A Guide to Interoperating with Information Cards in Windows CardSpace v1.0

December, 2006

Authors

Microsoft Corporation
Ping Identity Corporation

Copyright Notice

Abstract

This document is intended for developers and/or architects who wish to design identity systems and applications that can interoperate with the Windows CardSpace system. The Windows CardSpace system allows users to manage their digital identities from different identity providers, and employ them in various contexts to access online services. This Guide describes a model, built upon the mechanisms described in [WS-Trust] and [WS-SecurityPolicy], to allow digital identity to be integrated into a user-centric identity framework. The mechanisms described in this document provide the framework for an identity metasystem that promotes interoperability between identity providers and relying parties with the user in control. The interactions between the Windows CardSpace system and a relying party or an identity provider are illustrated, and the message exchanges with an identity provider are described in detail. This document is intended to be read alongside the document entitled "A Technical Reference for Information Cards in Windows CardSpace v1.0" [CardSpace-Ref] which provides the normative schema definitions and behaviors referenced by this document.

Status

This draft of the Guide reflects what is implemented by the Windows CardSpace v1.0 system that ships in the Windows Vista release and is available as a download to run on Windows XP SP2 or higher and Windows Server 2003. The documented behavior and schema described here are informative; the normative definitions can be found in [CardSpace-Ref].

Table of Contents

1. Introduction

Identity is fundamental to enabling interactions in everyday life. The same is true of the digital world as well where digital identity is fundamental to enabling digital interactions in an interconnected online world. Digital identities are used to authenticate parties to each other in the online world. Knowing, with a high degree of assurance, who one is interacting with is a key element in deciding whether to trust the other party and for what.

Broadly speaking, a *digital identity* of a subject is a set of *claims* asserted by a *claims authority* about the subject (see glossary of terms in Appendix B). Claims are generally communicated in signed security tokens, and may represent identifying and other personal information about a subject. Users will typically have a portfolio of digital identities analogous to the multiple forms of identities they employ in the physical world – drivers' licenses, other government-issued identity cards, credit cards, company affiliation cards such as frequent flyer cards, etc. The use and acceptance of a digital identity in any given context is usually an intersection of a user's choice to offer an identity based on its appropriateness to the context, and the recipient's choice to accept that identity based on its requirements and willingness to trust the claims authority that is making the claims inherent in the digital identity. Hence it is important to create a system that allows users to employ digital identities issued by different authorities in contexts of their choosing through a consistent and understandable user interface. The system should be capable of handling all forms of digital identities regardless of the underlying identity technologies at play.

Information Card presents a model that allows users to manage a portfolio of identities from various authorities, and employ them in various contexts when acceptable to access online services. Windows CardSpace is a specific implementation of the model presented by Information Cards. It is grounded in the Web services architecture which is based on a suite of specifications that define rich functions that may be composed to meet varied service requirements. A crucial application for these services is to establish a framework in which consumers of user identities can ask for exactly what they need, and providers of identities can furnish the needed identity with intermediation by the user when appropriate.

In the Web services architecture, digital identities are encoded as *security tokens* containing claims about a user made by an *Identity Provider* (IP) and presented to a *Relying Party* (RP). The security token presented may be used for authenticating the user and/or providing authorized access to services offered by the relying party. Furthermore, relying parties can express their identity and other security requirements in the form of security policy that can be queried by client applications through which the user desires to access the services offered by the relying party.

It should be noted that just as claims about users may be asserted by a 3rd party identity provider, some claims could be self-asserted by users acting as their own identity providers. Ultimately, it is upto the relying party to determine if it is willing to trust and accept it. It turns out that such self-issued identities are commonly used and find applicability in many everyday online interactions. For example, when users visit an online retailer site and must create new user accounts in order to purchase merchandise from that site, they would typically fill in one or more online forms divulging personal information to the site. In these circumstances, the online retailer site is usually willing to accept the users' *self asserted* personal information to register and create accounts for them. Usually, what is important for the relying party in such scenarios is that an user can prove on a repeat visit that she is the same user that registered.

