

Layered Identity Infrastructure Model for Identity Meta Systems

Suriadi Suriadi

Ernest Foo

Rong Du

Information Security Institute
Queensland University of Technology
GPO Box 2434, Brisbane, QLD 4001
Australia

Email: s.suriadi@isi.qut.edu.au, e.foo@qut.edu.au, r.du@qut.edu.au

Abstract

There are several Identity Meta Systems emerging in the identity management field, such as CardSpace and Higgins Trust Framework. The goal of an Identity Meta System (IMetS) is to integrate existing or new Identity Management System (IMS) to provide users with seamless interoperability and a consistent user experience. IMetS is a complex system that tries to integrate the already complicated IMS services. With such a complex system, we need a way to assess IMetS in order to determine how well an IMetS integrates the various IMS services. However, as IMetS is a relatively new concept, there is no *framework* to identify the properties that an ideal IMetS should have. The contribution of this paper is to introduce the Layered Identity Infrastructure Model (LIIM) that can be used as a framework to assess IMetS. In addition, the LIIM framework can also be used to identify the missing components of an IMetS, to guide and improve the design of an existing IMetS, to serve as a design benchmark for a new IMetS, as well as to aid the understanding of a complicated IMetS.

Keywords: identity management, identity meta-system, identity model, Higgins, Cardspace, OpenID

1 Introduction

There has been a push toward a user-centric system in the field of identity management system (IMS). This system aims at providing an identity management system that is *usable* from the users' point of view. It allows the users to control the use of their personal information (Kearns 2005, 2006). Consistent with this approach, Identity Meta System (IMetS) has been developed. IMetS is a system that enables interoperability between various IMS technologies, implementations and providers (Microsoft 2005a, 2006). The idea is to hide the underlying differences of various IMSs to provide the users with a single interface to use in their identity management activities and to enable a *consistent user experience*.

IMetS is *not* an IMS, and the purpose of IMetS is *not* to replace or provide new functionalities (such as authentication and single sign-on) like an IMS. Rather, IMetS provides a structure in which various IMSs can be joined together to enable interoperability between them, and to provide a single user interface, regardless of the underlying differences of the IMSs joined. IMetS *does not* and *should not*, replace

the functionalities or services that the joined IMSs have provided. For example, an IMetS could join an OpenID (Recordon & Fitzpatrick 2006) and Liberty Identity Federation (Alliance 2004b), but that IMetS should not impose or replace any of the services that these two IMSs have provided, that is, services such as the OpenID authentication or Liberty Single Sign-On protocol (Alliance 2004a,b) should remain intact. However, from the users' point of view, they are unaware of the differences because they only have a single interface to interact with.

In short, the main goal of an IMetS is to give the users of IMetS the illusion that they are dealing with one identity management system only by hiding the underlying differences between various IMSs joined - such as the differences in the protocols used, the policy representations, as well as how and where the users' information is actually stored and represented. On the other hand, the main goal of an IMS is to provide the actual identity management services, such as authentication, single sign-on, dissemination of users' attributes, privacy protections, access control and repository of the users' information.

However, attempting to join these various IMSs is challenging because the IMSs themselves are complicated systems consisting of various components, protocols (Kerberos, Microsoft's Passport), and services (such as single sign on, access control and others). Given the goal of IMetS and the complexity of IMSs, it is important that we can identify the components of IMSs that an IMetS should integrate and do so. To these ends, a model is needed to provide a *framework* for the analysis and evaluation of various IMetS.

Unfortunately, there is currently no useful model to do so. While there are some models for IMS (such as those proposed by Mont et al (Mont et al. 2002) and Overbeek (Overbeek 2006)), they are not usable for evaluating IMetS. Therefore, the contribution of this paper is to introduce a model that could be used to assess IMetS. This model is called the Layered Identity Infrastructure Model (LIIM). In addition, to show the applicability and usefulness of LIIM, two current IMetSs: CardSpace and Higgins, will be described and evaluated using the LIIM framework.

This paper is organized as follows: section 2 provides the necessary background information of existing IMetS, such as CardSpace and Higgins Trust Framework. For later discussion, a description of one IMS (OpenID) is provided. This section also provides a brief explanation of the previous attempts to model IMS and show their shortcomings to be used for IMetS evaluation, and thus, the need for the LIIM framework. Section 3 introduces LIIM and the requirements for each of the LIIM's layers. Examples are given to show how each of the LIIM layers fits into CardSpace and Higgins systems. Section 4 provides an explanation of how the LIIM framework can

be used to assess IMetS. Examples of Higgins and CardSpace evaluation using the LIIM framework are provided, including the evaluation results. Section 5 provides a discussion of the LIIM framework’s validity by showing how OpenID can be integrated with Higgins and CardSpace, and how the integration scenarios support the results from the evaluation of Higgins and CardSpace from the previous section 4. A conclusion is provided in section 6.

2 Related Works

This section describes some existing IMetSs, such as CardSpace and Higgins, as well as a promising IMS (OpenID) for the purpose of the discussion of the LIIM framework in section 5.

Some recurring concepts that will be used throughout this paper are the users, the identity providers (IdP) and the relying parties (RP). The users are the ones that use IMetSs. The IdPs are the entities that are trusted to provide and vouch for the users’ identity information. The RPs are the entities that need to consume the users’ identity information provided by the IdPs. For example, an online shopping website (RP) needs to know a user’s identity and payment-related information from the user’s bank (IdP) before a transaction can be completed.

2.1 OpenID

An example of an IMS that will be used throughout this paper is OpenID. In the OpenID Authentication scheme (Recordon & Fitzpatrick 2006), a user’s unique identifier is in the form of a URL, called the Identifier URL. When an RP needs to authenticate a user, the user is normally presented with a special HTML form field with the name ‘*openid_url*’. The Identifier URL that the user enters into this HTML form field will resolve into an HTML document, with the HEAD section of the document pointing to the user’s IdP for authentication service. The RP will then send a ‘check id’ message to the IdP to authenticate the user, by a redirect message through the user’s browser. The IdP will then try to authenticate the user (if not already authenticated) and send the response back to the RP containing an assertion about the user’s claimed Identifier URL. In addition, the RP can also include - by piggybacking into the ‘check id’ message, a request for additional attributes of the user.

2.2 Microsoft’s CardSpace

CardSpace (Chappell 2006, Microsoft 2005b) is the latest identity management system from Microsoft that is included in the Microsoft’s Vista operating system. CardSpace claims to be an IMetS (Microsoft 2005a). Users of the CardSpace system normally have a set of ‘cards’. Each of these cards contains a collection of users’ personal information, such as name, address, and age. The cards can be generated by the users themselves (that is, the users are the IdPs - also called Personal cards), or by external IdPs who have the authority to vouch for the users’ identity information (also called Managed cards). The general flows of CardSpace system is provided in Figure 1.

