

# OpenID Identity Discovery with XRI and XRDS

Drummond Reed  
Cordance Corp.

3020 Issaquah-Pinelake RDF #74  
Sammamish WA 98075  
+1.206.618.8530

drummond.reed@cordance.net

Les Chasen  
Neustar, Inc.

46000 Center Oak Plaza  
Sterling VA 20166  
+1.571.434.5474

les.chasen@neustar.biz

William Tan  
Neustar, Inc.

46000 Center Oak Plaza  
Sterling VA 20166  
+1.571.434.5400

william.tan@neustar.biz

## ABSTRACT

The work examines the identity discovery problems that needed to be addressed by the OpenID 2.0 protocol in order to enable a user-centric Internet identity layer. The paper illustrates how the OASIS XRI and XRDS specifications were applied to help solve these identity discovery challenges. The work also considers interoperable identity discovery for other Internet identity frameworks such as SAML, Information Cards, and the Higgins Project, and recommends future work.

## Categories and Subject Descriptors

C.2.4 [Computer-Communications Networks]: Distributed systems – *distributed databases*. D.4.6 [Operating Systems]: Security and protection. H.5.2 [Information Interfaces and Presentation]: User Interfaces – *user-centered design*. H.5.4 [Information Interfaces and Presentation]: Hypertext/Hypermedia – *architectures, navigation, user issues*. K.6.5 [Management of Computing and Information Systems]: Security and protection – *authentication, unauthorized access*. K.8.3 [Management and Maintenance]: Security and protection.

## General Terms

Design, Security, Human Factors, Standardization, Verification.

## Keywords

User-centric identity, identity discovery, XRI, Extensible Resource Identifier, identifier, resolution, XRDS, Extensible Resource Descriptor Sequence, OpenID, Yadis, SAML, information card, i-card, Higgins Project.

## 1. INTRODUCTION

In enterprise identity management frameworks, the context of an identity being asserted is generally known, or can be discovered directly via mechanisms specified in the framework. But when the context is the Internet as a whole, this approach is no longer viable.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*IDtrust '08*, March 4-6, 2008, Gaithersburg, MD, USA.

Copyright 2008 ACM 978-1-60558-066-1 ...\$5.00.

Internet identity management frameworks such as SAML [1], Shibboleth [2], Liberty Alliance [3], Information Cards [4], Higgins [5], and OpenID [6] all must deal with the “identity discovery problem”.

Of these frameworks, the one most distinguished by how it handles discovery is OpenID. Due to its origin as a means of combating blog spam, the premise of OpenID is that a user may assert their identity by using their own identifier—for example the URL of their blog, home page, social network profile, or any other web resource the user controls. An OpenID relying party (RP) can then discover from that identifier the user’s OpenID provider (OP) and initiate OpenID authentication.

This approach is unquestionably “user-centric” because it gives the user complete control over the identifier they use and therefore the set of OPs that an RP may query. (The issue of whether an RP will accept authentication from the user’s selected OP selected is out-of-scope for the OpenID protocol.) However even with this very direct approach, there were still identity discovery challenges that needed to be solved in the steps up from OpenID Authentication 1.0 [7] in 2005 to OpenID Authentication 2.0 [8] in 2007. In this paper we explore these challenges and show how the OASIS XRI and XRDS specifications were employed to overcome them. We also look at how XRI and XRDS are being used by other Internet identity management frameworks, and conclude by suggesting future work.

## 2. IDENTITY DISCOVERY CHALLENGES IN OPENID

The basic OpenID authentication scenario is that a user logs into an RP site by entering their OpenID identifier rather than a conventional RP-specific username. The RP then resolves the OpenID identifier to discover the user’s OP. In OpenID 1.0, the only supported identifier was a URL, and resolution was either directly to the user’s OpenID server, or to an HTML page containing a meta tag with the URL of the OpenID server.

Early OpenID usage raised the following challenges.

### 2.1 Service Description

As soon as OpenID began to evolve into V1.1 [9], there arose the need not just to discover the user’s OpenID service endpoint, but to describe its capabilities (if nothing more than whether it supported OpenID V1.0, V1.1, or both). In addition, the OpenID protocol was designed to be extensible so it can include transfer other useful identity information, such as attributes or authorizations. For example, the Simple Registration extension

(SREG) [10] was developed to automatically transfer the most frequently-request attributes needed for site registration. Thus OPs needed the ability to describe whether they supported SREG or other OpenID extensions.