To help users organize their various digital identities, the Information Card model introduces the notion of an "information card" which is an embodiment of a digital identity that the user can visualize, examine and reason about in user interfaces. Each information card corresponds to an identity provider and represents a digital identity for the user issued by that identity provider. Multiple digital identities for a user from the same identity provider would be represented by different information cards. Users may have a collection of information cards representing the various digital identities they have, some self-issued and others issued by 3rd party identity providers. Note that an information card itself is ***not*** the security token that is used to carry identity claims in Web service protocols, rather it is an artifact that represents the token issuance relationship of the user with the corresponding identity provider. An actual security token with specific claims can be requested from the identity provider when needed based on the information card. In other words, information cards help to provide a concrete visualization of a user's identities on a user interface in digital interactions much like the cards one carries inside one's wallet/purse for everyday physical interactions.

Further, to help users select from their various digital identities in different contexts, the Information Card model introduces the notion of an *Identity Selector* as an architectural component in the identity metasystem. It is the processing engine that determines which of a user's information cards are capable of meeting a relying party's requirements. It also provides a consistent user interface for users to visualize, examine and reason about their digital identities, and select one for use. When a client application (i.e., the user agent) needs a suitable security token to satisfy the security requirements of a target service it interacts with, it invokes the identity selector component to obtain the appropriate security token representing the user identity. The identity selector puts users in control of the use of their identities by applications in various contexts.

1.1. Goals of Information Card Model

The identity selector can use information cards from any identity provider of the user's choice, and offer those identities under user control to applications acting as user agents. The identity selector interoperates with the identity provider for an information card using open protocols. The primary goal of this Guide is to document and describe how a relying party expresses its identity requirements to a client such that the identity selector can process them, and how the identity selector interacts with identity providers to obtain security tokens that fulfill those requirements. We hope that this will enable any identity provider or relying party to interoperate using the Information Card model for the purpose of identity-based Web service interactions.

The following list identifies the key goals of the Information Card model:

- Enable use of digital identity in the form of security tokens carrying claims as authentication and/or authorization data using Web service mechanisms.
- Allow users flexibility in their choice of digital identities they wish to employ, and put users squarely in control of the use of their identities in digital interactions.
- Support cryptographically verifiable but human-friendly identification of the recipients of a user's digital identities.
- Enable interoperability with identity providers and relying parties using open protocols to allow an identity ecosystem to thrive.
- Remain agnostic of specific security token types and claim types so as to effectively be a conduit for flow of identity information between identity providers and relying parties under user control.
- Safeguard user privacy by providing privacy-friendly identity mechanisms to help thwart tracking of users' online behavior and unsolicited collusion.
- Provide a simple identity provider to allow users to construct and employ self-issued identities in Web service interactions when acceptable.

1.2. Information Card Usage Model

This section describes the overall model for using digital identity in the form of security tokens for authentication, authorization or any other purpose. The WS-SecurityPolicy, WS-MetadataExchange and WS-Trust Web service specifications define mechanisms for expressing security requirements and obtaining security tokens to satisfy those requirements. The Information Card model builds on this foundation by describing how these mechanisms are combined to enable rich expression of identity requirements and fulfillment of those requirements. The model promotes interoperability between identity providers and relying parties under user control.

The model described here can be used with:

- A dedicated rich client application accessing a *Web service* as the relying party, or
- A generic Browser client accessing a *Web site* as the relying party.

For the first case, a Web service can use the policy assertions defined in [WS-SecurityPolicy] to express its security token requirements and the necessary set of claims they must carry in order for it to accept incoming requests. A rich client application can query and learn the Web service policy using [WS-MetadataExchange] prior to requesting service. For the second case, a Web site can use the mechanism defined in [CardSpace-Browser] using HTML tags to express its security token requirements. A Browser client acting as the user agent can learn the Web site policy by interpreting the HTML content of the queried page.

In either case, the client evaluates the relying party policy and acquires the necessary security token(s) from suitable identity providers using the token issuance mechanism described in [WS-Trust]. This aspect of the model is the same for either case, and is the primary focus of this document. The tokens are then presented to the relying party using the mechanisms defined in [WS-Security] or [CardSpace-Browser] depending on whether the relying party is a Web service or Web site, respectively.