Assume that a user wants to access a service from an RP, but before the service can be granted, the RP needs some information about the user. In this case, the user’s CardSpace client will first retrieve the RP’s policies regarding the required information of the user, the way to format the user’s information in the way that the RP can consume, the security requirements, as well as the type of card required (Per-

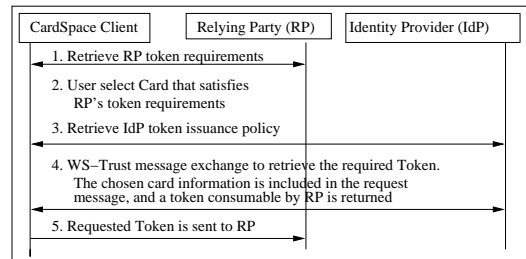


Figure 1: Simplified CardSpace Flow

sonal or Managed). The policies can be retrieved from the RP using the WS-Metadata Exchange protocol (Microsoft et al. 2006). The policy itself is defined using the WS-Security Policy language (OASIS 2007). Alternatively, the policy can be directly embedded into an HTML page using the CardSpace-compatible HTML object tags.

After a successful evaluation of the policies, the user’s CardSpace client will then display a set of cards that satisfy the RP’s policies for the user to choose. After a card is chosen, the CardSpace client will contact the IdP (who generated the chosen card) to issue a *token* that represents the user’s information. This normally requires the client to again retrieve the IdP’s policies regarding the token issuance, followed by a WS-Trust (OASIS 2006) message exchange with the IdP. The WS-Trust token request message includes information about the required token format, the user’s information, and other information required for a successful CardSpace-compatible token issuance. The WS-Trust response message from the IdP to the CardSpace client will include the requested token which will then be sent to the RP.

As an IMetS, CardSpace has many shortcomings which would make the integration with various IMSs problematic. The main problem with CardSpace is its rigid design that imposes many requirements on those who want to use and integrate with CardSpace. It requires specific methods to express the RPs and IdPs policies (by using the WS-Security Policy language, or its HTML object tags). CardSpace also requires the token request and response to be conducted in a CardSpace-compatible manner. For example, a CardSpace-compatible IdP will not be able to handle a token request expressed in a non-CardSpace-compatible format, even though the request could be expressed using the same WS-Trust protocol.

CardSpace is also unable to support integration of various IMS protocols. For example, when a user goes to an RP that uses the OpenID authentication protocol, the CardSpace client will not be able to recognize the special HTML form field tag with the name ‘*openid_url*’ as an indication that the OpenID authentication protocol is used. Thus, the user in this case will have to fill in their OpenID Identifier URL manually and be made aware that a different IMS is being used, rendering the main goal of IMetS unachievable.

2.3 Higgins Trust Framework

Higgins Trust Framework ¹ (Eclipsepedia 2006b, 2007a, Ruddy et al. 2006) is a platform and identity protocol independent software framework that could integrate various IMSs, existing or new ones in order to provide more convenience, privacy and security to the users. The architecture of Higgins is made up of many components. For our discussion, the important ones are: Identity Attribute Service (IdAS), Context Provider (CP), Token Service (TS), Token Provider

¹Higgins website: <http://www.eclipse.org/higgins/>

(TP), I-Card, RP Protocol support (RPPS), I-Card Selector Service (ISS), and Higgins Browser Extension (HBX).

The browser extension (HBX) (Eclipsepedia 2007b) with the ISS Web client or a richer GUI ISS Client are used to handle user authentication and other identity related interactions between the users, the RPs and the IdPs. When an RP needs a user's identity information, the user with Higgins installed will be presented with a set of graphical cards (called I-Cards) that contain their identity information and other related information (such as the issuer of the card) that can be used for this particular RP. The user then chooses an I-Card that they feel most comfortable to be used with the RP.

The ISS (Eclipsepedia 2007d) is the component that is used to assess the I-Cards that can be used for a particular interaction between a user and an RP (or IdP). It contains a Policy engine that is responsible for matching the RP's policy with a set of I-Card(s) that could satisfy the policies. Various policy classes can be developed to handle various policy expressions. The usable I-Cards will then be displayed for the users to choose.

Most RPs need to interact with users when they need to retrieve their information. To do this, an IMS-specific protocol is used. To help the users' agent to be able to handle the various IMS protocols employed by the RPs, Higgins provides the RPPS component which enables the users' agent to understand the various RPs protocols. Therefore, the actual protocols differences are hidden from the users.

From the users' perspective, I-Cards (Eclipsepedia 2007c) are the standard representation of their personal information. They contain information such as the issuers of the card, the users' personal information, and other information. Higgins provides an I-Card interface to support different types of cards. I-Card providers are responsible for writing the plugins for their respective I-Cards.

From the IdP's view, the TS (Token Service) is the main component that provides a standard interface that accepts requests for tokens from various RPs. The Higgins TS architecture is sophisticated enough to handle various token requests in a platform-independent manner. However, Higgins' TS is implemented using the WS-Trust language. Thus, token requests not expressed in WS-Trust protocol may not work. TS relies on TP (Token Provider) plugins to provide support for the generation of various token types, such as SAML, Kerberos, and X509.

The TS gets the required credential data of a user to generate the token from the IdAS service. CP (Eclipsepedia 2006a) is the plugin component to enable the users' information retrieval from various underlying sources, and IdAS (Eclipsepedia 2007e) is the 'interface' that acts as the wrapper of CPs to provide a common credential service. The services that IdAS provides mainly involve the management of digital subjects, including addition, deletion, editing, credential retrievals and so on. For example, several CPs can be plugged in to handle various credential retrieval/modification protocols, such as LDAP, SAML Attribute Query, and others.

As an IMetS, the design of Higgins is flexible enough to allow a good integration potential with various IMSs. Components such as TS and IdAS act as a wrapper to support various token types to be requested and generated by developing the necessary TP plugins. Similarly, various CPs can be written to support various protocols to retrieve users' information, such as through LDAP protocol.

Nevertheless, Higgins' Token Service only supports WS-Trust protocol. In many existing IMS, such as OpenID and Liberty, the request for a user's token is

not expressed in WS-Trust form. This might not be a problem since the protocol flows of both OpenID and Liberty work in such a way that the token request is forwarded from the RP to IdP directly. However, the Higgins' TS will fail if there is an IMS, similar to CardSpace, that generates a request for a token (after a user chooses an I-Card to be used) expressed in a non-WS-Trust protocol.

2.4 The Need for LIIM Framework

Section 2.2 and section 2.3 show that both CardSpace and Higgins, in their attempts to provide a seamless integration of various IMS, manage to do so up to a point, but not a complete and seamless integration as an ideal IMetS should. However, assessing an IMetS capability is difficult as normally IMetS has a complicated structure, not to mention the various aspects of IMS that an IMetS integrates.

