A Guide to Using the Identity Selector Interoperability Profile V1.5 within Web Applications and Browsers

July 2008

Author

Michael B. Jones, Microsoft Corporation

Copyright Notice

Abstract

The Identity Metasystem allows users to manage their Digital Identities from various Identity Providers and employ them in different contexts where they are accepted to access online services. In the Identity Metasystem, identities are represented to users as "Information Cards" (a.k.a. "InfoCards"). One important class of applications where Information Cards can be used is applications hosted on Web sites and accessed through Web browsers.

This paper documents the Web interfaces utilized by browsers and Web applications that support the Identity Metasystem. The information in this document is not specific to any one browser or platform.

This document supplements the information provided in two other Identity Selector Interoperability Profile references: the "Identity Selector Interoperability Profile V1.5" [ISIP], which provides the normative schema definitions and behaviors for Information Cards and the interoperable Identity Selectors that use them and "An Implementer's Guide to the Identity Selector Interoperability Profile V1.5" [ISIP Guide], which provides a non-normative description of the overall Information Card Model.

Status

The information presented in this document is informative; the normative definitions can be found in [ISIP].

The Identity Selector Interoperability Profile V1.5 was used to implement the Windows CardSpace software that is part of Microsoft .NET Framework 3.5 [.NET3.5] and Internet Explorer 7. Other implementations following these specifications are able to interoperate with Microsoft's implementations.

Table of Contents

1. Introduction
2. Design Goals
3. Browser Behavior with Information Cards
3.1. Basic Protocol Flow when using an Information Card at a Web Site
3.2. Protocol Flow with Relying Party STS
3.3. User Perspective and Examples
3.4. Browser Perspective
3.5. Web Site Perspective
4. Invoking an Identity Selector from a Web Page
4.1. Syntax Alternatives: OBJECT and XHTML tags
4.1.1. OBJECT Syntax Examples
4.1.2. XHTML Syntax Example
4.2. Identity Selector Invocation Parameters
4.2.1. issuer (optional)
4.2.2. issuerPolicy (optional)
4.2.3. tokenType (optional)
4.2.4. requiredClaims (optional)
4.2.5. optionalClaims (optional)
4.2.6. privacyUrl (optional)
4.2.7. privacyVersion (optional)
4.3. Data Types for Use with Scripting
4.4. Detecting and Utilizing an Information Card-enabled Browser
4.5. Behavior within Frames
4.6. Invocation Using the Document Object Model (DOM)
4.7. Auditing, Non-Auditing, and Auditing-Optional Cards
5. References
Appendix A – HTTPS POST Sample Contents
Appendix B – Detecting CardSpace Browser Support by Internet Explorer
Appendix C – Windows CardSpace .NET Framework 3.5 Service Pack 1 Constraints

1. Introduction

The Identity Metasystem allows users to manage their Digital Identities, whether they are self-issued or issued by third-party Identity Providers, and employ them in contexts where they are accepted to access online services. In the Identity Metasystem, identities are represented to users as "Information Cards" (a.k.a. "InfoCards"). One important class of applications where Information Cards can be used is applications hosted on Web sites and accessed through Web browsers.

This paper documents the Web interfaces utilized by browsers and Web applications that support the Identity Metasystem. The information in this document applies to all platforms, browsers, and Identity Selectors. These mechanisms are documented here to enable Web sites and browsers on all platforms to implement and use these mechanisms so they can utilize Information Cards.

Two other documents should be considered prerequisites for understanding this document: "An Implementer's Guide to the Identity Selector Interoperability Profile V1.5" [ISIP Guide], which provides a non-normative description of the overall Information Card Model, and the "Identity Selector Interoperability Profile V1.5" [ISIP], which provides the normative schema definitions and behaviors referenced by this document.

The Identity Selector Interoperability Profile V1.5 was used to implement the Windows CardSpace software that is part of Microsoft .NET Framework 3.5 [.NET3.5]. Windows CardSpace implements an Identity Selector that allows users to select Information Cards to use for sites and applications that accept them. Internet Explorer 7 uses Windows CardSpace, if installed. The information in this document corresponds to the features supported in the Identity Selector Interoperability Profile V1.5.

2. Design Goals

Numerous alternatives were considered for ways of bringing Information Cards to Web sites. The design goals that led to the eventual decisions described in this document are:

- **Browser independent:** A goal was to ensure that the protocols developed for using Information Cards on Web sites could be implemented by a broad range of Web browsers on the platforms of their choice.
- **Web server independent:** A closely-related goal was to ensure that the protocols developed for Information Cards on Web sites could be used by Web-based applications running on a broad range of Web servers on the platforms of their choice.
- **Minimal impact on Web sites:** A goal was to facilitate the adoption of Information Cards on existing Web sites by requiring as few changes to them as possible.
- **Seamless browser integration:** A goal was that Information Cards should be viewed as a seamless security feature that is a natural extension of the browser(s) being used.
- **Seamless user experience:** A goal was that the Information Card Web integration design should permit graceful fall-back when a browser or platform does not have Information Card support available.
- **Work with browser high security settings:** A goal was that the mechanisms chosen should remain enabled even when browser security settings are set to "high".

The choices described in this document are an attempt to balance among all these sometimes-competing goals and to achieve all of them as well as possible, given the realities of how the Web is used today.

3. Browser Behavior with Information Cards

This section explains the steps that a Web browser takes when using an Information Card to authenticate to a Web site. Two cases are described. The basic case is where the Web site provides all the Relying Party functionality via HTML extensions transported over HTTPS. The second case is where the Relying Party employs a Relying Party Security Token Service (STS), which it references via HTML extensions transported over HTTPS.

3.1. Basic Protocol Flow when using an Information Card at a Web Site

This section explains the protocol flow when using an Information Card to authenticate at a Web site where no Relying Party STS is employed.