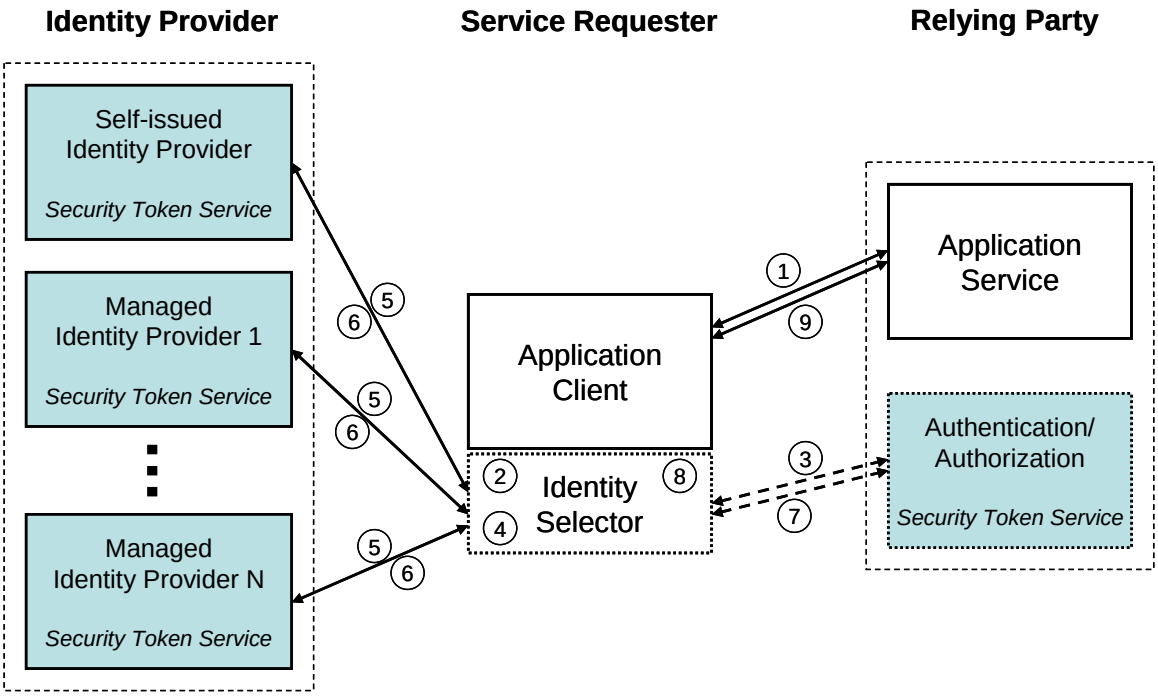

Figure 1 Information card interaction model

Figure 1 above illustrates the information card interaction model for using digital identity in the form of security tokens in a simple canonical scenario. The model consists of a service requester in the form of a client application running on a client system, a relying party in the form of a Web service (or Web site) that the user wishes to access through the client application, and one or more identity providers that can issue security tokens. The relying party may optionally maintain profile information about a user, but that is distinct from the digital identity employed by the user to gain access to the service.

The relying party may optionally delegate to an associated service, shown as the "Authentication/Authorization Security Token Service" in the figure, to authenticate and/or authorize a user's identity. Note that this is purely a service deployment choice and **not** a required configuration. The application service could just as well perform those functions itself instead of delegating. If that were the case, then the extra hop represented by steps 3 and 7 would be absent in Figure 1. Such choices in service deployment can be suitably reflected in the relying party policy causing the appropriate WS-Trust exchanges to occur between the service requester and the intermediate security token services employed by the relying party before the final request reaches the application service.

The user, interacting through the identity selector on the service requester, may have identities issued by one or more identity providers. Each such digital identity of the user is represented by an information card that the identity selector can process. An information card endows the identity selector with the ability to request and obtain security tokens from

the corresponding identity provider when the user selects that digital identity for use in a given interaction context.

Each identity provider, shown as “Managed Identity Provider 1 through N” in the figure, runs a Security Token Service (STS) to which a requester can submit security token requests. The security token service can issue security tokens containing the requested claims after the requester has provided suitable proof of authentication as required by the identity provider’s security policy. Note that a simple identity provider, shown as the “Self-issued Identity Provider” in the figure, may be used in the Information Card model to allow users to issue self-issued security tokens.