Therefore, to help with this assessment, the LIIM framework is proposed in this paper. The LIIM framework breaks an IMetS into five layers, each layer with its own set of properties that an 'ideal' IMetS should have. The LIIM framework allows a structured dissection of various aspects of an IMetS into layers to aid the assessment of the IMetS integration capability. It allows assessment of the layers of an IMetS that are problematic, and provides insights into why they are problematic. In addition, the LIIM framework can also be used as the benchmark to guide developers in building, or improving, a new or existing IMetS. Finally, using the LIIM framework allows a measurement of the 'depth' of an IMetS's integration capability.

2.4.1 Previous IMS Models

There are some attempts to model IMS, such as those proposed by Mont et al. (Mont et al. 2002) and Overbeek (Overbeek 2006). Mont et al. model IMS as a three-layer structure, consisting of the Identity Management Infrastructure at the lowest level, the Identity Management Lifecycle on the second layer, and the Added-value Tools and Solutions at the top layer. The infrastructure layer (lowest layer) is responsible for the basic identity management operations, such as authentication, single sign-on, and access control. The lifecycle layer (middle-layer) is responsible for providing a mechanism for the creation, certification and evolution of identity information over a period of time. The Added-value Tools and Solutions (top layer) provides tools and services to simplify the operational usage and management of identities, such as identity mapping, tracing and others.

There is an overlap between the Mont et al.'s and Overbeek's models. Overbeek's model can be fitted into the Mont et al.'s Identity Management Infrastructure layer (lowest layer). However, Overbeek's model is more detailed in that it breaks that layer into 3 tiers: Authentication (at the base), Identification (middle) and Authorization (at the top). The actual model is more elaborate, but in summary, it implies that authentication is needed to identify a user for service access authorization.

These models, while valid, are models for IMS, not for IMetS. The layering is based on IMS functionalities. However, these models are not useful to assess IMetS because the goal of IMetS is *not*, as explained in section 1, to replace or modify the functionalities of IMS, but instead, to provide a framework to join various IMSs in order to provide a single consistent user interface for users in their identity management activities. As a result of this modeling approach, Mont et al.'s and Overbeek's models depict the Identity Management Infrastructure layer as a group of

services such as authentication, single sign-on, and others, without any finer layering. However, implicit in this layer itself are layers that support and enable those Identity Management Infrastructure functionalities. It is at this layer that IMetS works to join the various IMSs and that LIIM models.

3 Layered Identity Infrastructure Model - LIIM

LIIM refers to the layers that an ideal IMetS should support. LIIM is derived from an observation of the IMetSs. From this observation, the infrastructure of an IMetS is identified and a model of the layerings that an ideal IMetS should have to be able to integrate various IMSs seamlessly is developed.

LIIM is made up of five layers, the Presentation, Protocol, Policy, Token and Credential Data Source layers. In this section, each layer will be described and the properties of each layer as they apply to IMetS will be listed. The properties are described from generic to specific. The usage of this model in order to assess an IMetS integration capability will be detailed in section 4.

3.1 Presentation Layer

The main concern of this layer is how to present the identity management systems to users. That is, this layer has to provide a usable interface for users to interact with. As this is the main point of contact between a user and an IMetS, the presentation layer needs to provide a concise information about the user's Personal Identifiable Information (PII) that will be released, the party to whom the PII will be sent to, and the related policies.

As per user-centric philosophy, it is important for users to have a consistent experience in their identity management activities. And for IMetS, this means that the Presentation layer should provide a single interface for the users to use in dealing with the variety of underlying IMS. An example of this layer would be a consistent login screen to various systems. The properties of this layer can be summed up to:

- **Presentation Property 0 (PRS-0):** An IMetS must have a single common interface for the users to interact with regardless of the various underlying IMSs.
- **Presentation Property 1 (PRS-1):** An IMetS should have a usable user-interface. Following the usability principles from Jøsang et al (Jøsang et al. 2007), a usable IMetS interface should have the following properties:
 - **PRS-1.1** Users with sufficient knowledge and practical ability must be able to understand the identity-management-related actions required of them.
 - **PRS-1.2** The mental and physical loads to perform the identity management activities in using the IMetS must be tolerable, even if it is done repeatedly.
- **Presentation Property 2 (PRS-2):** An IMetS should provide a clear and concise information about the PII used, the recipient of the PII, and the related policies.

The Presentation layer is one of the most important components in IMetS for it is where the users interact with the various underlying IMSs. It is at this layer that a user can 'feel' that they are dealing with a unified system.

Both Higgins and CardSpace provide a common interface for the users. Therefore, the property PRS-0 is fulfilled. In Higgins, by using the card-metaphor, it is hoped that it makes the presentation more usable as users normally are familiar with using cards in their daily life transactions (such as presenting a driver's license card to prove age). The concept of using 'cards' as a source for identity information is quite intuitive, thus PRS-1.1 is satisfied. The activities required to use the I-Cards are tolerable (choosing the cards to use and clicking them). Therefore, PRS-1.2 is satisfied. In this manner, PRS-1 is satisfied.

The user-interface provides information about the PII included in that I-Card and the RP that the information on that I-Card will be sent to. Only I-Cards that fulfill the Policy requirements are selectable by the users. Thus, this layer also fulfills PRS-2.

CardSpace's Presentation layer is very similar to Higgins. The card-metaphor is hoped to provide a usable user-interface, and thus, fulfillment of PRS-1. The CardSpace's interface provides information about the IdP that issued the card, which RP the card is to be used, claims information, and only cards that fulfill the RP's Policy requirements are displayed and usable, thus PRS-2 is satisfied.

3.2 Protocol Layer

The Protocol layer of an IMetS handles various identity management protocols, such as the OpenID Authentication protocol (see section 2.1), SAML 2.0 Single Sign-on protocol, and many others. For an IMetS to be able to support integration with various IMS, the ability to understand and act accordingly with various IMS protocols is crucial.

An identity management protocol normally involves interaction with the users, RPs and IdPs. Consequently, the various protocols for conducting various IMS-related activities (authentication, token request and policy exchange and others), should be understood by the IMetS. Therefore, the properties that an IMetS should have at this layer are:

- **Protocol Property 0 (PROT-0):** An IMetS must support at least a protocol that can be used to aid the communication between the users, RPs and IdPs.
- **Protocol Property 1 (PROT-1):** An IMetS should be able to handle the protocol interaction between the *users and the RPs*, regardless of the variety of the protocols used. The required processing that users need to perform from this protocol interaction should be understood by the IMetS system and should be reflected accordingly to the Presentation layer and to the users.
- **Protocol Property 2 (PROT-2):** An IMetS should be able to handle the protocol interaction between the *users and the IdPs*, regardless of the variety of the protocols used. The required processing that users need to perform from this protocol interaction should be understood by the IMetS system and should be reflected accordingly to the Presentation layer and to the users.
- **Protocol Property 3 (PROT-3):** An IMetS should be able to handle the protocol interaction between the *RPs and the IdPs*, regardless of the variety of the protocols used. An IdP should be able to understand and act accordingly to an interaction using various RP protocols.