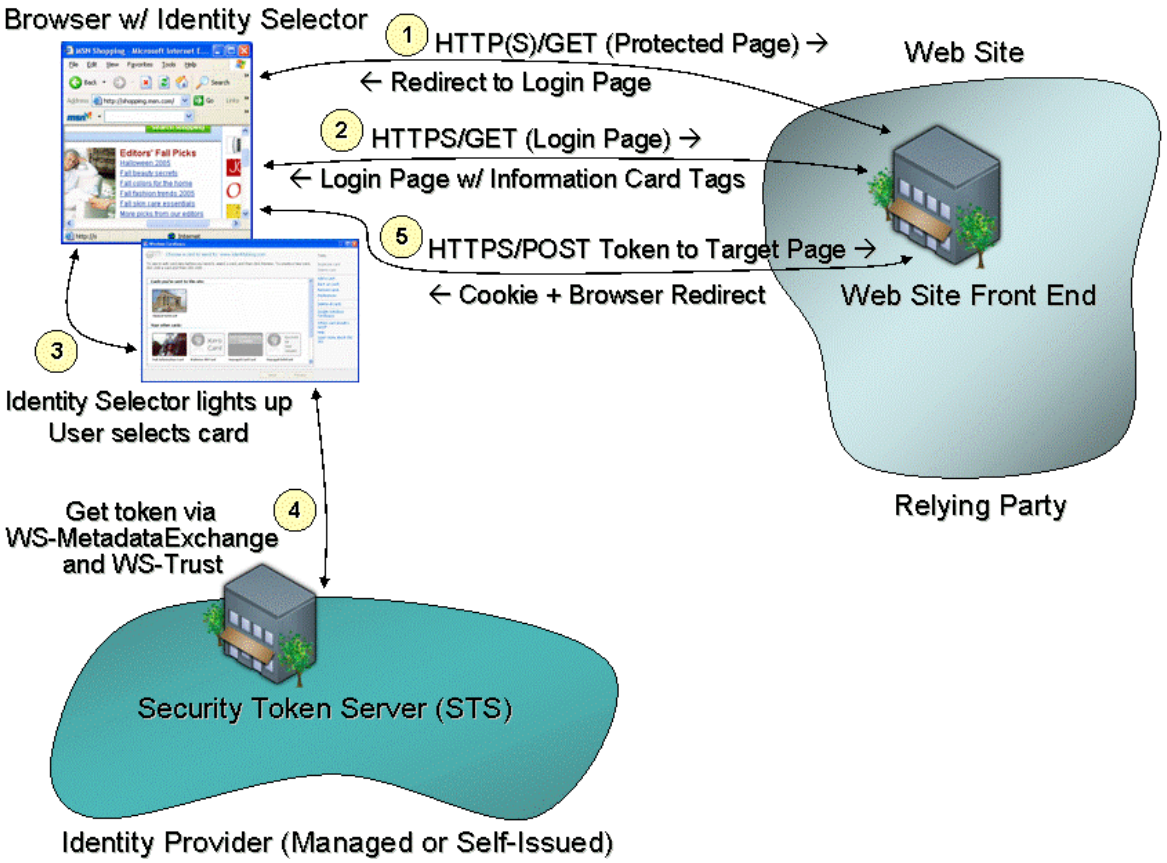

Figure 1. Basic protocol flow when using an Information Card to authenticate at a Web site

Figure 1 gives an example of the basic protocol flow when an Information Card is used to authenticate at a Web site that employs no Relying Party STS. Steps 1, 2, and 5 are essentially the same as a typical forms-based login today: (1) The user navigates to a protected page that requires authentication. (2) The site redirects the browser to a login page, which presents a Web form. (5) The browser posts the Web form that includes the login credentials supplied by the user back to the login page. The site then validates the contents of the form including the user credentials, typically writes a client-side browser cookie to the client for the protected page domain, and redirects the browser back to the protected page.

The key difference between this scenario and today's site login scenarios is that the login page returned to the browser in step (2) contains an HTML tag that allows the user to choose to use an Information Card to authenticate to the site. When the user selects this tag, the browser invokes an Identity Selector, which implements the Information Card user experience and protocols, and triggers steps (3) through (5).

In Step (3), the browser Information Card support code invokes the Identity Selector, passing it parameter values supplied by the Information Card HTML tag supplied by the site in Step (2). The user then uses the Identity Selector to choose an Information Card, which represents a Digital Identity that can be used to authenticate at that site. Step (4) uses the standard Identity Metasystem protocols [ISIP] to retrieve a Security Token that represents the Digital Identity selected by the user from the STS at the Identity Provider for that identity.

In Step (5), the browser posts the token obtained back to the Web site using a HTTPS/POST. The Web site validates the token, completing the user's Information Card-

based authentication to the Web site. Following authentication, the Web site typically then writes a client-side browser cookie and redirects the browser back to the protected page.

It is worth noting that this cookie is likely to be *exactly the same cookie* as the site would have written back had the user authenticated via other means, such as a forms-based login using username/password. This is one of the ways that the goal of "minimal impact on Web sites" is achieved. Other than its authentication subsystem, the bulk of a Web site's code can remain completely unaware that Information Card-based authentication is even utilized. It just uses the same kinds of cookies as always.

3.2. Protocol Flow with Relying Party STS

In the previous scenario, the Web site communicated with the client Identity Selector using only the HTML extensions enabling Information Card use, transported over the normal browser HTTPS channel. In this scenario, the Web site also employs a Relying Party STS to do part of the work of authenticating the user, passing the result of that authentication on to the login page via HTTPS POST.

There are several reasons that a site might factor its solution this way. One is that the same Relying Party STS can be used to do the authentication work for both browser-based applications and smart client applications that are using Web services. Second, it allows the bulk of the authentication work to be done on servers dedicated to this purpose, rather than on the Web site front-end servers. Finally, this means that the front-end servers can accept site-specific tokens, rather than the potentially more general or more complicated authentication tokens issued by the Identity Providers.