In addition, OpenID 1.0 was not the only URL-based authentication protocol; it was actually preceded by Lightweight Identity (LID) [11]. Users who wanted their OpenID identifier to work with either protocol needed a way to describe that it supported both capabilities.

These requirements were all difficult to fulfill with HTML link tags; they called for a more generalized, standardized service description mechanism.

## 2.2 OpenID Recycling

In conventional username/password identity schemes, if a user account is abandoned, the RP can delete the credential and reassign the username to another user. With OpenID, this option is not available since the RP does not assign the user's identifier—that choice is up to the user.

This introduces the “OpenID recycling problem”: if a user loses control of their OpenID identifier (for example, if the domain name registration for their URL expires), a new registrant of that URL can gain access to the same private resources as the previous registrant because the new registrant can point the URL to their own choice of OP.

This problem still exists even if the OpenID identifier is assigned by a service provider who can control reassignment, because service providers with large namespaces cannot afford to permanently lock up names within that namespace.

## 2.3 Resolution Integrity and Trust

From a security standpoint, the weakest link in the OpenID protocol is the discovery stage. This is not only where an untrustworthy RP will focus a phishing attack, but also where resolution of a standard HTTP URL may be hijacked. The use of an HTTPS URL provides significant (but not complete) protection from these attacks.

However OpenID cannot make HTTPS resolution the default due to implementation and usability challenges—it is often not feasible for an individual to obtain an SSL certificate, and outsourced Web hosting services do not always support HTTPS infrastructure. So although HTTPS is recommended, a user must explicitly have it provisioned, then request to use it at every OpenID RP by typing their fully qualified HTTPS URL (not a shortcut like “username.provider.com”).

## 2.4 Privacy and Non-Correlation

Another widely recognized privacy implication of the OpenID 1.x protocol is that it required users to share the same OpenID identifier (or one of a small set they control) with every site. While in some cases this is desirable for cross-site attribute and reputation management, in many other cases such correlation keys are neither necessary nor desirable. In fact other Internet identity frameworks including Liberty Alliance and Information Cards have gone out of their way to avoid introducing such keys.

## 2.5 Extensibility

Lastly, OpenID architects and users recognized that if OpenID is to develop into a generalized framework for user-centric Internet identity, it must be extensible to a wide variety of services that may be associated with an OpenID identifier. These services should all share a common, interoperable description format, including the ability for services—with the user's permission—to communicate and interact with each other on the user's behalf.

Again, these were not the purposes for which HTML header link tags were designed. Clearly a more robust but still lightweight solution is needed.

## 3. ADDRESSING THESE CHALLENGES WITH XRI AND XRDS

At the same time OpenID 1.1 was evolving, the XRI Resolution 2.0 specification [12] was under development at the XRI Technical Committee at OASIS. The previous 1.0 version was a generalized identity discovery framework developed for use with XRIs (Extensible Resource Identifiers)—abstract identifiers designed for network-, domain-, and application-independent resource identification. XRIs essentially serve the same function for URIs (and any other form of network address) that domain names serve for IP addresses.

However because XRI resolution was based on HTTP(S) and XML, there was nothing to prevent the resolution protocol from being generalized to work with URLs as well as XRIs. This became a key design goal of XRI Resolution 2.0, and led to the following features being incorporated into identity discovery for OpenID Authentication 2.0.

### 3.1 Service Endpoint Discovery with XRDS Documents

XRI infrastructure uses the XRDS (Extensible Resource Descriptor Sequence) format for discovery documents. By contrast with DNS, which describes resources using binary resource record types, XRDS documents are a simple, easily extensible XML format for describing the capabilities of any XRI-, IRI-, or URI-identified resource in a manner that can be consumed by any XML-aware application (or non-XRI aware browsers via a proxy resolver).

Figure 1 is an example of an XRDS document describing the resource identified by an XRI for the user of a telephone number, `xri://(tel:+1-201-555-0123)*home`. (This particular XRI illustrates the ability of XRI syntax to include identifiers from other namespaces, essentially acting as an “XML for identifiers”).