Higgins provides support, to an extent, for multiple IMS protocols to be used. Therefore, PROT-0 is satisfied. In Higgins, PROT-1 is satisfied by the use

of the RPPS component which is an interface that allows the Higgins users' agent to understand the various RPs' protocols.

Based on the design of Higgins, the possible interaction between the users and the IdPs is when a user chooses an I-Card, and a token request is sent to the IdP. Higgins uses the WS-Trust protocol in this interaction. Therefore, PROT-2 is not satisfied because if a token request is not done using the WS-Trust language, then a Higgins IdP will not be able to handle it. Similarly, this also applies to interaction between IdPs and RPs, thus PROT-3 is also not satisfied.

CardSpace provides a specific protocol that has to be used by users, RPs and IdPs. Therefore, PRS-0 is satisfied. However, the integration at this layer is very restrictive. Users that use CardSpace system cannot handle RPs that do not express their protocols in a CardSpace-compatible manner (such as the use of WS-Metadata Exchange protocol to retrieve the RP's policy). Therefore, PROT-1 is not satisfied. Interaction between a user and an IdP, and between an RP and an IdP in CardSpace is similar to Higgins, therefore PROT-2 and PROT-3 are also not satisfied.

3.3 Policy Layer

This layer is responsible for handling the policies that are associated with the use of one's PII (Personal Identifiable Information). Policy is a broad term that encompasses various aspects. For users, this could be as simple as their privacy requirements and the PII that they are willing to disclose. For IdPs and RPs, the policies could be the sort of PII required, the strength of the assertion required, and their privacy policies to be associated with the use of the PII.

For IMetS, this layer should be able to handle various policy definitions expressed in a variety of formats. For example, an RP might use P3P (Cranor et al. 2002) as the language to describe their privacy policy, while other RPs might use other expressions. IMetS should be able to understand these various policy expressions so that they can be handled accordingly. The properties for IMetS at this layer are:

- **Policy Property 0 (POL-0):** An IMetS must be able to handle the RPs', IdPs', and if applicable, users' policy expressions.
- **Policy Property 1 (POL-1):** An IMetS should be able to understand various RPs', IdPs' and, if applicable, users' policies expressed in *various* formats.
- **Policy Property 2 (POL-2):** An IMetS should not impose its own policy expression format because this will cause unnecessary changes on the RPs and IdPs sides to express their policies in an IMetS-conformant manner.

To further clarify, the enforcement of the policies should be done by IMetS. Although IMetS does not impose its own security requirements, IMetS should enforce the policy requirements. The policy requirements should be reflected in the Presentation layer, as per PRS-2. How this is done depends on the design of the IMetS. Of course, the ability of an IMetS to understand and handle RPs and IdPs policies also depends on its ability to support the protocols that are used to carry the policies.

In both Higgins and CardSpace, there are ways to understand and handle the RPs and IdPs policy expressions. Therefore, POL-0 is satisfied.

In Higgins, various policy styles can be included in the ISS Policy engine. In this manner, POL-1 is satisfied. Higgins *does not define* its own policy expression style. Instead, the ISS Policy engine understands

various policy expressions *without* imposing its own. Therefore, POL-2 is also satisfied. The enforcement of policies is achieved by allowing users to use only those cards that satisfy the given policies.

CardSpace, on the other hand, does not support various policies other than those expressed in WS-Policy format or embedded HTML policy expressed in CardSpace-specific object tags. Thus POL-1 is not satisfied. Since CardSpace imposes its own policy style, POL-2 is also not satisfied.

3.4 Token Layer

This layer is responsible for the representation of users' PII into a token. The token in itself *does not* represent any functionality (such as authentication). It is merely a representation of users' PII, either vouched for by an IdP or claims that users make. Of course a token can be used to facilitate various identity related services, however, those services happen at layers above this. There are many token styles that can be used to represent users' PII, depending on the IMS used, such as SAML (Security Assertion Markup Language) token, certificates, LDAP's LDIF representation, and many others. This is a very important layer that all IMetS should support.

In IMetS, this layer deals with various token formats. Requests for a token comes from an RP to an IdP, or from a user to an IdP, expressed in various formats. The response for the token request should be formatted in a way that the RP can use. For users, there should be a standard representation of their identity information regardless of which IMS is used. This layer's properties are:

- **Token Property 0 (TOK-0):** An IMetS has a support for requesting and generating a token to represent a user's identity information.
- **Token Property 1 (TOK-1):** An IMetS should provide a consistent representation of the user's PII to the users, regardless of where or how the actual information is stored or represented.
- **Token Property 2 (TOK-2):** An IMetS should provide a common service for token requests that accepts token requests expressed in various protocols.
 - **Token Property 2.1 (TOK-2.1):** An IMetS should be able to accept token requests expressed using various formats of a particular protocol.
- **Token Property 3 (TOK-3):** An IMetS should be able to generate various token types as required.

Both Higgins and CardSpace provide support for token generation at this layer. Therefore, property TOK-0 is fulfilled.

For users, this layer could be represented as a standard web page with standardized format in representing their PII. Or, as what CardSpace and Higgins do, this can be represented using the 'Card' metaphor. Information stored in Cards is also known as *claim*: a statement about a user's PII, such as name, gender, address, and so on. In this way, the TOK-1 property is fulfilled by both CardSpace and Higgins.

However, RPs may require assertions that the claims are valid. To do this, RPs can make requests to IdPs (whom they trust) to generate tokens that contain the users' information (vouched for by the IdPs). The tokens generated by the IdPs need to be in the form consumable by the RPs. Depending on the RPs, sometimes they might accept tokens that are validated by the users themselves (the IdP is also

the user). On other occasions, RPs might prefer the users' claim to be asserted by external IdPs that they trust. In an on-line shopping scenario, the RP would most likely prefer to have a bank to assert the correctness of a user's credit card information.

In Higgins, the TS is the main component that provides a standard interface that accepts requests for tokens from various RPs. The Higgins TS architecture is sophisticated enough to handle various token requests in a platform-independent manner. In this sense, TOK-2.1 is satisfied. However, Higgins only supports token requests using the WS-Trust protocol, therefore, property TOK-2 is not satisfied. TS allows various plugins to be added to enable the generation of various token types, such as SAML, X509 and so on. In this way, Higgins also fulfills the TOK-3 property.