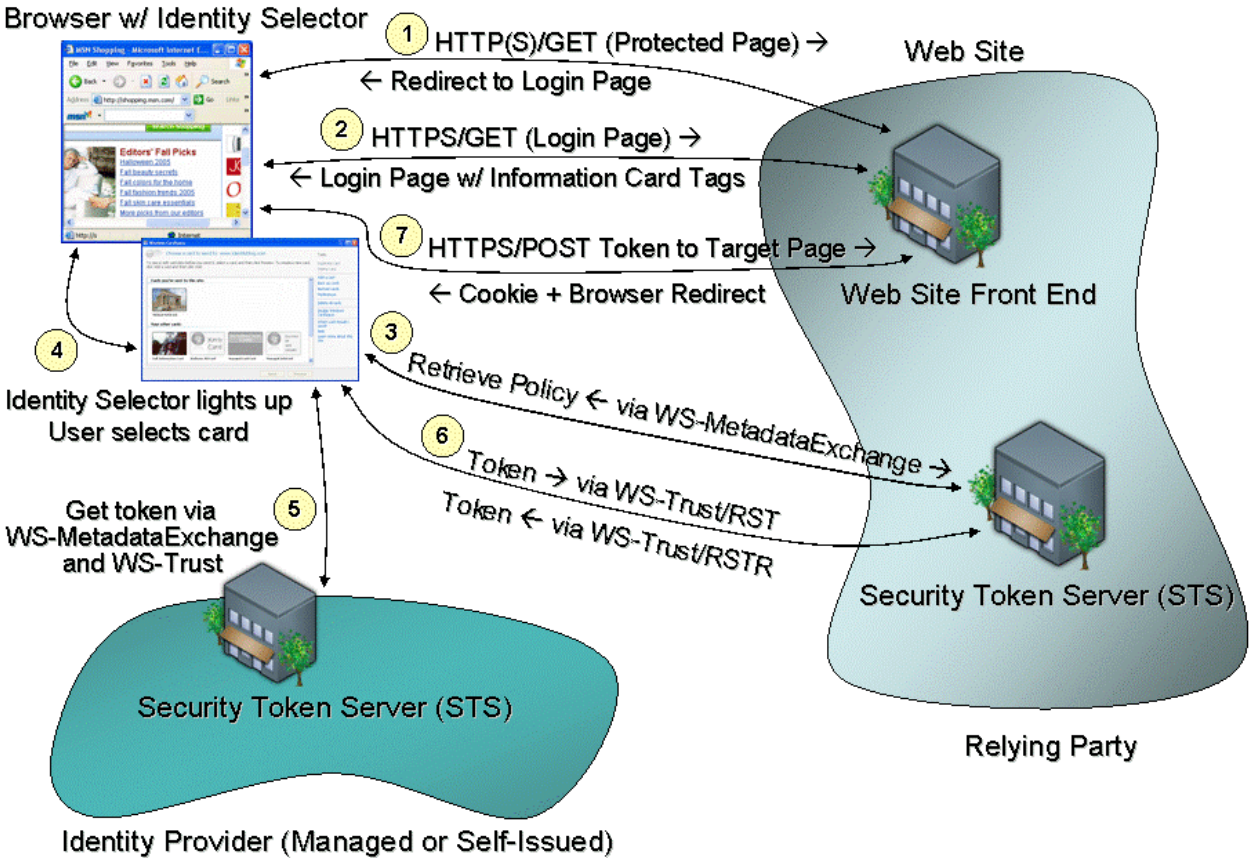

Figure 2. Protocol flow when using an Information Card to authenticate at a Web site, where the Web site employs a Relying Party STS

This scenario is similar to the previous one, with the addition of steps (3) and (6). The differences start with the Information Card information supplied to the browser by the Web site in Step (2). In the previous scenario, the site encoded its WS-SecurityPolicy information using Information Card HTML extensions and supplied them to the Information Card-extended browser directly. In this scenario, the site uses different Information Card HTML extensions in the Step (2) reply to specify which Relying Party STS should be contacted to obtain the WS-SecurityPolicy information.

In Step (3), the Identity Selector contacts the Relying Party STS specified by the Web site and obtains its WS-SecurityPolicy information via WS-MetadataExchange. In Step (4) the Identity Selector user interface is shown and the user selects an Information Card, which represents a Digital Identity to use at the site. In Step (5), the Identity Provider is contacted to obtain a Security Token for the selected Digital Identity. In Step (6), the Security Token is sent to the Web site's Relying Party STS to authenticate the user and a site-specific authentication token is returned to the Identity Selector. Finally, in Step (7), the browser posts the token obtained in Step (6) back to the Web site using HTTPS/POST. The Web site validates the token, completing the user's Information Card-based authentication to the Web site. Following authentication, the Web site typically then writes a client-side browser cookie and redirects the browser back to the protected page.

3.3. User Perspective and Examples

The Information Card user experience at Web sites is intended to be intuitive and natural enough that users' perspective on it will simply be "That's how you log in". Today, Web sites that require authentication typically ask the user to supply a username and password at login time. With Information Cards, they instead ask users to choose an Information Card. Some sites will choose to accept only Information Cards whereas others will give users the choice of Information Cards or other forms of authentication.

A site that accepts Information Cards typically has a login screen that contains button with a label such as "**Sign in with an Information Card**" or "**Log in using an Information Card**". Upon clicking this button, the user is presented with a choice of his Information Cards that are accepted at the site, and is asked to choose one. Once a card is selected and submitted to the site, the user is logged in and continues using the site, just as they would after submitting a username and password to a site.

Sites that accept both Information Cards and other forms of authentication present users with both an Information Card login choice and whatever other choices the site supports. For instance, a site login screen might display both "**Sign in with your username and password**" and "**Sign in with an Information Card**" buttons.

