# Understanding Digital Identity

Who are you? It's a simple question, but it doesn't have a simple answer. The way you represent your identity changes as you move through the world. When you present your passport at an airport's Immigration Desk, you're a citizen of some country. When you show your driver's license to a policeman who's stopped you for speeding, you're a legal driver who resides in some locality. When you use your credit card to pay for a best-selling novel at a bookstore, you're a customer with a particular account number. Different contexts require different identities, each of which is expressed in a different way and provides different information.

All of these contexts have well-understood ways for you to establish your identity. Yet, in one very important context—the networked world—identity is currently a much more muddled thing. Just as in the physical world, all of us have a variety of *digital identities*, and they're expressed in different ways. Today, however, there's no consistent way to deal with this portfolio of digital identities. Instead, we're left struggling in a complex, confusing, and insecure environment.

Yet different kinds of digital identities will always be necessary—no single identity will suffice. And the reality is that these identities will always be provided by a range of different sources—no single identity provider will suffice, either. This means that the solution is not to mandate a single system for digital identity, but rather to find a coherent way to use multiple digital identity systems. What's required is a system of systems—a *metasystem*—focused on identity.

Working with others, Microsoft has played a major role in defining this standards-based identity metasystem. Microsoft is also adding new capabilities to Windows to help make the identity metasystem a reality. Windows CardSpace, originally code-named "InfoCard," lets any Windows application, including Microsoft technologies such as the next release of Internet Explorer and those created by others, give its users a common way to work with digital identities.

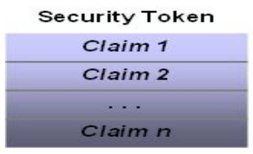
Windows is a widely used operating system, and so Cardspace is an important part of making the identity metasystem real. Still, this solution can't succeed unless other organizations also implement it. Accordingly, Microsoft is actively encouraging the creation and use of software that can participate in the identity metasystem.

## Describing Digital Identity

Just as in the real world, there are good reasons to use different digital identities in different contexts. It's common, for instance, to associate different information with each identity. An identity that you use with Amazon might allow access to your credit card number, while one used with MySpace.com does not. The rules for getting each identity are also different. Getting a digital identity at Amazon is easy: just make up a username and password. Getting a digital identity at your employer is probably somewhat more difficult, since, at a minimum, it requires the approval of the administrators who run your company's network.

## Representing Digital Identity: Security Tokens

Despite their diversity, digital identities all have one important thing in common: when transmitted on the network, every digital identity is represented by some kind of *security token*. A security token is just a set of bytes that expresses information about a digital identity. As shown in Figure 1, this information consists of one or more *claims*, each of which contains some part of the total information conveyed about this identity. A simple security token might include only a claim containing a username, while a more complex one might include claims containing a user's first name, last name, home address, and more. Security tokens for some digital identities might also include claims that contain sensitive information such as credit card numbers.



With most security tokens, some information is provided in order to prove that these claims really do belong to the user who's presenting them. The security tokens that represent digital identities commonly provide some kind of proof that allows a receiver of the token to verify that this token really does represent the person or organization with that identity.

For example, your driver's license includes your name, your age, perhaps your picture, and other information, all asserted to be correct by some governmental organization. A digital identity that expressed this information could be useful for various things, such as proving that you're 21 or older, or that you really do wear glasses. Similarly, each of your credit cards carries a card number and expiration date along with your name. Just as these cards are useful in the physical world, it would also be useful to create a digital identity for each card that could be used to generate a security token carrying the proper claims.

Life would be simpler if a single digital identity, represented with a single security token format, could be used for everything. Yet for the same reasons that we have different identities in the physical world, we will always have different identities in the digital world. The challenge is to create, use, and manage these diverse digital identities in an understandable and effective way. This is exactly the problem that the identity metasystem addresses. Rather than invent yet another technology for creating and representing digital identities, the identity metasystem instead provides a consistent way to work with multiple digital identities, regardless of the kinds of security tokens they use.

## What Windows CardSpace Provides

The multiple digital identities that we use come from several different sources, and they are expressed in a variety of ways. In other words, we typically rely on a number of different digital identity systems, each of which may use a different underlying technology. To think about this diversity in a general way, it's useful to define three distinct roles:

- **User**—Also known as the *subject*, the user is the entity that is associated with a digital identity. Users are often people, but organizations, applications, machines, and other things can also have digital identities.
- **Identity provider**—An identity provider is just what the name suggests: something that provides a digital identity for a user. For the digital identity assigned to you by your employer, for example, the identity provider is typically a system such as Active Directory. For the digital identity you use with Amazon, the identity provider is effectively you, since you define your own username and password. Digital identities created by different identity providers can carry different information and provide different levels of assurance that the user really is who he claims to be.
- **Relying party**—A relying party is an application that in some way relies on a digital identity. A relying party will frequently use an identity (that is, the information contained in the claims that make up this identity's security token) to authenticate a user, and then make an authorization decision, such as allowing this user to access some information. A relying party might also use the identity to get a credit card number, to verify that the same user is accessing it at different times, or for other purposes. Typical examples of relying parties include Internet websites such as online bookstore and auction sites, and any application that accepts requests through Web services.