Similar to Higgins, CardSpace-style cards provide a standard representation of the user's PII to user, thus, TOK-1 is satisfied. Similar to Higgins, CardSpace only supports token requests expressed using the WS-Trust language. Therefore, TOK-2 property is violated. However, unlike Higgins, CardSpace requires that a token request using WS-Trust must be formatted in the CardSpace-compatible manner according to the CardSpace technical specifications. Therefore, TOK-2.1 is also violated. CardSpace specification states that IdPs should be able to generate various token types, thus TOK-3 is satisfied.

3.5 Credential Data Layer

This layer is responsible for providing the mechanism to handle the registration, retrieval, deletion, modification and other operations on users' PII to support operations on the layers above it. The Token layer, in generating various tokens, normally accesses services from this layer to retrieve the actual users' credential data. This layer of IMetS provides an interface to allow a common way of accessing users' credential data sources implemented using various technologies (such as whether the data source is to be accessed using the LDAP system, simple XML files, databases, or federated using the SAML Attribute Query method).

From the IdP's view, a common way is needed to access these various credential systems. Since this layer provides services to layers above it, it has to have a consistent public façade to allow layers above it to access its services. Failure to do so would cause the layers above it to have to change or adopt a variety of implementation methods to communicate with the Credential layer. For example, an IdP uses LDAP as its credential system, and the token layer thus uses the LDAP protocol to communicate with this layer. However, the Token layer might also have to retrieve user's data from another credential system (an external entity) using SAML Attribute Query (OASIS 2005) protocol, and this would cause the Token layer to have to adapt to this protocol manually. If this layer is abstracted with a common façade to provide common access methods, then the token layer can access this credential system using a consistent method. By doing so, there is an elegant stream-lined process in handling identity management activities for all involved parties (users, RPs and IdPs).

Therefore, the properties for this layer are:

- **Credential Property 0 (CRED-0):** Users' credential data must be able to be retrieved by parties who need the data.
- **Credential Property 1 (CRED-1):** An IMetS should provide a common facade to handle the common operations of user's PII regardless of the actual data source systems used.

Higgins	CardSpace	Layer
ISS Web/Client UI	CardSpace interface	Presentation
RPPS	CardSpace-compatible protocol	Protocol
ISS Policy Engine	CardSpace-compatible policy	Policy
I-Card TS, TP	CardSpace-style cards CardSpace-compatible Token Service	Token
IdAS, CP	No integration support	Credential

Figure 2: Higgins and CardSpace at LIIM layers

- **Credential Property 2 (CRED-2):** An IMetS should not impose its own method to handle users' credential data source.

There are specific methods employed in Higgins to enable users' credential data retrieval. The CP (Eclipsepedia 2006a) is the plugin component for different types of credential data source systems, and IdAS (Eclipsepedia 2007e) is the 'interface' that acts as the wrapper for CPs to provide a common credential services to layers above it. The services that IdAS provides mainly involve the management of digital subjects, including addition, deletion, editing, credential retrievals and so on. For example, several CPs can be plugged in to handle various credential retrieval/modification protocols, such as LDAP, SAML Attribute Query, and others. By doing so, Higgins allows the Token layer component (TS) to retrieve the required credential data for an entity to generate the token from the IdAS service in a standard manner without having to worry about the unnecessary details. In this manner, Higgins fulfills the CRED-1 property. In addition, Higgins does not impose any specific methods to handle users' PII data source, therefore, CRED-2 is also satisfied.

As for CardSpace, there is no integration effort at this layer. Thus, CRED-1 and CRED-2 properties are not fulfilled. Nevertheless, CRED-0 is still fulfilled as the IdPs are still able to retrieve users' credential data, despite in a non-integrated manner.

A summary of the components of Higgins and CardSpace and how they fit into the LIIM framework is provided in Figure 2.

4 LIIM Usage

The previous section 3 describes the properties that an ideal IMetS should have. However, how this model should be used has not been discussed. For simplicity, several details have been left out in the description of the LIIM framework, but which will be elaborated in this section.

4.1 Types of Integration

Certain properties of LIIM apply at the users' side, while others apply at the IdPs' side. This is because by using the LIIM framework, we can access an IMetS's integration ability at both the users' and IdPs' side. The underlying assumptions with using IMetS are as follows:

1. The RPs are allowed to use whatever IMS system they prefer. Hence, they do not attempt to have any integration of various IMS. Consequently, IMetS at the RPs' side is not relevant, though some modification might have to be done for the RP to be able to talk to users using certain IMetS.
2. The IMetS at the users' side should be able to integrate various IMS that the RPs and the IdPs

PRS-0	PRS-1	PRS-2	
PROT-0	PROT-1	PROT-2	PROT-3
POL-0	POL-1	POL-2	
TOK-0	TOK-1	TOK-2	TOK-3
CRED-1		CRED-2	

User
 IdP
 Both

Figure 3: LIIM IMetS Properties for User’s and IdP’s

use to enable a seamless users’ experience when interacting with both the IdPs and the RPs.

3. The IdPs may not have to support integration of various IMSs, although it is *desirable* if they do.

For the third assumption, some may question the necessity for the IdPs to provide the integration of various IMSs. After all, what has happened up till now is that most IdPs only use one type of IMS. By enabling integration of various IMSs at the IdPs side, we can have a truly interoperable identity management system. In this scenario, an RP *does not have to support a specific IMS to be able to receive identity assertion vouched for by a certain IdP*. Therefore, an RP could choose to use whatever IMS system they prefer and still have the IdP be able to provide seamless interoperability support.

Based on these assumptions, there are two main types of IMetS integration:

1. Integration at the users’ side. Users are unaware of the underlying differences in the various IMSs used by the RPs and the IdPs. Users have a consistent experience in their identity management activities.
2. Integration at the IdPs’ side. This can be further subdivided into two types:
 - (a) IdPs’ integration with various RPs’ IMS systems. IdPs can support requests for users identity information from RPs using various IMSs.
 - (b) IdPs’ integration with various RPs’ systems *and* with various credential data source systems. In this case, there is a streamlined integration from the RPs to the IdPs communication all the way to how the IdPs should access the data source systems to retrieve the users’ PII for the RPs.

Based on these types of integrations, the LIIM framework properties can be divided into those that are exclusive to the users and the IdPs, and those that apply to both the users and the IdPs. IMetS at the RPs’ side is not relevant due to the first assumption. Referring to Figure 3, some properties are exclusive for the users, some are exclusive for the IdPs, while others are relevant to both. The Presentation layer is not relevant for the IdP because it is expected that the IdP will handle interactions with the users and RPs programmatically. The Credential data source layer is not relevant to the users because this layer handles users’ PII data storage operations and data retrieval for token generation vouched by the IdPs.