Figure 1. An example XRDS document

```
<XRDS xmlns="xri://$xrds" ref="xri://(tel:+1-201-555-0123)*home">
  <XRD xmlns="xri://$xrd*($v*2.0)" version="2.0">
    <Query>*home</Query>
    <Status code="100"/>
    <ServerStatus code="100"/>
    <Expires>2005-05-30T09:30:10Z</Expires>
    <ProviderID>xri://(tel:+1-201-555-0123)</ProviderID>
    <LocalID>*residence</LocalID>
    <EquivID>https://example.com/example/resource/</EquivID>
    <CanonicalID>xri://(tel:+1-201-555-0123)!1234</CanonicalID>
    <CanonicalEquivID>
      xri://=!4a76.c2f7.9033.78bd
    </CanonicalEquivID>
    <Service>
      <ProviderID>
        xri://(tel:+1-201-555-0123)!1234
      </ProviderID>
      <Type>xri://$res*auth*($v*2.0)</Type>
      <MediaType>application/xrds+xml</MediaType>
      <URI priority="10">http://resolve.example.com</URI>
      <URI priority="15">http://resolve2.example.com</URI>
    </Service>
    <Service>
      <ProviderID>
        xri://(tel:+1-201-555-0123)!1234
      </ProviderID>
      <Type>xri://$res*auth*($v*2.0)</Type>
      <MediaType>application/xrds+xml;https=true</MediaType>
      <URI>https://resolve.example.com</URI>
    </Service>
    <Service>
      <Type>http://openid.net/signon/1.0</Type>
      <URI>http://example.com/openid/</URI>
      <LocalID>https://example.com/example/resource/</LocalID>
    </Service>
    <Service>
      <Type match="null" />
      <Path select="true">/media/pictures</Path>
      <MediaType select="true">image/jpeg</MediaType>
      <URI append="path" >http://pictures.example.com</URI>
    </Service>
  </XRD>
</XRDS>
```

By requesting an XRDS document (MIME type `application/xrds+xml`) when resolving an OpenID identifier, an OpenID RP can easily determine the OpenID service endpoints associated with a user's identifier. Each service endpoint, described by the `<xrd:Service>` element, can be identified using one or more `<xrd:Type>` elements. This element accepts a URI, IRI, or XRI to identify the service type. This makes the XRDS format extensible by any specification or service provider without the need for a central type registry.

In addition to advertising the service types it supports, each service endpoint can include a set of URIs representing concrete network endpoints at which this service is available. Redundant network endpoints can be expressed by using more than one `<xrd:URI>` element. Priority among multiple URIs for the same service endpoint (or multiple service endpoints of the same type) is expressed using a priority attribute with the same semantics as

in DNS [13]. Elements with equal priority are selected randomly, which can be used to achieve round robin behavior [14].

### 3.2 Preventing OpenID Recycling with Persistent XRI I-Numbers

The OpenID recycling problem reflects a generic issue in resource identification: the fact that semantic identifiers—the identifiers people find easiest to remember and use—are often the least persistent. The reason is the evolutionary nature of semantics—people constantly change names, addresses, and service providers. This runs directly contrary to identity management policies that depend on a persistent binding between an identifier and the resource it represents (user, device, application, domain, etc.)

This problem is minimized in enterprise contexts because identifier reassignment policies can be tightly enforced. Unfortunately such policies are not feasible at Internet scale. The

domain name secondary market, for example, exists for the very purpose of transferring domain name registrations.

One solution has been to create separate Internet registry and resolution infrastructure for persistent identifiers. The IETF URN (Uniform Resource Name) [15] and Handle [16], [17], [18] specifications were developed for this purpose. However because they lack the human usability of DNS, their adoption has largely been limited to digital artifact registries and DRM systems where identifier non-reassignability is an absolute requirement.

A primary motivation for development of the XRI specifications at OASIS was solving this usability problem by providing a uniform syntax and resolution protocol for *both* reassignable and persistent identifiers. In XRI parlance these known as *i-names* and *i-numbers*. XRI resolution makes it very efficient for each reassignable i-name to have a synonymous persistent i-number that is discovered in the same resolution call. For example, in Figure 1, the local i-name **\*home**, shown in the `<xrd:Query>` element is synonymous with the i-number **xri://=!4a76.c2f7.9033.78bd**, shown in the `<xrd:CanonicalEquivID>` element.

This architecture enables XRI registry infrastructure such as that implemented by XDI.org [19] to enforce policies requiring each i-name registration to have a synonymous i-number, and for i-numbers to never be reassigned, as in URN or Handle registries.