Let us assume that the user has previously obtained one or more information cards from various identity providers which are available to the identity selector running on the service requester. The sequence of actions that occurs in the model depicted in Figure 1 where the user wants to access the relying party service through the client application is as follows:

1. The client application obtains the security policy of the relying party using the mechanisms described in [WS-MetadataExchange] or [CardSpace-Browser] depending on whether the relying party is a Web service or Web site, respectively. The relying party policy requires that the requester present a security token issued by either an identity provider or the delegate Authentication/Authorization STS.
2. The client application requests the identity selector to produce a security token that can satisfy the relying party policy.
3. (OPTIONAL) If a token is required from the delegate STS, then the identity selector obtains the security policy of the delegate STS using metadata exchange. This step is iteratively repeated for as many intermediate STS as is necessiated by the relying party deployment configuration. At the end, the client has a policy that requires a security token with a specified set of claims issued by a specific identity provider (could be the self-issued identity provider).
4. The identity selector displays the matching information cards which can satisfy the relying party policy, and the user selects and approves one for use.
5. The identity selector requests and obtains the security policy of the identity provider STS corresponding to the selected information card using metadata exchange. The security policy specifies the security binding to use for requesting tokens.
6. The information card specifies the required credential to use for authenticating the user to the identity provider. The identity selector authenticates the user to the identity provider STS using the credential specified in the information card, and requests a security token with the desired claims as specified by the relying party using the mechanisms described in [WS-Trust].
7. (OPTIONAL) The identity selector presents the token to the delegate Authentication/ Authorization STS, and requests a security token for the relying party service using the mechanisms of [WS-Trust].
8. The identity selector returns the requested security token to the client application.
9. The client application presents the token obtained in step 6 (or optionally step 7) to the relying party service to gain access.

1.2.1. Web service interactions

When the relying party is a Web service, Figure 2 below shows the interactions between the participating entities and the sequence of message flows between them. For convenience, the optional Authentication/Authorization STS is omitted from the interactions shown, and

the relying party application service is assumed to process the security token presented by the service requester by itself.