4.2 LIIM Framework for IMetS Assessment

One of the main purposes of the LIIM Framework is for evaluating an IMetS system. An assessment can be conducted to determine whether an IMetS can provide integration at the users and IdPs sides.

4.2.1 Assessing IMetS Integration for Users

Evaluation of an IMetS ability to provide a seamless user experience on the users’ side can be conducted as follows:

- For each of the relevant LIIM layers, evaluate if an IMetS fulfills those properties that are applicable to the users’ side, as shaded in Figure 3. The evaluation of those properties should result to a ‘Yes’ or a ‘No’.
- Trace the *evaluation tree* for each layer as provided in Figure 4 to obtain the evaluation for that particular layer. Assign the value of 0 if it results to ‘No Support’, 1 if it results to ‘Very Restrictive Support’, 2 if it results to ‘Some Restrictions’, and 3 if it results to ‘Good Support’.
- Combine the result of each layer’s evaluation to calculate the overall IMetS integration capability on the user’s side as explained in section 4.2.3.

Figure 4 shows the evaluation path for both Higgins and CardSpace, and the results of their respective evaluation at each layer are also shown.

4.2.2 Assessing IMetS Integration for IdP

A similar evaluation procedure is done for assessing an IMetS integration capability at the IdP’s side. However, this time the properties relevant to IdPs will be assessed as per Figure 5.

Figure 5 shows the evaluation path for both Higgins and CardSpace, and the results of their respective evaluation at each layer are also shown.

4.2.3 Overall IMetS Evaluation

For an overall evaluation of an IMetS’s integration at either the user or the IdP’s side, the following procedure applies: If evaluation of any of the layers resulted in 0, then it means that the integration efforts will fail. Else, an average score will be calculated from each of the layers to derive an overall value of anywhere between 1 (very restrictive integration - many integration efforts will fail) to 3 (good integration support, integration will very likely to succeed).

Overall evaluation of an IMetS is done using the same approach for both the evaluation at the user’s side and the IdP’s side. The only difference is that for the user’s side, the layers involved are the Presentation, Protocol, Policy and Token layers, while for the IdP’s side, the layers involved are the Protocol, Policy, Token and Credential layers.

None of the layers for Higgins and CardSpace integration evaluation (both for user’s and IdP’s sides) results to a score of 0. The overall calculation of the Higgins and CardSpace integration score is shown in Table 1.

IMetS System	Side	Overall Score
CardSpace	User’s side	$3+1+1+3/4=2$
	IdP’s side	$1+1+1+1/4=1$
Higgins	User’s side	$3+2+3+3/4=2.75$
	IdP’s side	$1+3+2+3/4=2.25$

Table 1: Overall CardSpace and Higgins Evaluation

The overall evaluation results can be interpreted as follows: CardSpace has an average integration support on the user’s side (overall score of 2), therefore, there will be some cases where integration efforts will fail. CardSpace’s integration support at the IdP side is very restrictive (overall score of 1), therefore, most integration efforts will fail. For Higgins, it has a very

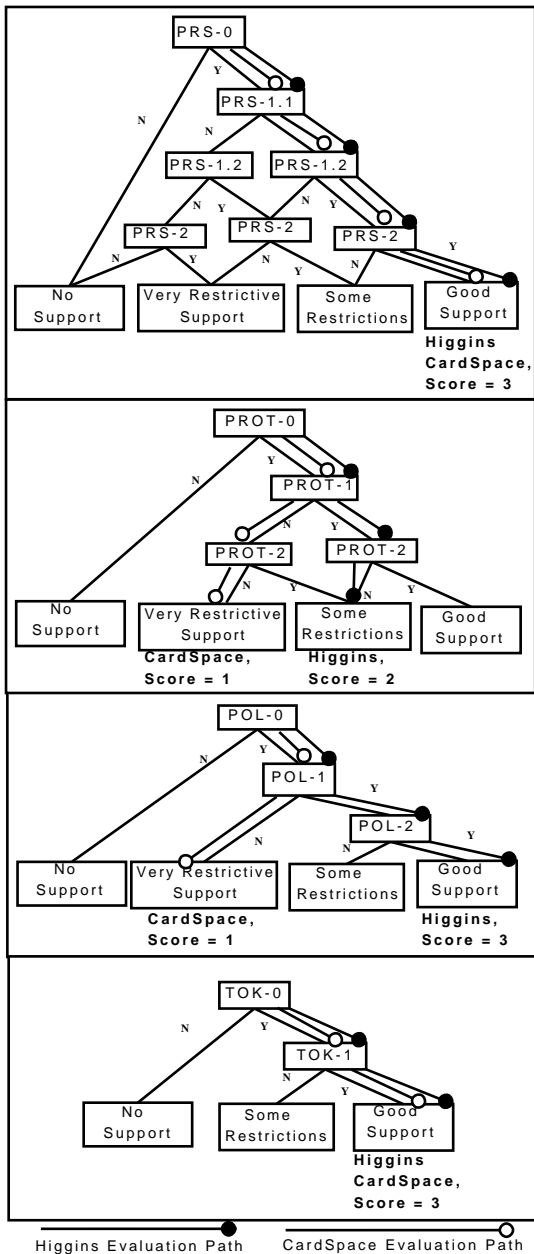


Figure 4: IMetS User's Side Integration Evaluation

good integration support at the user's side (overall score of 2.75), though there will be very few cases where integration effort will fail. Higgins has an average integration support for the IdP's side (score of 2.25), therefore some integration efforts will fail.

5 LIIM Framework Discussion

To show the validity of the LIIM framework, an OpenID Authentication integration scenario will be provided. An RP is using the OpenID Authentication protocol interacting with two types of users and IdPs, one uses CardSpace, while the other uses Higgins. Using the same integration scenario, the validity of LIIM framework will also be further argued by showing how it can be used as a tool to design or improve a new or existing IMetS, as well as to aid understanding of a complicated IMetS.