3.4. Browser Perspective

Very little additional support is required from today's Web browsers to also support Information Cards. The main addition is that they must recognize special HTML and/or XHTML tags for invoking the Identity Selector, pass encoded parameters on to the Identity Selector on the platform, and POST back the token resulting from the user's choice of an Information Card.

3.5. Web Site Perspective

Web sites that employ Information Card-based authentication must support two new pieces of functionality: adding HTML or XHTML tags to their login page to request an Information Card-based login and code to log the user into the site using the POSTed credentials. In response to the Information Card-based login, the Web site typically writes the same client-side browser cookie that it would have if the login had occurred via username/password authentication or other mechanisms, and issue the same browser redirects. Thus, other

than the code directly involved with user authentication, the bulk of a Web site can remain unchanged and oblivious to the site's acceptance of Information Cards as a means of authentication.

4. Invoking an Identity Selector from a Web Page

4.1. Syntax Alternatives: OBJECT and XHTML tags

HTML extensions are used to signal to the browser when to invoke the Identity Selector. However, not all HTML extensions are supported by all browsers, and some commonly supported HTML extensions are disabled in browser high security configurations. For example, while the OBJECT tag is widely supported, it is also disabled by high security settings on some browsers, including Internet Explorer.

An alternative is to use an XHTML syntax that is not disabled by changing browser security settings. However, not all browsers provide full support for XHTML.

To address this situation, two HTML extension formats are specified. Browsers may support one or both of the extension formats.

4.1.1. OBJECT Syntax Examples

An example of the OBJECT syntax is as follows:

```
<html>
  <head>
    <title>Welcome to Fabrikam</title>
  </head>
  <body>
    <img src='fabrikam.jpg'/>
    <form name="ctl00" id="ctl00" method="post"
        action="https://www.fabrikam.com/InfoCard-Browser/Main.aspx">
      <center>
        <img src='infocard.bmp' onClick='ctl00.submit()'/>
        <input type="submit" name="InfoCardSignin" value="Log in"
          id="InfoCardSignin" />
      </center>
      <OBJECT type="application/x-informationCard" name="xmlToken">
        <PARAM Name="tokenType"
Value="urn:oasis:names:tc:SAML:1.0:assertion">
        <PARAM Name="issuer" Value=
            "http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self">
        <PARAM Name="requiredClaims" Value=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname">
</OBJECT>
    </form>
  </body>
</html>
```

This is an example of a page that requests that the user log in using an Information Card. The key portion of this page is the OBJECT of type "`application/x-informationCard`". Once a card is selected by the user, the resulting Security Token is included in the resulting POST as the xmlToken value of the form. Appendix A shows a sample POST resulting from using a login page similar to the preceding one. If the user cancels the authentication request, the resulting POST contains an empty xmlToken value.

Parameters of the Information Card OBJECT are used to encode the required WS-SecurityPolicy information in HTML. In this example, the Relying Party is requesting a SAML 1.0 token from a Self-issued Identity Provider, supplying the required claims "emailaddress", "givenname", and "surname". This example uses the basic protocol described in Section 3.1 (without employing a Relying Party STS).

A second example of the OBJECT syntax is as follows:

```
<html>
  <body>
    <form name="ctl01" method="post"
        action="https://www.fabrikam.com/InfoCard-Browser-STS/login.aspx"
        id="ctl01" onSubmit="fnGetCard();">
      <img src='infocard.bmp' onClick='ctl01.submit()'/>
      <input type="submit" name="InfoCardSignin" value="Log in"
          id="InfoCardSignin" />
      <OBJECT type="application/x-informationCard" name="xmlToken"
          ID="oCard" />
    </form>
    <script type="text/javascript">
    <!--
      function fnGetCard(){
        oCard.issuer = "http://www.fabrikam.com/sts";
        oCard.issuerPolicy = "https://www.fabrikam.com/sts/mex";
        oCard.tokenType = "urn:fabricam:custom-token-type";
      }
    //-->
    </script>
  </body>
</html>
```

This example uses the enhanced protocol described in Section 3.2, which employs a Relying Party STS. Note that in this case, the "issuer" points to a Relying Party STS. The "issuerPolicy" points to an endpoint where the Security Policy of the STS (expressed via WS-SecurityPolicy) is to be obtained using WS-MetadataExchange. Also, note that the "tokenType" parameter requests a custom token type defined by the site for its own purposes. The "tokenType" parameter could have been omitted as well, provided that the Web site is capable of understanding all token types issued by the specified STS or if the STS has prior knowledge about the token type to issue for the Web site.

The object parameters can be set in normal script code. This is equivalent to setting them using the "PARAM" declarations in the previous example.

4.1.2. XHTML Syntax Example

An example of the XHTML syntax is as follows:

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity">
  <head>
    <title>Welcome to Fabrikam</title>
  </head>
  <body>
    <img src='fabrikam.jpg'/>
    <form name="ctl00" id="ctl00" method="post"
        action="https://www.fabrikam.com/InfoCard-Browser/Main.aspx">
      <ic:informationCard name='xmlToken'
          style='behavior:url(#default#informationCard)'
          issuer="http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self"
          tokenType="urn:oasis:names:tc:SAML:1.0:assertion">
```