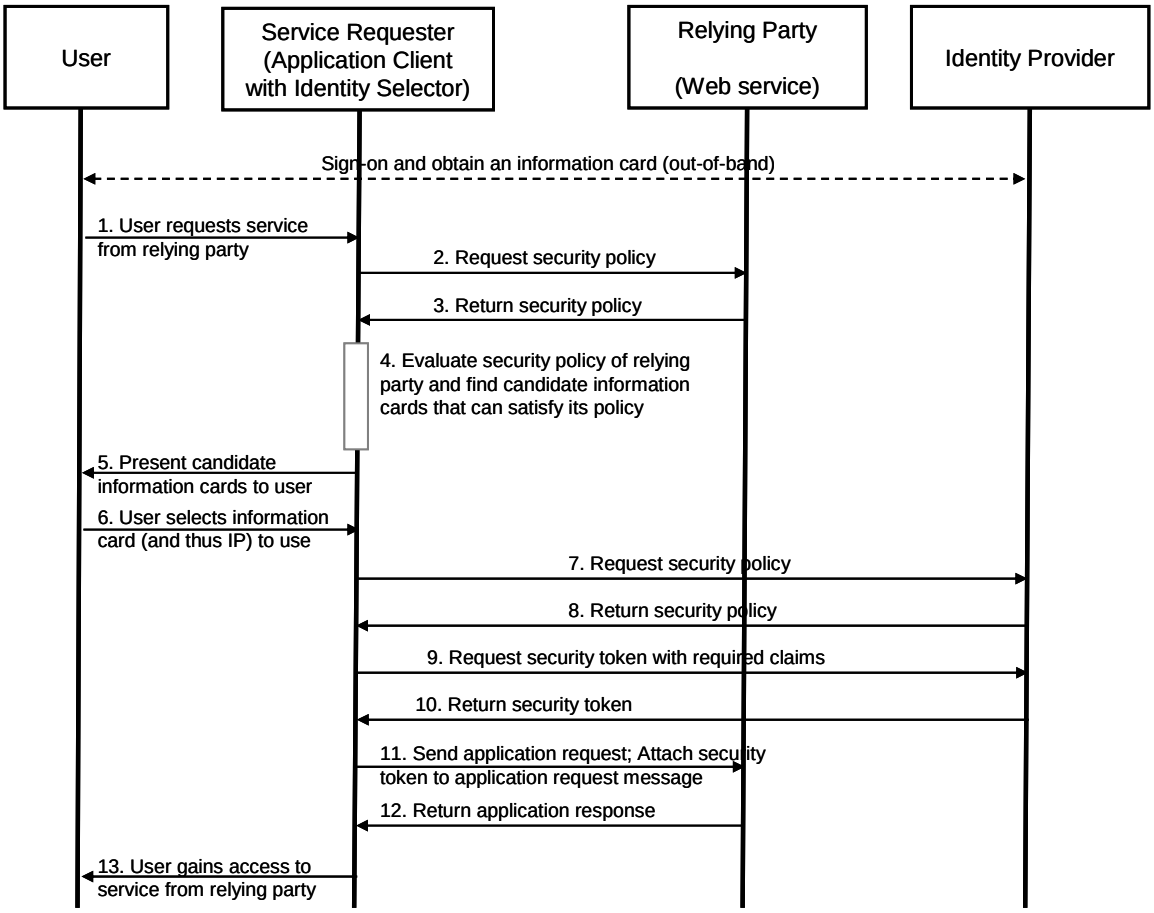

Figure 2 Message sequence for Web service interactions

Note that the above sequence describes the interactions when a dedicated client application is used to access a Web service. The interaction sequence illustrated in this figure provides the framework for the details of the Information Card model described in the remainder of this document.

1.2.2. Web site interactions

When the relying party is a Web site, Figure 3 below shows the interactions between the participating entities and the sequence of message flows between them. As before, the optional Authentication/Authorization STS is omitted from the interactions shown, and the relying party Web site is assumed to directly process the security token presented by the service requester.

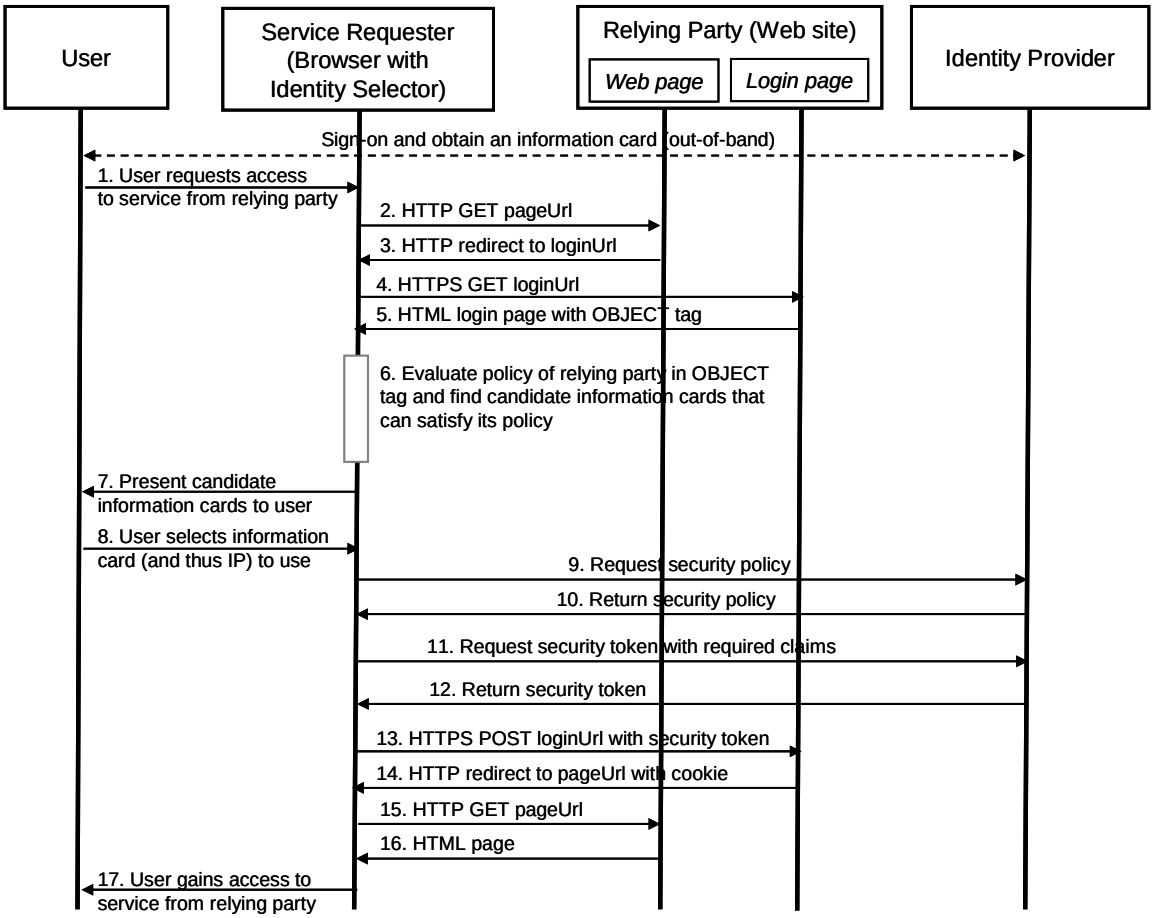

Figure 3 Message sequence for Web site interactions

This interaction model, using a lightweight HTTP mechanism for accessing the relying party with a Browser client, is described in more detail in [CardSpace-Browser].