5.1 User's Side Integration Scenario

When a Higgins user goes to the RP site using a browser, the HBX component would call the RPPS

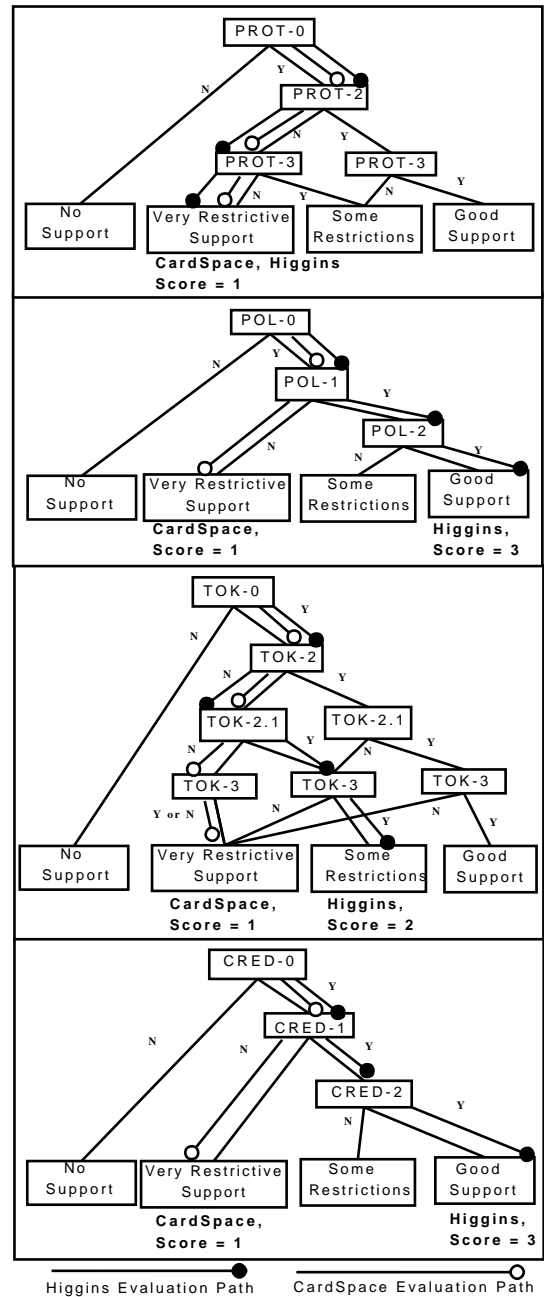


Figure 5: IMetS IdP's Side Integration Evaluation

component to detect that a returned HTML page is request for the OpenID authentication (because the HTML page will contain a form with a field named 'openid_url', which is recommended in the OpenID 1.1 specification). Recognizing that this is an OpenID authentication, the RPPS (Protocol layer) and ISS Policy class component (Policy layer) along with other supporting components will evaluate which I-Card (Token layer) that can be used for this RP. Assume that there is an I-Card for OpenID that contains the user's OpenID Identifier URL, the ISS Policy engine will evaluate usable I-Cards and present them through the ISS Web UI (Presentation layer). Upon selecting the appropriate I-Card containing the user's OpenID identifier, the RPPS could automatically fill in the 'openid_url' field with the URL identifier from the card selected and return the HTML form to the RP. By doing so, a Higgins user can have a seamless integrated experience and should be unaware that an OpenID Authentication has been used.

This is consistent with our evaluation result of Hig-

gin's integration capability from section 4.2.1. The Protocol layer evaluates to 'some restrictions' which means we may or may not expect integration problems. In this case, we did not. However, this is because the property PROT-2 does not apply as there is no direct communication between the user and the IdP in the OpenID Authentication protocol. The Presentation, Policy and Token layers evaluate to a score of 3, thus no integration problem is expected at those layers, which is exactly the case in this scenario.

With CardSpace, however, the Policy and Protocol layers evaluate to 'very restrictive'. Thus, we should expect a good chance of integration failure. In CardSpace, the user agent does not have the support component, such as RPPS in Higgins, to detect that the RP sent an OpenID authentication request and the CardSpace policy engine will not be able to find CardSpace-style cards that can be used with this RP. In short, the Presentation layer of CardSpace becomes useless (although we evaluated it to have a 'good support') because the Protocol layer does not support any other request from RP except if it was expressed in CardSpace-compatible manner. In this situation, the user, therefore, has to be presented with a traditional HTML form to fill in their URL identifier.

5.2 IdP's Side Integration Scenario

Next in the OpenID Authentication protocol, the RP would find the IdP based on the given URL identifier and send a 'check id' request. However, in both CardSpace and Higgins, the IdP's side Protocol layer integration evaluation results in 'very restrictive support'. Thus, we can expect that integration of OpenID Authentication protocol at the IdP side will most likely fail.

In Higgins, the 'check id' request sent from the RP to the IdP will not be understood as Higgins' IdP does not have any support for PROT-3 (the protocol interaction between RP and IdP). Therefore, the integration will fail at this point. Integration assessment at other layers are thus irrelevant. In fact, this is exactly what happened. In the OSIS (Open-Source Identity System) interoperability space², Higgins is not involved at all in the interaction between OpenID RP and IdP. A similar thing would happen to an IdP that uses the CardSpace system.

5.3 Identification of Missing or Problematic Components

From section 5.1, we know that CardSpace has problems with user's side integration with OpenID. By using the LIIM Framework, it is straightforward to see that the missing components that cause the integration problem is the lack of support for PROT-1, POL-1 and POL-2 properties in CardSpace. Thus, what needs to be done is to add components in CardSpace that can handle various RP protocols and policy expressions, instead of those expressed using CardSpace-compatible format only.

Similarly, for a seamless integration at the IdP's side, the most obvious missing component for both CardSpace and Higgins is the component to support PROT-3 property (the protocol support for communication between RP and IdP).

5.4 Improvement of IMetS

We have now identified the need to develop components that would provide PROT-3, PROT-1, POL-1 and POL-2 to CardSpace. However, this may not be

enough. Using the LIIM Framework, it is straightforward to identify which other components that we might need.

For every layer that evaluates to a score of 1 or lower, it is very likely that new component(s) will have to be developed so that the evaluation will result in a score of 2 or higher (because a score of 2 indicates at least a reasonable integration support for that layer, despite still having some restrictions). For example, for CardSpace integration at the IdP's side, not only do we need to develop support for PROT-3, we also need to provide support for TOK-2.1 in order to have the Token layer evaluation to result in the score of 2. Depending on the improvement goal, property TOK-2 may also have to be supported if the goal is to have the Token layer evaluation to result in a score of 3. Similar process applies to Higgins as well.

5.5 Better Understanding of IMetS Problems

Using the LIIM framework, a better understanding an IMetS's problems is possible. The LIIM framework *isolates* problems into their respective layers. Thus, instead of grouping problems indiscriminately, the LIIM framework allows a clear and elegant way to describe and understand IMetS problems.

For example, we know that CardSpace has a problem with integration at the user's side. By using the LIIM framework, we have evaluated the CardSpace Protocol and Policy layers to a score of 1, while the Presentation layer of CardSpace evaluates to a score of 3. These evaluations tell us that the actual problems with CardSpace is *not* at the Presentation layer, but lies on the Protocol and Policy layers. After determining the problematic layers, it is straightforward to identify exactly the components that need to be 'fixed'. In this case, we want CardSpace to at least provide support for properties PROT-1 and POL-1.