Given these three roles, it isn't difficult to understand how Windows CardSpace and the identity metasystem can support any digital identity. Figure 2 shows the fundamental interactions among the three roles.

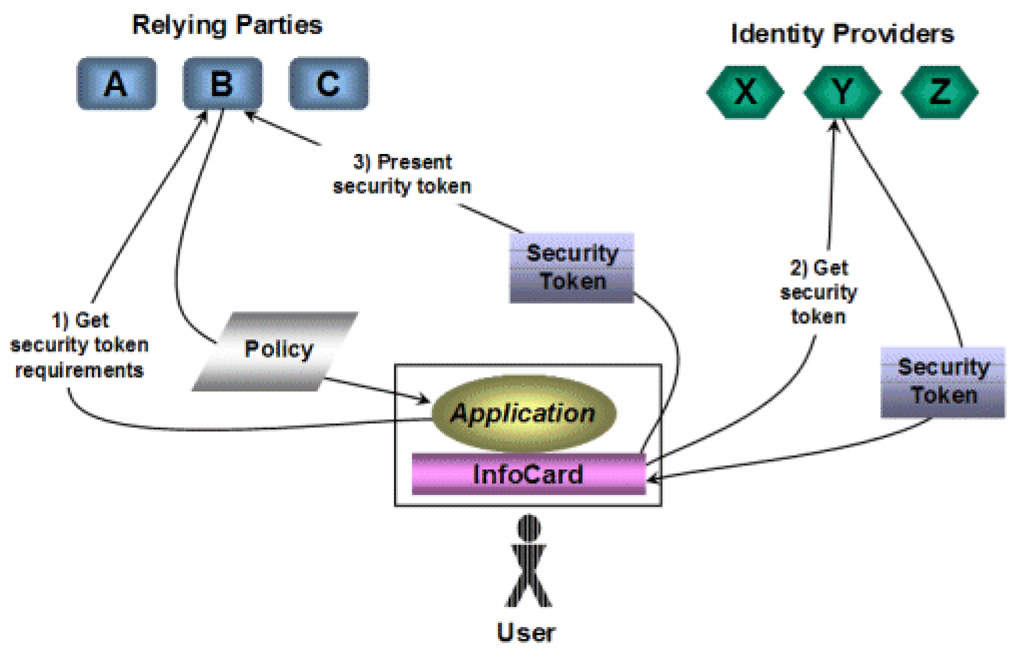


**Figure 2. Interactions among the user, identity provider, and relying party roles**

As Figure 2 suggests, a user might rely on an application that supports CardSpace, such as a Web browser, to access any of several relying parties. She might also be able to choose from a group of identity providers as the source of the digital identity she presents to those relying parties. Whatever choice she makes, the basic exchange among these parties has three steps:

1. First, the application gets the security token requirements of the relying party that the user wishes to access. This information is contained in the relying party's *policy*, and it includes things such as what security token formats the relying party will accept, and exactly what claims those tokens must contain.
2. Once it has the details of the security token this relying party requires, the application passes this information to CardSpace, asking it to request a token from an appropriate identity provider.
3. Once this security token has been received, CardSpace gives it to the application, which passes it on to the relying party.

The relying party can then use this token to authenticate the user or for some other purpose.

This high-level view illustrates the most important aspects of the process. They include the following:

- Windows CardSpace and the identity metasystem are entirely agnostic about the format of the security token that's requested from an identity provider and passed on to a relying party. In fact, CardSpace typically isn't even aware of what format this token is in. Because of this, CardSpace can work with any digital identity system, using any type of security token, including simple usernames, X.509 certificates, Kerberos tickets, SAML tokens, or anything else. This allows CardSpace and the identity metasystem to be used together with whatever digital identity technologies are in place. It also allows plugging in yet-to-be-created digital identity systems that might appear in the future.
- All of the exchanges defined by the identity metasystem and implemented by CardSpace are done using open, published protocols.

In the most general scenario, a relying party's policy is described using WS-SecurityPolicy, that policy is retrieved using WS-MetadataExchange, a security token is acquired using WS-Trust, and that token is conveyed to the relying party using WS-Security (all if the WS-* protocols necessary to enable the secure exchange of identity tokens in the identity metasystem are (or soon will be) submitted to standards bodies.