```
        <ic:add claimType=
       "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"
            optional="false" />
        <ic:add claimType=
         "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"
            optional="false" />
        <ic:add claimType=
         "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"
            optional="false" />
      </ic:informationCard>
      <center>
        <input type="submit" name="InfoCardSignin" value="Log in"
            id="InfoCardSignin" />
      </center>
    </form>
  </body>
</html>
```

4.2. Identity Selector Invocation Parameters

The parameters to the OBJECT and XHTML Information Card objects are used to encode information in HTML that is otherwise supplied as WS-SecurityPolicy information via WS-MetadataExchange when an Identity Selector is used in a Web services context.

4.2.1. issuer (optional)

This parameter specifies the URL of the STS from which to obtain a token. If omitted, no specific STS is requested. The special value "`http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self`" specifies that the token should come from a Self-issued Identity Provider.

4.2.2. issuerPolicy (optional)

This parameter specifies the URL of an endpoint from which the STS's WS-SecurityPolicy can be retrieved using WS-MetadataExchange. This endpoint must use HTTPS.

4.2.3. tokenType (optional)

This parameter specifies the type of the token to be requested from the STS as a URI. This parameter can be omitted if the STS and the Web site front-end have a mutual understanding about what token type will be provided or if the Web site is willing to accept any token type.

4.2.4. requiredClaims (optional)

This parameter specifies the types of claims that must be supplied by the identity. If omitted, there are no required claims. The value of `requiredClaims` is a space-separated list of URIs, each specifying a required claim type.

4.2.5. optionalClaims (optional)

This parameter specifies the types of optional claims that may be supplied by the identity. If omitted, there are no optional claims. The value of `optionalClaims` is a space-separated list of URIs, each specifying a claim type that can be optionally submitted.

4.2.6. privacyUrl (optional)

This parameter specifies the URL of the human-readable Privacy Policy of the site, if provided.

4.2.7. privacyVersion (optional)

This parameter specifies the Privacy Policy version. This must be a value greater than 0 if a privacyUrl is specified. If this value changes, the UI notifies the user and allows them review the change to the Privacy Policy.

4.3. Data Types for Use with Scripting

The object used in the Information Card HTML extensions has the following type signature, allowing it to be used by normal scripting code:

```
interface IInformationCardSigninHelper
{
  string issuer;              // URI specifying token issuer
  string issuerPolicy;        // MetadataExchange endpoint of issuer
  string tokenType;           // URI specifiying type of token to be requested
  string [] requiredClaims;   // Array of required claims
  string [] optionalClaims;   // Array of optional claims
  string privacyUrl;          // URL of the Privacy Policy of the site
  string privacyVersion;      // Version number of the Privacy Policy
  boolean isInstalled;        // True when an Identity Selector is available
                              // to the browser
}
```

4.4. Detecting and Utilizing an Information Card-enabled Browser

Web sites may choose to detect browser and Identity Selector support for Information Cards and modify their login page contents depending upon whether Information Card support is present, and which of the OBJECT and/or XHTML syntaxes are supported by the browser and supported by the Web site. This allows Information Card capabilities to be shown when available to the user, and to be not displayed otherwise.

Detecting an Information Card-enabled browser may require detecting specific browser and Identity Selector versions and being aware of the nature of their Information Card support. A method of accomplishing this for Internet Explorer and Windows CardSpace is described in Appendix B.

4.5. Behavior within Frames

When the object tag is specified in an embedded frame, the certificate of the frame is compared to that of the root frame. For this configuration to work, the scheme, domain, and security zone (for example https, microsoft.com, and Intranet) of the URL of the embedded frame must be the same as that of the root frame. If they do not match, the object tag should not be acted upon. This prevents a form of cross-site scripting attacks.

4.6. Invocation Using the Document Object Model (DOM)

In addition to being invokable using static HTML tags and script code, Identity Selectors can be invoked from script injected into the page using the Document Object Model [DOM]. Invocation from dynamically generated script allows the Web site's requirements to be set dynamically.

4.7. Auditing, Non-Auditing, and Auditing-Optional Cards

- **Auditing Card:** When a managed card with an `ic:RequireAppliesTo` element and no `Optional` attribute or `Optional=false` attribute is used at a Web site, the Request Security Token (RST) sent to the Identity Provider contains a `wsp:AppliesTo` element.

- **Non-Auditing Card:** When a managed card with no `ic:RequireAppliesTo` element is used at a Web site, the Request Security Token (RST) sent to the Identity Provider contains no `wsp:AppliesTo` element.
- **Auditing-Optional Card:** When a managed card with an `ic:RequireAppliesTo` element with `Optional=true` attribute is used at a Web site, the Request Security Token (RST) sent to the Identity Provider contains a `wsp:AppliesTo` element.

5. References

[DOM]
"Document Object Model (DOM)", November 2000.

[ISIP]
A. Nanda and M. Jones, "Identity Selector Interoperability Profile V1.5", July 2008.

[ISIP Guide]
Microsoft Corporation and Ping Identity Corporation, "An Implementer's Guide to the Identity Selector Interoperability Profile V1.5", July 2008.

[.NET3.5]
"Microsoft .NET Framework 3.5 Community Web Site", November 2007.

[WS-MetadataExchange]
"Web Services Metadata Exchange (WS-MetadataExchange)", August 2006.

[WS-SecurityPolicy 1.1]
"Web Services Security Policy Language (WS-SecurityPolicy), Version 1.1", July 2005.

[WS-SecurityPolicy 1.2]
OASIS, "WS-SecurityPolicy 1.2", July 2007.

[WS-Trust 1.2]
"Web Services Trust Language (WS-Trust)", February 2005.

[WS-Trust 1.3]
OASIS, "WS-Trust 1.3", March 2007.

Appendix A – HTTPS POST Sample Contents

The contents of an HTTPS POST generated by a page like the first example in Section 4.1.1 follow:

```
POST /test/s/TokenPage.aspx HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Content-Length: 6478
Content-Type: application/x-www-form-urlencoded
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-sh
ockwave-flash, */*
Accept-Encoding: gzip, deflate
Accept-Language: en-us
Host: calebb-tst
Referer: https://localhost/test/s/
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR
2.0.50727; .NET CLR 3.0.04506.30)
UA-CPU: x86