5.6 Aids Understanding of Complex IMetS

The LIIM framework is also useful in aiding the understanding of a complex IMetS. For example, a first look at Higgins Trust Framework may make it seem very complicated. Trying to understand Higgins may prove to be difficult as one would not know the purpose of each of the Higgins' components, and how they all fit together to form an IMetS.

By using the LIIM framework, it allows the decomposition of any IMetS into its five layers. Thus, LIIM framework makes the seemingly daunting task of understanding a complex IMetS into a more manageable task; the decomposition of IMetS into its five layers and identifying components that fit into each of those layers. The next step is to understand how each of the identified components works together to provide integration at the user's and the IdP's sides.

5.7 Design Benchmark for IMetS

The LIIM framework can also be used as the design benchmark for developing new IMetS. LIIM describes the necessary layers and properties that an ideal IMetS should provide in order to achieve integration at the user's and/or the IdP's side, as well as at how 'comprehensive' the integration should be.

Therefore, by using this framework, a designer and developer can develop properties that they know their IMetS should support in order to achieve a certain integration level. For example, if a developer needs to develop an IMetS that provides integration at the IdP's side with the most comprehensive Token layer

²http://osis.netmesh.org/wiki/Interop_Use_Cases

integration (a score of 3), then he/she knows that the developed IMetS should provide support for TOK-0, TOK-2, TOK-2.1 and TOK-3 properties.

5.8 Further Advantages of LIIM Framework

We would argue that LIIM's IMetS framework is *practical* in that the layers are generally easy to identify and properties of IMetS are straightforward enough to be assessed for its supports or lack of them. This framework is also *extensible*. The framework proposed here is based on the current development of IMetS. However, there could be additional requirements of IMetS in the future that are not obvious at this stage. Extending the framework is as simple as adding or modifying the properties for each layer, or perhaps adding new layer as needed. Finally, this framework is *independent* of any existing IMetS. Even after Higgins or CardSpace are outdated and are not in the market anymore, this framework can still be used to analyze any new IMetS.

6 Conclusion

In conclusion, the benefits and usefulness of LIIM's IMetS framework rest in the abstraction of IMetS system into its layers. This framework can be used as a tool to analyze and evaluate any existing or new IMetS. The LIIM framework provides a new model of the IMetS infrastructure layers. LIIM's IMetS framework introduces several key properties that should be satisfied for an IMetS to be successful in integrating various IMSs to provide users with a consistent user experience.

This paper has also evaluated both CardSpace and Higgins, and shows that CardSpace is a restrictive IMetS, and thus allowing limited integration possibilities, while Higgins is a more flexible IMetS, thus able to support better integration of various IMSs, although it still has its significant shortcomings.

LIIM framework can be used not only to assess IMetS, but also for many other purposes, such as for identification of missing components in an IMetS, as a guide to improve an IMetS, as a design benchmark for IMetS, and many others as detailed in section 5. However, one drawback of the LIIM framework is that in determining whether an IMetS fulfills a certain property, it requires subjective human judgment as well as a substantial technical knowledge of the assessed IMetS.

References

- Alliance, L. (2004a), *Liberty ID-FF Bindings and Profiles Specification*, Liberty Alliance.
- Alliance, L. (2004b), *Liberty ID-FF Protocols and Schema Specification*, Liberty Alliance.
- Chappell, D. (2006), 'Introducing windows cardspace', *Windows Vista Technical Article*. <http://msdn2.microsoft.com/en-us/library/aa480189.aspx>. Last access 18 Oct 2007.
- Cranor, L., Langheinrich, M., Marchiori, M., Presler-Marshall, M. & Reagle, J. (2002), *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*, W3C.
- Eclipsepedia (2006a), 'Context Provider'. http://wiki.eclipse.org/index.php/Context_Provider. Last access 18 Oct 2007.
- Eclipsepedia (2006b), 'Introduction to Higgins'. http://wiki.eclipse.org/index.php/Introduction_to_Higgins. Last access 18 Oct 2007.
- Eclipsepedia (2007a), 'Higgins Architecture'. <http://wiki.eclipse.org/index.php/Architecture>. Last access 15 August 2007.
- Eclipsepedia (2007b), 'Higgins Browser Extension'. http://wiki.eclipse.org/index.php/Higgins_Browser_Extension. Last access 18 Oct 2007.
- Eclipsepedia (2007c), 'I-Card'. <http://wiki.eclipse.org/index.php/I-Card>. Last access 18 Oct 2007.
- Eclipsepedia (2007d), 'I-Card Selector Service'. http://wiki.eclipse.org/index.php/I-Card_Selector_Service.
- Eclipsepedia (2007e), 'IdAS API'. http://wiki.eclipse.org/index.php/IdAS_API. Last access 18 Oct 2007.
- Jøsang, A., AlZomai, M. & Suriadi, S. (2007), 'Usability and privacy in identity management architectures', *Proceedings of the Australasian Information Security Workshop (AISW) 2007*.
- Kearns, D. (2005), 'Users should be in control of their own identity attributes', *Network World*. <http://www.networkworld.com/newsletters/dir/2005/0808id2.html>. Last access 18 Oct 2007.
- Kearns, D. (2006), 'What is 'user-centric' identity?', *Network World*. <http://www.networkworld.com/newsletters/dir/2006/0710id1.html>. Last access 18 Oct 2007.
- Microsoft (2005a), 'Microsoft's vision for an identity metasytem', *MSDN*.
- Microsoft (2005b), *A Technical Reference for InfoCard v1.0 in Windows*, Microsoft.
- Microsoft (2006), 'The identity metasytem: Towards a privacy-compliant solution to the challenges of digital identity', White Paper.
- Microsoft, IBM et al. (2006), *Web Services Metadata Exchange (WS-MetadataExchange)*.
- Mont, M. C., Bramhall, P., Gittler, M., Pato, J. & Rees, O. (2002), Identity management: a key e-business enabler, Technical report, Trusted E-Services Laboratory, HP Laboratories, Bristol.
- OASIS (2005), *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*, OASIS. OASIS Standard.
- OASIS (2006), *WS-Trust 1.3*, OASIS. OASIS Standard.
- OASIS (2007), *WS-SecurityPolicy 1.2*, OASIS. OASIS Standard.
- Overbeek, S. (2006), 'Unifying framework for identity management'. <http://www.security-assessment.com/files/presentations/Unifying%20Framework%20for%20Identity%20management%20-%20Stephan%20Overbeek%20-%20Breakfast%20seminar%20SA%20com%20-%2020060328.pdf>. Last access 18 Oct 2007.
- Recordon, D. & Fitzpatrick, B. (2006), *OpenID Authentication 1.1*. http://openid.net/specs/openid-authentication-1_1.html. Last access 18 Oct 2007.
- Ruddy, M., Trevithick, P., Nadalin, T. & Olds, D. (2006), 'Higgins trust framework', *Digital ID World*.