In the simpler (and probably more common) scenario of a Web browser interacting with a website, the relying party's policy can be expressed using HTML, and both this policy information and the security token can be exchanged with the site using HTTPS. While interactions with an identity provider still depend on

WS-Trust, a website isn't required to implement any of the WS-* specifications in order to act as a relying party.

In either scenario, working with CardSpace does not require relying parties or identity providers to implement any proprietary protocols.

As Figure 2 indicates, CardSpace is useful only if identity providers and relying parties implement the protocols used by the identity metasystem. While Microsoft has led the effort to define the metasystem, and it has created Windows CardSpace to provide key metasystem components for Windows, the effort cannot succeed without participation from other organizations.

## Using Windows CardSpace: Example Scenarios

The clearest way to get a sense of how CardSpace and the identity metasystem might be used is to look at some typical examples. Think about what happens, for instance, when you access an online merchant such as an Internet bookstore. In the simplest case, no digital identity is involved—anybody can browse through the books on offer, without telling the merchant who they are. When you try to place an order, however, you need to log in, which requires providing a digital identity. Today, you'll most likely do this by entering a username and password, both of which you've chosen yourself. If this online merchant also supports CardSpace and the identity metasystem, it will provide another option for identifying yourself: using an information card. To allow this, the merchant might have a separate button on its login screen that you can click to log in with an information card, rather than entering your username and password.

Clicking this button causes the browser to use CardSpace to log in to this site. As usual, you'll be presented with the CardSpace selection screen, and you'll choose one of the cards in order to identify yourself to this merchant. It's likely (although not required) that the information card you choose in this case will be one created by the self-issued identity provider—that is, one that you created yourself. Since all the site needs to do at this point is identify you as a unique customer, this simple form of digital identity is sufficient. To pay for your purchases, you might enter your credit card information and mailing address on a Web form as usual.

In this simple scenario, CardSpace provided a way to log in to an online merchant without using a password. This is useful, and it's a step forward for digital identity on the Internet. Yet it's only the beginning of how digital identity can be used. In this example's payment step, for example, it might be possible to use an information card to send your credit card information to the site. Suppose that the company issuing your credit card offers an identity provider that allows you to request an information card corresponding to your physical card. Rather than entering the credit card information into a Web form, the site would alternatively offer a button on the payment screen that allows you to provide an information card. Clicking this button would result in your system displaying the CardSpace selection screen, letting you choose from all of the cards that will be accepted for payment at this site. Clicking on one of the cards would cause CardSpace to contact the identity provider of this card's issuer, get a security token containing your credit card information, and then present this to the online merchant.

As this example shows, digital identities aren't just for proving who you are. Like the physical identities represented by the various cards in your wallet, digital identities can be used to make payments, or for other purposes. For instance, suppose that the organization issuing driver's licenses in your locality made available an identity provider. You could now have a digital identity that allowed you to verify a variety of personal information to any relying party. In the United States, for example, customers must prove that they're 21 or older in order to purchase wine. An online wine merchant could require its customers to present the CardSpace version of their driver's license, which includes their birth date, perhaps also accepting a CardSpace version of a credit card to pay for the wine.

This driver's license identity provider could also provide other kinds of information cards. For example, some people might wish to have a way to prove that they're 21 or older, without revealing their name or other identifying information. A driver's license identity provider knows your age, and so along with an information card containing everything on your license, it might also offer a simpler card containing nothing more than your age. For people who qualified, this identity provider might even offer an information card that contained just an assertion that you are 21 or older, avoiding revealing something as potentially sensitive as your age. Even though they're called "digital identities," there's no requirement that they must contain any information at all that can actually be used to identify a particular user. For many uses, all that's needed is a way to assert a claim, such as being 21 or older, that's backed up by a trusted authority.

There are many more examples of how Windows CardSpace and the identity metasystem might be used. A provider of mobile phone service might offer an information card that its subscribers can use to charge online purchases to their phone bill. An employer might give its employees multiple digital identities, each with its own information card, for use on the company network. One identity could be used for normal access, while another, perhaps stripped of everything other than the fact that this person is an employee, might exist solely for the purpose of making anonymous suggestions to company management. Just as identities in the real world carry different information and are used for different purposes, digital identities can also be used in a variety of ways. The goal of CardSpace and the identity metasystem is to make this broad use possible.