InfoCardSignin=Log+in&xmlToken=%3Cenc%3AEncryptedData+Type%3D%22http%3A%2F%2F
www.w3.org%2F2001%2F04%2Fxmlenc%23Element%22+xmlns%3Aenc%3D%22http%3A%2F%2Fww
w.w3.org%2F2001%2F04%2Fxmlenc%23%22%3E%3Cenc%3AEncryptionMethod+Algorithm%3D%
22http%3A%2F%2Fwww.w3.org%2F2001%2F04%2Fxmlenc%23aes256-cbc%22+%2F%3E%3CKeyIn
fo+xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2F09%2Fxmldsig%23%22%3E%3Ce%3AEn
cryptedKey+xmlns%3Ae%3D%22http%3A%2F%2Fwww.w3.org%2F2001%2F04%2Fxmlenc%23%22%
3E%3Ce%3AEncryptionMethod+Algorithm%3D%22http%3A%2F%2Fwww.w3.org%2F2001%2F04%
2Fxmlenc%23rsa-oaep-mgf1p%22%3E%3CDigestMethod+Algorithm%3D%22http%3A%2F%2Fww
w.w3.org%2F2000%2F09%2Fxmldsig%23sha1%22+%2F%3E%3C%2Fe%3AEncryptionMethod%3E%
3CKeyInfo%3E%3Co%3ASecurityTokenReference+xmlns%3Ao%3D%22http%3A%2F%2Fdocs.oa
sis-open.org%2Fwss%2F2004%2F01%2Foasis-200401-wss-wssecurity-secext-1.0.xsd%22
%3E%3Co%3AKeyIdentifier+ValueType%3D%22http%3A%2F%2Fdocs.oasis-open.org%2Fwss
%2Foasis-wss-soap-message-security-1.1%23ThumbprintSHA1%22+EncodingType%3D%22
http%3A%2F%2Fdocs.oasis-open.org%2Fwss%2F2004%2F01%2Foasis-200401-wss-soap-m
essage-security-1.0%23Base64Binary%22%3E%2BPYbznDaB%2FdlhjIfqCQ458E72wA%3D%3C
%2Fo%3AKeyIdentifier%3E%3C%2Fo%3ASecurityTokenReference%3E%3C%2FKeyInfo%3E%3C
e%3ACipherData%3E%3Ce%3ACipherValue%3EEq9UhAJ8C9K5l4Mr3qmgX0XnyL1ChKs2PqMj0Sk
6snw%2FIRNtXqLzmgbj2Vd3vFA4Vx1hileSTyqc1kAsskqpqBc4bMHT61w1f0NxU10HDor0DlNVcV
Dm%2FAfLcyLqEP%2Boh05B%2B5ntVIJzL8Ro3typF0eoSm3S6UnINOHIjHaVWyg%3D%3C%2Fe%3AC
ipherValue%3E%3C%2Fe%3ACipherData%3E%3C%2Fe%3AEncryptedKey%3E%3C%2FKeyInfo%3E
%3Cenc%3ACipherData%3E%3Cenc%3ACipherValue%3ErBvpZydiyDzJtzl1%2FjUFX9XAzO1mOR
q0ypPLjh%2FBagXcfZeYwWD57v4Jvn1QwGajadcDASCisazswn1skdkwgmd4IUWJpPMRH7es9zY0U
vnS4ccsakgDcmscq3pDYTrxbSBfhdvrzjDiHC2XCtowOveoHeB51C5N8UAbff18IxCNtkWO8y3wLH
VGdvwaDOSakK%2FK%2Fv1UgXIc51%2FtYvjeFGeGbbSNxo8DTqeDnAMQ%2B4Y%2B1aUGhI%2FtbSr
EyJECkDgtztcxhrumbupKO%2BogWKUTTpSt851xjOFxAMiVaPZ%2FAm8V8H3ZLsR087sX%2FJ%2Bn
bRqze%2BfbdUwimN5pNoJDdMnF%2BEDLass1dPsvhL4EXzuIp5deGBaqAIoaOMEUW7ssuh1PtwkEM
eqwlOzOhu%2FHtwP1qh3D02U59MtyQnJMD5UwIwO7sZJl6%2BPg6Zp9HHtKKUMnkguvFmhyXS4BFS
ZVxPl18i%2B0MLO1um5dejEFd4nwGO%2FmNw6yEI8DdGVjXcYOT6JhPz9rHNh9%2F%2FOj5snJfL6
j2sg0EvIYoRs%2BhT4sdHZ95tGAiwMwT6cFOXbAQZUbYTr1ZOC6XPsfL2CFwiTM3mI%2Blco4Hc%2
F7IakIA8jwAJdtnd2mGuV67ZbY1mzibM1LUApixZj59El83ixctSQbV7iyywQ4IYN2CAq%2BCLMdl
R%2BDHfgEe8O3IVaGBDUEcd2MYimEiA7Yw3NIDrC14SbLzNvU702HpVJMeYv9q6S9xIVGApSrARsw
RFXyMbkMDp5WIQaJEXon7qLcsZONpdlX9bCcmaiikdpxmCeyS638te%2FhGBLmYJSQ0stf7BhA6E0
kwDRgdwsAa88bODiWHek0vDhAN4HlXFZ%2BCxp53L9Mmvy%2FCAOI%2B9OkPL2yxS22yjWQxom%2F
yZuawsK98JHVShsIVmmbKvRM6xJwvHDSzuBAOlQKS%2FMHcFZn8vHZR4lMhm5nL3F%2B%2BumMKh0
vMuKk6JiCqG9OEj996bVIIkLzESU5Z5vT6I1Kr9Brdx8ckDElipdH3x54WVfaItHJTYU%2BsxIR1T
25fi9k%2FOc%2FMX7Q%2B6NSDs4nGqkn4rzqpez9BUWNZw7caVOrDeao85f%2FiDCGymtl0A3JaSZ
dTKfzHLGmUfSkCAlVeisdvB6R7uBw8tR%2BZlgLIGS28wppFlnUYvSK7DnPrzId%2BGfHwLfL6WA%
2FEzBMMgppb5Vi%2BauHq%2BHxpCamlkrcUkzagbwNkGV8TfafkqUvRwJbxRwNVPI%2F%2Fxs%2Fp
```