However, synonym assertions must be verified before they can be trusted. XRI Resolution 2.0 specifies two automated verification methods: a) confirming that the synonyms were assigned by the same XRI authority, or b) if they were assigned by different authorities, confirming that they are authorized synonyms by checking for the existence of an `<xrd:EquivID>` reference between the two XRDS documents.

The result is a deep structural solution to the OpenID recycling problem. Although OpenID Authentication 2.0 does not require XRIs, it specifies that when an XRI i-name is used as an OpenID identifier, after discovery the RP must use the synonymous canonical i-number as the user's claimed identifier, and that this synonym must be verified [20]. Since this i-number will never be reassigned, both the registrant and the RP are protected from future reassignment of the i-name. This also enables XRI registries to safely reassign i-names to new registrants by pairing them with a new persistent i-number.

It should be noted that for backwards compatibility, the OpenID Authentication 2.0 specification also supports the ability for an OpenID service provider to add a fragment to a URL in order to distinguish the current user of that URL from a previous or future user [21]. However this solution to OpenID recycling works only for service providers who strictly control their entire URL namespace. For other URLs, transfer of the domain will also transfer control of URL fragments. Furthermore, reassignment of the base URL to a new registrant terminates the ability of the previous registrant to assert that identity because a fragment cannot be resolved. XRI i-numbers do not have this limitation; they can continue to be used indefinitely without regard to the reassignment of any i-names with which they have previously been associated.

### 3.3 Automatic Trusted Resolution with XRI I-Names

XRI architecture includes two options for secure resolution. The first is to require HTTPS for each request in the resolution chain. For example, the i-name **@cordance\*drummond** requires two XRDS document requests: 1) query the @ registry for **\*cordance** (\* is assumed after the @), 2) query the **@cordance** registry for **\*drummond**. In HTTPS secure resolution, each resolution query must use an HTTPS service endpoint, or else resolution fails.

The second option is for each XRDS document in the resolution chain to include a signed SAML assertion that resolvers can verify via public key information discovered in the previous XRDS. The two options are not exclusive; indeed the specification recommends that SAML trusted resolution be used in conjunction with HTTPS trusted resolution to ensure confidentiality.

RPs using OpenID Authentication 2.0 can advantage of this capability by automatically resolving all XRIs using at least the HTTPS trusted resolution protocol. It is relatively easy for XRI authorities to comply with this requirement because XRI is an abstraction layer for URIs; thus an XRI resolution service endpoint only needs to be provisioned with one SSL certificate, no matter how many XRI identifiers it hosts. XDI.org, for example, mandates HTTPS support for the XRI global registry and resolution infrastructure it oversees [22].

The result is that unlike URLs, all XRI i-names used as OpenID identifiers can automatically default to secure resolution, without the need for a user to type a special prefix.

### 3.4 Anti-Correlation with Pairwise Identifiers

The premise of OpenID 1.x was that users would share one globally unique URL (or one of a presumably small set of URLs) with RPs. This stood in stark contrast to other Internet identity frameworks such as Liberty Alliance ID-WSF and Information Cards, which go to great lengths to use pairwise identifiers so they introduce no new correlation handles at the protocol level.

OpenID Authentication 2.0 addressed this issue by adding support for “directed identity”—a term for the use of pairwise-unique identifiers coined by Microsoft Chief Identity Architect Kim Cameron [23]. This was accomplished by adding the new service endpoint type “`http://specs.openid.net/auth/2.0/identifier_select`”. When a user enters an OpenID identifier resolving to a service endpoint of this type (typically by entering the URL or i-name of their OpenID provider, rather than their own OpenID identifier), the RP knows it must ask the OP for the user's identifier. The OP can then offer the user the choice of using one of their existing OpenID identifiers, or having the OP generate a pairwise-unique identifier for this specific relationship. In fact the user need not know or remember this identifier as the OP can store and automatically use it in future logins to the same RP.

This directed identity feature works with both URLs and XRIs, however by assigning XRI i-numbers in the OP's own XRI delegation space, OPs can take advantage of their persistence and security features discussed above.

### 3.5 Extensibility to New Services

Much of the market interest in OpenID lies in its larger potential to serve as a framework for many user-centric identity services, all keyed off a user's OpenID identifier(s). To do this, these services need to share a common discovery mechanism that enables different services to interoperate on the user's behalf.