## 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 Server (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.

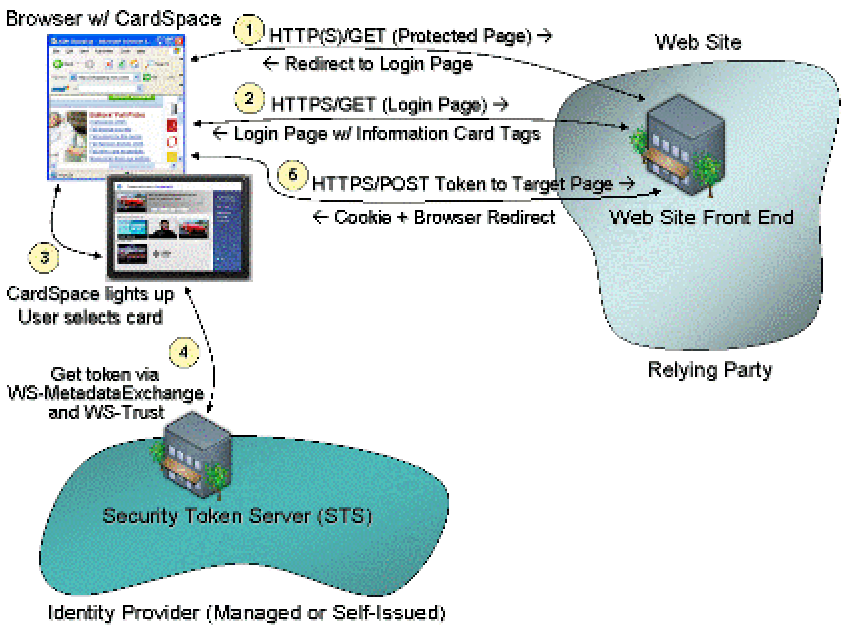

Browser w/ CardSpace
1 HTTP(S)/GET (Protected Page) →
← Redirect to Login Page
Web Site
2 HTTPS/GET (Login Page) →
← Login Page w/ Information Card Tags
5 HTTPS/POST Token to Target Page →
← Cookie + Browser Redirect
Web Site Front End
3
CardSpace lights up
User selects card
4
Get token via
WS-MetadataExchange
and WS-Trust
Relying Party
Security Token Server (STS)
Identity Provider (Managed or Self-Issued)

**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 [CardSpace-Guide] 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.

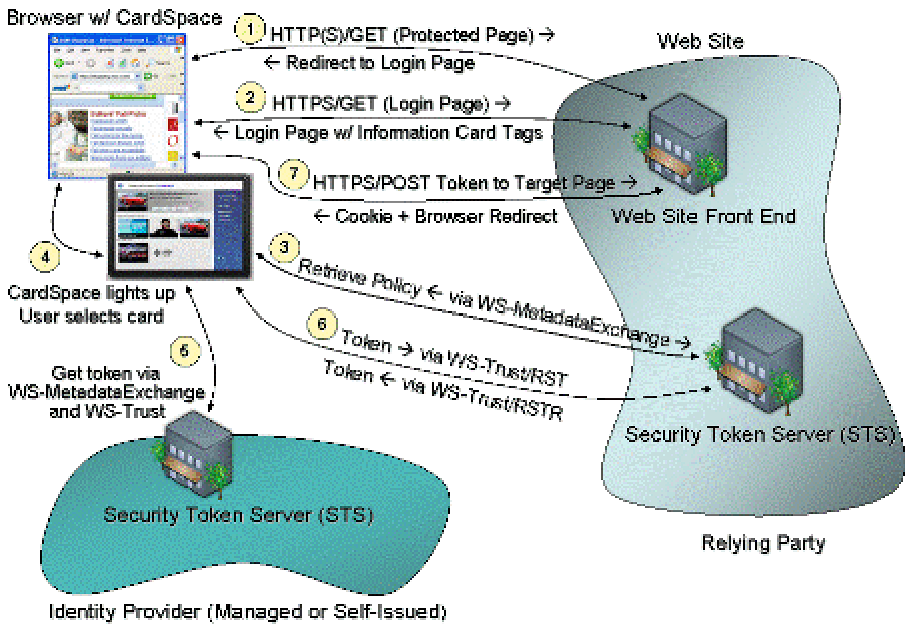

Browser w/ CardSpace
1 HTTP(S)/GET (Protected Page) →
← Redirect to Login Page
Web Site
2 HTTPS/GET (Login Page) →
← Login Page w/ Information Card Tags
7 HTTPS/POST Token to Target Page →
← Cookie + Browser Redirect
Web Site Front End
4
CardSpace lights up
User selects card
3 Retrieve Policy ← via WS-MetadataExchange →
6 Token → via WS-Trust/RST
Token ← via WS-Trust/RSTR
5
Get token via
WS-MetadataExchange
and WS-Trust
Security Token Server (STS)
Relying Party
Security Token Server (STS)
Identity Provider (Managed or Self-Issued)

**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.

# 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>
  <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. If omitted, the value "<issuer>`/mex`" is used. 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 is described in Appendix B.