```
Lcu1dh6eKcmU00%2FNx0zNOScd9XoeEU3zsV78PgvPIBT4EDugdv4bMR6dExXvZBl%2F84b1gOMhK
ZRplF8t6EAc4LCct01ht7VOVNz25NtP27ct9QPrDJc%2FoxihT4Df6NV3l4vlTnu%2B%2BzVB%2BH
JAxNkiO9gx3uLUJM9XEZCDzZKihaBk2y%2F3RhsJpABVneUd%2B3sCRbQXhgKYNBHZyRAUGpMDLhL
qpjoF9x%2FNvUujQ5DBLJafxxzNVshG52jRz%2BikhCNhJDDbeA5MQ8Q7QsYcKDC0DBFsewtWaA%2
FsKxl3JU6hyTotnFS%2FoS2EzbOSvn25qZuBERsZ3w%2B5WMkRzfQadyIYOSv2Df1YoljubDKy1l9
St%2FbCIBgXbVIZKYtQ%2BLyepxxFjrN7cWo2aYFnB6YLurg4USJwhXzcGcvA3%2BR5dRT6Fr37U6
OcHc%2Fz2MaZmn1cQWiDGNxHtRVxEvirBc1x47hWfSRjrKzf3orL5LzgMlYc7Iwclw2rbeWljCqOb
oV3d71ez%2FvNz1pxEMi4w8yUAQL8p%2FRCZ%2BpzvsgORu4RWKWiSwbl7AN0J3jiWShyZgDmxd2O
DDYffXjNiuH1mQWnDTkJX1ig88mqjhOYJEal0W6L0ErwrRIy29tOiAvXZANC8kA1HexulH0e38x8E
IOaVaJtNz9mqrnmnp4GdZ38txV%2BCUeWHOZaHLF4xkdtRxMAu%2FbzQ03YmUOhgxqkTfNzV6Ymne
v2nv5VsyQGJaQsNjb0M4yOe6kX2qNTwKBN2%2Bp%2Fz3f15i8KuGCgBcfP%2BP9xBizBeo7FbFtyo
2pfFhzBPmZeSOJ6kEbF1yQKHYQAT5iZ4SyTIfqqmwGxsQpWMstx3qJF8aW8WFzU1qXcC1LmgClg19
rx9NYFaQshX4f729B9Ue5MX7gTrMgwAnlXty9BsoP7nzGbr3HSXy8pR%2BimuAFW3c2NaQSbjSH5Z
FOr7PZdLHsNVJzFIsaufAwr0CAEtvlPJUt7%2B%2FE5MQsMsVqMoXFmefgdxbvY1Ue6MX1wtuJYY1
PAX7MHTyRUR3RfJDO054EoflVTwNE1fmocUXUh5rtFFuzy2T%2F2Y6pLAARXzo8uslAuH67VkuXv%
2BEMc7e3ogbf5%2BROsgJirZS6qkcYpfEUwqHiQYLnSIP4bt%2BWI5j1bxs7yzcSCkNZ2rd%2FHWr
A41AyGMfYzqxfGcrOaxHsds3JUcByB5Zw17W58GBC32Iusqa69BFTPagEapM0Fb5CbTqXnWTNNB5J
t40BVZvLv3u5oy%2BBRaMKXZhwnbT2WUTp0Ebsn17xvte52B%2BLMlSWJn96Nl5thd%2Ft1D7PlWA
sUvpJAd0UHPizCkY8VIhcXTrsSyEwer2J2I9TQTUosmssFjoP8Lx9qMfXo0eGVmneV8kVBtu4J7N1
QmWfV%2B%2FK8vGbCwW3Gm%2FEUlOO4ZbbK39y0JgNQ7fshxHr5Hdtd%2F6S%2FQkb6NPVDwn7Srh
Y0diWujXz5QlIYBSN7vDfMun3yF%2BGbmMExZ8MkOthuYkgMS9qiFoJGUXGyELsJfxbzdcRE9iyJn
p88L4%2BCtcO3l2JxIhMAgxOZx42RfAiDV1Gbpa4f%2F0urmWQ2VK7uZ%2FlViVrGAJ2kpH0EfwYE
Mb2YYT8FFjogqEpDSJX48BLIh1TE4nMbqQVG1cksCGDc0XyGKaF5Z7Ikw493Xz0JQ0BZvaf2Kceb7
MUZlsU1DSHcQQ9X%2Bxu9RcgUePJEe9BgCMpZ5Kr6r43qyk79noBSgrsSkDhT5sg%2Fc20RHQB8OX
%2BC4r3XGQFWF2m2j0xTc%2Boy14xqUmSB2qJtuWGOXDJspejDRP1GIfFnqDFdqSO3%2FkV9AC5Ee
39iJGv8I%2B5nErtQao645bCytn4B2bJah8R2fXLs8Dd4%2BC2ykxVrLxTUmJaGqd2RK%2F6t1E47
l%2B90Vp4WEzC0CFXXt9XNqdVjo2bZsXbfKQgO2zT2q2qCsgwbxVzIF5y39R%2BrkSkX16uuz3q6w
n3I5RI9M8Hn3DCzzv6Ms4rYxYuiqxaIcb7DgjI2fk1bdyiiRjSxzpCHpK6CWjBD8DPQYdkqGr%2Bs
oWeSvHvPLMSDxEPzwlnaxysRXzKphHUeUa2CCqcpagux2mbKkwHSXemX9I3V3AhePp5XI5eCRiy3
D4%2BcBXOydie94Nz9DIhW749hPiVD9CioAgyqgAzFwCxEEUCXKTzu9xXX4DXg9b3CUfGzwERtY7x
TGT2y%2F9i7r5Xs0lrKi9ftws4JO5v%2Be3WuAEtWv0w%2FVKCl1WwTbV9xtx%2B4RZQ3%2Fewvv%
2F0GqiiSrhiVBGuCDaQs7stwqfkF3vFgGXmmODGTIkIxvYm2fzcEfq4A6LRp5RkYyJyUTF87c56tn
Qa%2Bo3xeiX5WRJybpabrRou09vyWLdlkhcUaBElGWB7iYUJ9bCltByEdNZnuDV%2FXlfnmDARKp8
RVN028czIk57wQMuizgWrM6S9Ku20noDmLgbT554UBf7FnjRWOb%2FF9OJuPpUcARBPrfuqTcOsBq
tZr7AJl3zz%2F53mpyn9rgzw5gBLgkvrdbciabJOAacccTDEB5kEzCLuprC3SlVedhgY%2BMQ5%2F
xgN%2Faf3TtJiBKFvb1V37BlbXXGosnPFcoH8I0XbqW5FSsxmcnpg48poJcB7j5eHq7Y%2F01RLb4
iMmzNap4%2BFg2F3LrwOI0Wk7ueIjgFd5KJ1iTda1ivGU%2Fchr9aTNpM5HiLb2fDW0pZ%2FFBJcI
XxpT9eNY%2FpVj5pnTW2ubpPnBulPOQTLCi1EOxbl33wnhUIfnGiVWJdrls2j3GWgqOnrYUbP%2FX
tNJqIucnMYGqPbcGIF2QRuiwD%2FiTRMvCRCmdCsYE%2FaXjOMhskX7KYC%2B9iG%2FT1wQRbfHSK
WD%2Fpv450OVDsfc1Adq6FCr1LesDNTew%2FF8Z3SiHnWS76OVsNM2SB%2FhMP67iu5UWVkb3%2FQ
qCN0aosOPs2QX0XBCZFmN6p3FhFnXPbAbaGz9y6KzUiUxC03U0fZcToKl4y%2Bw0P4IvxpjVt4t8b
84Q9hiBxd5xu1%2BRE973a%2FyIWO%2Fit1MdUSmxWakxWuGxDnQxwkNCN7ekL%2FQ%2B6FItm86b
w9cc%2FMiI7q2fK7y7YAzM3tmamhF1%2FWJNj1lH0vh%2BhNehJlLlb4Z%2F9ZtxMWV4LVTyrFaF1
zyCEqcKUTk0jc%2FXDwyKZc%2FSV9EOoPk2fVnmzs3WkA74GB%2BWtjdvQjSmnJYtPkMNsikHw%2B
RyB1hTkYbn3iQ6BUiJ0v97j7MVZHxCa1KS3t2gx8H7ts6Tfy5il89xVUdiZwfj0w06g199qlAqUMZ
EWxh0%3D%3C%2Fenc%3ACipherValue%3E%3C%2Fenc%3ACipherData%3E%3C%2Fenc%3AEncryp
tedData%3E
```


An un-escaped and reformatted version of the preceding xmlToken value, with the encrypted value elided, is as follows:

```
<enc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element" xmlns:enc=
"http://www.w3.org/2001/04/xmlenc#">
<enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"
/>
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
<e:EncryptedKey xmlns:e="http://www.w3.org/2001/04/xmlenc#">
<e:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1
p">
```

```
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
</e:EncryptionMethod>
<KeyInfo>
<o:SecurityTokenReference xmlns:o="http://docs.oasis-open.org/wss/2004/01/oas
is-200401-wss-wssecurity-secext-1.0.xsd">
<o:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-mes
sage-security-1.1#ThumbprintSHA1" EncodingType="http://docs.oasis-open.org/ws
s/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary">
+PYbznDaB/dlhjIfqCQ458E72wA=
</o:KeyIdentifier>
</o:SecurityTokenReference>
</KeyInfo>
<e:CipherData>
<e:CipherValue>
Eq9UhAJ8C9K5l4Mr3qmgX0XnyL1ChKs2PqMj0Sk6snw/IRNtXqLzmgbj2Vd3vFA4Vx1hileSTyqc1
kAsskqpqBc4bMHT61w1f0NxU10HDor0DlNVcVDm/AfLcyLqEP+oh05B+5ntVIJzL8Ro3typF0eoSm
3S6UnINOHIjHaVWyg=
</e:CipherValue>
</e:CipherData>
</e:EncryptedKey>
</KeyInfo>
<enc:CipherData>
<enc:CipherValue>
...=
</enc:CipherValue>
</enc:CipherData>
</enc:EncryptedData>
```

Appendix B – Detecting CardSpace Browser Support by Internet Explorer

Script code can detect browser support for Windows CardSpace within Internet Explorer by testing the userAgent string to determine whether the browser version is greater than or equal to "`MSIE 7.0`". A second issue with Internet Explorer 7 is that the Information Card support might not be installed (because Microsoft .NET Framework 3.0 or greater is not installed on the machine). This can be detected within the browser by using the "`isInstalled`" property on the Information Card OBJECT from scripting code. .NET 3.0 installation can be detected on Web servers by testing whether the userAgent string contains "`.NET CLR 3.0`" and .NET 3.5 can be detected by testing for the presence of the string "`.NET CLR 3.5`".

For example, the userAgent string on a Windows XP machine using IE 7 and .NET 3.0 will contain at least these elements:

```
MSIE 7.0; Windows NT 5.1; .NET CLR 3.0.04506.30
```

The original release of .NET 3.5 contained the userAgent string element:

```
.NET CLR 3.5.21022
```

.NET 3.5 Service Pack 1 uses the userAgent string element:

```
.NET CLR 3.5.30707
```

Appendix C – Windows CardSpace .NET Framework 3.5 Service Pack 1 Constraints

This section documents any additional constraints imposed by the Windows CardSpace .NET Framework 3.5 Service Pack 1 implementation.

- In reference to ISIP Section 9, use of the XHTML syntax at Relying Parties without certificates (Web sites using the HTTP scheme) is not supported.