An example is the new OAuth 1.0 protocol, released by the OAuth community in October 2007 [24]. OAuth might be called "OpenID for applications", i.e., it enables a user to delegate to a website or application the ability to access the user's private resources—without the user needing to reveal their actual credentials.

OAuth 1.0 assumes that OAuth providers and consumers are configured manually. However the OAuth community quickly recognized the need for automated discovery of OAuth service endpoint URIs and other configuration metadata. By December 2007 they had published the first draft of OAuth Discovery 1.0 [25]. This specification makes extensive use of XRDS architecture, and specifically the ability for it to be extended by: a) new service type URIs, IRIs, or XRI from any namespace, b) new XML elements and attributes from other XML namespaces, and c) new trust models based on existing XRDS elements such as `<xrd:ProviderID>` and `<xrd:LocalID>` and/or extension elements.

## 4. INTEROPERABILITY WITH OTHER INTERNET IDENTITY FRAMEWORKS

Although we have focused on the relevance of XRI and XRDS to OpenID discovery, these technologies are equally applicable to other Internet identity frameworks. In fact they may play a key interoperability role, as discussed in this section.

### 4.1 SAML

The OASIS SAML specifications include authentication flows very similar to OpenID except for the initial discovery steps [26]. So it is not surprising that they can be adapted to use the same XRDS discovery mechanism as OpenID 2.0. The only difference is the use of a SAML authentication service endpoint. This flow was demonstrated by Pat Patterson of Sun at Internet Identity Workshop in December 2006 [27].

This flow can be further enhanced to provide automated discovery of the SAML metadata [28] necessary to interact with the SAML service provider. By including an XRI as the value of the `<xrd:ProviderID>` element in the SAML authentication service endpoint, an RP can use XRI trusted resolution to resolve this identifier and obtain another XRDS with service endpoint(s) advertising the location of the service provider's SAML metadata documents (which should also be retrieved using HTTPS).

### 4.2 Information Cards

The information card architecture implemented by Microsoft CardSpace, the Higgins Project, and others takes a different approach to identity discovery. First an RP publishes a machine-readable policy description of the claims they require for authentication/authorization to a Web resource. When the user browses that page, their identity selector client reads the policy and presents the user with a choice of the information cards they

have (if any) that satisfy it. If the claims are not "self-issued", but come from a third-party identity provider ("managed"), the card itself contains the metadata necessary for the selector to send an authentication token to the provider and obtain a security token bearing the claims, which it then passes to the RP.

In this architecture, an RP does not need to discover a service endpoint for the identity provider directly; all interactions are handled through the selector client. This has clear privacy advantages. However one drawback is that it does not provide the RP with an addressable network endpoint for further discovery or interaction with the user. This has led to proposed "OpenID Information Cards" [29]—standard information cards issued by a user's OP and conveying a security token containing the user's authenticated OpenID identifier. RPs accepting OpenID information cards can then invoke OpenID 2.0 discovery to locate other identity services for the user.

## 4.3 The Higgins Project

The Higgins Project is a protocol-independent open-source Internet identity framework designed to integrate identity, profile, and relationship information across multiple heterogeneous systems. Started by Parity and including IBM, Novell, Oracle, and Google as contributors, Higgins achieves interoperability via three primary framework elements:

1. *The Higgins Data Model*—a uniform identity data model based on RDF and OWL [30].
2. *Context providers*—Higgins components that implement the Higgins data model to provide a common API for access to any identity data store, from an LDAP directory to an XML document [31].
3. *I-cards*—a consistent user interface metaphor for all identity interactions, regardless of the underlying protocols or token types. I-cards are essentially synonymous with "information cards", but broader in function because they include card types not defined by Microsoft [32].

In the Higgins data model, every identity subject in a context that exposes a Higgins API is addressable with the combination of a *ContextId* and a local *SubjectId*. For interoperability, Higgins required ContextIds to be in a form that enables automated discovery of Higgins context provider configuration metadata, while at the same time supporting the native identifier types that may be used across a very wide variety of Higgins contexts.

The Higgins Project was able to satisfy these requirements with the XRI/XRDS 2.0 framework. First, it lets them express ContextIds as filenames, URLs, or XRIs [33]. Second, it lets them define a small set of Higgins service endpoint types and extension elements for expressing Higgins context provider configuration metadata in XRDS documents [34]. The result is that any Higgins component can resolve the ContextId portion of a Higgins address, discover the Higgins configuration metadata in the XRDS document, and perform automatic configuration.

