID URLs will be relaxed to be able to support Constructed LIDs.
- Going forward, LID implementations should recognize when a non-LID authentication is requested and respond appropriately. They are encouraged to support OpenID authentication as well.
- Going forward, OpenID implementations should recognize when non-OpenID authentication is requested and respond appropriately. They are encouraged to support LID authentication as well.

The following capability identifiers have been defined for OpenID and LID so far:

Identifier	Capability
http://openid.net/	OpenID SSO
http://lid.netmesh.org/sso	LID 2.0 SSO
Http://lid.netmesh.org/traversal	LID 2.0 Traversal Profile (for xpath queries)
Http://lid.netmesh.org/traversal/vcard	LID 2.0 Profile for Contact Information Management (“VCARD Profile info”)

8 Examples

8.1 Log on at an OpenID site (non-delegated case)

1. Action: User enters URL of Identity Consumer into browser, browser requests page from Identity Consumer
   Response: Identity Consumer's page is displayed in browser
2. Action: User enters their YADIS URL (**MyURL**, which in this case is an OpenID URL), into the login field, clicks “submit”
   Response: Control gets handed to the Identity Consumer
3. Action: Identity Consumer performs a "meta=capabilities" request on **MyURL**.
   Response: HTML page is returned, which contains **identity.server** field. (This is an OpenID URL)
4. Action: Identity Consumer performs a "meta=capabilities" request on the URL obtained from the **IDENTITY.SERVER** field.
   Response: Document of type "application/x-meta-identity" is returned, which lists **http://openid.net/** as one of the supported capabilities.
5. Action: Identity Consumer interacts with the **IDENTITY.SERVER** (which now has been identified as an OpenID server) as per the OpenID protocol.
   Response: The browser session has been authenticated, the Identity Consumer shows the logged-in page, and control returns to the user.

8.2 Log on at a LID site (non-delegated case)

1. Action: User enters URL of Identity Consumer into browser, browser requests page from Identity Consumer
   Response: Identity Consumer's page is displayed in browser
2. Action: User enters their YADIS URL (**MyURL**, which in this case is a LID URL), into the login field, clicks "submit"
   Response: Control gets handed to the Identity Consumer
3. Action: Identity Consumer performs a "meta=capabilities" request on **MyURL**.
   Response: Document of type "application/x-meta-identity" is returned, which lists **http://lid.netmesh.org/** as one of the supported capabilities.
4. Action: Identity Consumer interacts with the **MyURL** (which now has been identified as an LID URL) as per the LID protocol.
   Response: The browser session has been authenticated, the Identity Consumer shows the logged-in page, and control returns to the user

9 Terminology

We distinguish between end-user and developer terminology. Developer terminology is intended to be technically precise and unambiguous, while end-user terminology is intended to be easy to understand and to use by non-technical users.

9.1 Developer Terminology

Term	Meaning
YADIS URL	A URL that is used to identify an indivual. A YADIS URL may or may not directly support YADIS Capability Discovery (see Capability Discovery protocol described above). See REST-ful YADIS URL, Hosted YADIS URL, and Delegating YADIS URL.
Identity Consumer	A server, website, application, or URL that can take advantage of YADIS URLs (and/or information accessible through or with them) provided by clients. The Identity Consumer typically discovers the capabilities of any provided YADIS URL according to the YADIS Capability Discovery protocol defined above, and modifies its own behavior accordingly.
Identity Server	A server that hosts one or more Hosted YADIS URLs. The Identity Server may or may not be located at the same URL as the YADIS URL.
REST-ful YADIS URL	A YADIS URL that supports the `meta=capabilities` query for capability discovery. LID URLs are REST-fule YADIS URL.
Hosted YADIS URL	A YADIS URL that does not support the `meta=capabilities` query itself, but which specifies an Identity Server, but no delegate YADIS URL in the HTML HEAD field of a returned HTML document. OpenID URLs are either Hosted YADIS URLs, or Delegating YADIS URLs.

Term	Meaning
Delegating YADIS URL	A YADIS URL that does not support the `meta=capabilities` query itself, but which specifies an Identity Server, and a delegate YADIS URL in the HTML HEAD field of a returned HTML document. OpenID URLs are either Hosted YADIS URLs, or Delegating YADIS URLs.

9.2 End-user terminology

Term	Meaning
My URL	A URL the user uses to identify an individual, such as themselves or somebody else. Same as YADIS URL.

10For more information

Please visit http://yadis.org/ for additional information, including contact information. You can also sign up to the YADIS e-mail list there.