XRI and XRDS also help enable a new of i-card called a "relationship card" or "r-card". R-cards are intended not just for one-time attribute exchange, but for ongoing, user-permissioned data sharing relationships. To support this functionality, r-card

metadata includes an XRI provisioned by the r-card issuer. An identity selector accepting this r-card can resolve this XRI to discover the service endpoint(s) for the r-card data sharing protocol(s) spoken by both the selector and the RP. The appropriate protocol can then be used to synchronize updates to r-card data, such as a change-of-address for a magazine or mailing list subscription.

Initial r-card implementations use an early version of the XDI (XRI Data Interchange) data sharing protocol under development at the OASIS XDI Technical Committee [35]. Although the final XDI 1.0 specifications are not expected until later this year, XDI is well suited to sharing data in the Higgins Data Model because it too uses an RDF graph model—one in which all nodes are addressable using XRIs. Higgins r-cards correspond to XDI *link contracts*—XDI graphs that express the policies governing usage, synchronization, redistribution, and retention of XDI data. Higgins clients can resolve the XRI in an r-card to an XRDS document to discover an XDI service endpoint and request the associated link contract to set up a data sharing relationship.

## 5. CONCLUSION AND FUTURE WORK

This paper has shown that performing secure, privacy-protecting identity discovery is a challenge even for a discovery-oriented protocol like OpenID. OpenID 2.0 was able to meet these challenges by taking advantage of the abstract resource identification and discovery features of the OASIS XRI and XRDS framework. Other Internet identity frameworks including SAML, Information Cards, and the Higgins Project have also been able to support new features and address interoperability issues using XRI and XRDS.

However we are still a long ways from a fully generalized and interoperable identity discovery layer for the Internet. For this level of abstraction, XRI and XRDS are at the same stage DNS was twenty years ago, and must mature under usage just as DNS did. Key areas of future work include:

- *Caching and scalability testing.* XRI authority servers and resolvers need the same high-performance XRDS caching as DNS nameservers and resolvers.
- *Proxying.* XRI 2.0 includes basic support for proxy resolvers that offload the work of XRI resolution and XRDS parsing to another web server. Proxy resolvers are attractive both for simplicity and performance reasons, but special attention must be paid to security, privacy, and caching requirements.
- *PKI integration.* While XRI 2.0 includes basic support for signed SAML assertions and a simple mechanism for key discovery, it has the potential to become a much more robust and generalize framework for key distribution and management. The XRI Technical Committee recently joined the OASIS IDtrust Member Section to further explore this area.
- *Reputation.* After basic location and configuration metadata, the type of discovery metadata in greatest demand is reputation. In a context as large as the Internet, where relationships are often dynamic and traditional trust cues and metrics may not be available, reputation is an essential ingredient, as sites like eBay and Slashdot have shown. It is especially relevant to an identity discovery layer because

that layer must be able to both *support* and *consume* reputation services. This is another area of focus of the OASIS IDtrust Member Section, and may spawn a new Technical Committee in that section by mid-2008.

## 6. REFERENCES

- [1] S. Cantor, J. Kemp, R. Philpott, E. Maler, 2005. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. <http://www.oasis-open.org/committees/security>.
- [2] The Shibboleth Project, 2007. Internet2/MACE. <http://shibboleth.internet2.edu/>.
- [3] Liberty Alliance Project Specifications, 2007. The Liberty Alliance Project. [http://www.projectliberty.org/liberty/specifications\\_\\_1](http://www.projectliberty.org/liberty/specifications__1)
- [4] A. Nanda, 2007. Identity Selector Interoperability Profile V1.0. Microsoft Corporation.
- [5] Higgins Project Charter, 2005, Eclipse Foundation. <http://www.eclipse.org/higgins/higgins-charter.php>.
- [6] OpenID Specifications, 2007. OpenID Foundation. <http://openid.net/developers/specs/>.
- [7] B. Fitzpatrick, 2005. OpenID Authentication 1.0. OpenID Foundation. <http://openid.net/specs/specs-1.0.bml>.
- [8] B. Fitzpatrick et al, 2007. OpenID Authentication 2.0. [http://openid.net/specs/openid-authentication-2\\_0.html](http://openid.net/specs/openid-authentication-2_0.html).
- [9] B. Fitzpatrick, D. Recordon, 2006. OpenID Authentication 1.1. OpenID Foundation. [http://openid.net/specs/openid-authentication-1\\_1.html](http://openid.net/specs/openid-authentication-1_1.html).
- [10] J. Hoyt, J. Daugherty, D. Recordon, 2006. OpenID Simple Registration 1.1. OpenID Foundation. [http://openid.net/specs/openid-simple-registration-extension-1\\_0.html](http://openid.net/specs/openid-simple-registration-extension-1_0.html).
- [11] J. Ernst, 2005. Lightweight Identity (LID). Netmesh Corporation. <http://lid.netmesh.org/>.
- [12] G. Wachob et al, 2007. Extensible Resource Identifier (XRI) Resolution 2.0 Committee Draft 02. OASIS XRI Technical Committee. <http://docs.oasis-open.org/xri/2.0/specs/cd02/xri-resolution-V2.0-cd-02.pdf>.
- [13] Ibid, Section 4.3.3.
- [14] Ibid, Section 4.3.3.
- [15] R. Moats, 1997. URN Syntax. Internet Engineering Task Force (IETF) Request for Comments (RFC), RFC 2141. <http://www.ietf.org/rfc/rfc2141.txt>.
- [16] Sam Sun, Larry Lannom, Brian Boesch, 2003. Handle System Overview. Internet Engineering Task Force (IETF) Request for Comments (RFC) 3650. HDL= <http://hdl.handle.net/4263537/4060>
- [17] Sam Sun, Sean Reilly, Larry Lannom, 2003. Handle System Namespace and Service Definition. Internet Engineering Task Force (IETF) Request for Comments (RFC) 3651. HDL= <http://hdl.handle.net/4263537/4068>
- [18] Sam Sun, Sean Reilly, Larry Lannom, Jason Petrone, 2003. Handle System Protocol (Ver 2.1) Specification. Internet

- Engineering Task Force (IETF) Request for Comments (RFC) 3652. HDL= <http://hdl.handle.net/4263537/4086>
- [19] S. Blackmer et al, 2006. XDI.org Global Services Specifications V1.0. XDI.org. <http://gss.xdi.org>.
- [20] Ibid [8]. Section 7.3.2.3.
- [21] Ibid. Section 11.5.1.
- [22] Ibid [19], Section 5.3
- [23] K. Cameron, 2005. The Laws of Identity. Microsoft Corporation.
- [24] Atwood, M et al, 2007. OAuth Core 1.0. OAuth.net. <http://oauth.net/core/1.0/>
- [25] Hammer-Lahav, E., 2007. OAuth Discovery 1.0 Draft 1. OAuth.net. <http://oauth.googlecode.com/svn/spec/discovery/1.0/drafts/1/spec.html>.
- [26] Hodges, J., 2007. Technical Comparison: OpenID and SAML, Draft 6. <http://identitymeme.org/doc/draft-hodges-saml-openid-compare-06.html>
- [27] Patterson, P, 2006. YADIS/XRI Identifier Resolution with SAML 2.0. [http://blogs.sun.com/superpat/entry/yadis/xri\\_identifier\\_resolution\\_with\\_saml](http://blogs.sun.com/superpat/entry/yadis/xri_identifier_resolution_with_saml)
- [28] Cantor, S. et al, 2005. Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard.
- [29] Hardt, D; Bufu, J., 2007. OpenID Information Cards 1.0 – Draft 01. Sxip Identity Corporation. <https://openidcards.sxip.com/spec/openid-infocards.html>
- [30] Higgins Project, 2007. The Higgins Data Model. [http://wiki.eclipse.org/Higgins\\_Data\\_Model](http://wiki.eclipse.org/Higgins_Data_Model)
- [31] Higgins Project, 2007. Context Providers. [http://wiki.eclipse.org/Context\\_Provider](http://wiki.eclipse.org/Context_Provider)
- [32] Higgins Project, 2007. I-Cards. <http://wiki.eclipse.org/I-Card>
- [33] Higgins Project, 2007. ContextId. <http://wiki.eclipse.org/ContextId>
- [34] Higgins Project, 2007. Context Discovery. [http://wiki.eclipse.org/Context\\_Discovery](http://wiki.eclipse.org/Context_Discovery)
- [35] Reed, D., Sabadello, M., 2008. The XDI RDF Model. OASIS XRI Technical Committee. <http://wiki.oasis-open.org/xdi/XdiRdfModel>