1.3. An Example

Let us illustrate the information card based interactions described in the previous section with a real world example using a Web service and a dedicated rich client. John Kane is an employee of Fabrikam, Inc. Fabrikam has a partnership with Blue Yonder Airlines for making travel arrangements for its employees and purchasing tickets at specially discounted prices. Fabrikam has issued all its frequent traveler employees, including John, information cards to prove that they are employees of Fabrikam. It also runs a STS at the address *http://fabrikam.com/employee/sts* which issues security tokens for the issued information cards. Fabrikam has also given those employees smart cards to use as strong two-factor credentials for authenticating to the employee STS when using their information cards on the road.

Employees of Fabrikam use a special travel reservation smart client application for requesting travel arrangements from Blue Yonder Airlines which runs a travel portal and airline reservation service at the address *http://www.blueyonderairlines.com/travel*. When John runs the smart client reservation application on his personal computer to make travel reservations with Blue Yonder Airlines, the following sequence of interactions occur:

- The travel reservation client application obtains the security policy of the airline reservation service at *http://www.blueyonderairlines.com/travel* using metadata

exchange. The travel portal service policy requires that the client application submit a security token issued by the user's employer STS, namely the Fabrikam STS at *http://fabrikam.com/employee/sts* (There is a trust relationship between the airline reservation service and each of the partner company STS with which it federates).

- The travel reservation client application requests the identity selector component on John's personal computer to produce a security token that can satisfy the reservation service policy. The identity selector displays the matching information cards, namely the information card given to John by his employer, and John selects and approves it for use.
- The identity selector on John's personal computer then obtains the security policy of John's employee STS at *http://fabrikam.com/employee/sts* using metadata exchange to determine the security binding to use when requesting the security token.
- The employer issued information card selected by John specifies the required authentication mechanism to be X.509 certificate based, and the credential selector in the information card provides a hint for John to insert his smart card given to him by his employer. The identity selector prompts John to insert his corporate smart card into the reader and enter his PIN.
- The identity selector now authenticates to the Fabrikam employee STS at *http://fabrikam.com/employee/sts* using the X.509 certificate from John's smart card, and requests a security token with the required claims specified by the airline reservation service. Upon successful authentication, it receives the security token.
- The identity selector then hands the requested security token to the travel reservation application running on John's personal computer.
- The travel reservation application running on John's personal computer then presents the token obtained from the identity selector and presents it to the travel portal service along with proof-of-possession to gain access.
- Now, John can look at possible travel choices and request reservations.

Although the above example describes a dedicated client application used to access a Web service, the model can also be used in a lightweight manner with a Browser client accessing a Web site using the HTTP protocol as shown in Section 1.2.2. This lightweight usage is described in [CardSpace-Browser].

2. Using This Document

In this document we will cover what you need to know and the steps you need to take to support information cards either as a relying party or as an identity provider. Following is a brief navigational summary of the parts of this document that are pertinent for each role.

In order to support information cards, a relying party using Web-services based application will need to:

- Support identification of its organization using X.509 certificates with logotypes, whenever possible, to allow end users to clearly identify who they are dealing with. It is highly recommended that "extended validation" certificates be employed for this purpose. This is described in Section 3.1.
- Support expressing its security requirements, including its security token requirements, using the policy assertions described in [WS-SecurityPolicy]. This is described in Section 3.1.1.

- Support the retrieval of its service metadata, including its WSDL and policy, using the mechanism described in [WS-MetadataExchange]. This is briefly described in Section 3.5.
- Support the submission of security tokens bound to application mesasges by a service requester using the mechanisms specified in [WS-SecurityPolicy]. This is briefly described in Section 3.6.
- Optionally, when appropriate, support self-issued security tokens and the strong cryptographic keys in such tokens as user credentials instead of passwords. This is described in Appendix A.

In order to support information cards, an identity provider will need to:

- Support issuing information cards to its users using the format and mechanism described in Sections 4.1.
- Support the mechansim described in [WS-Trust], using the *RequestSecurityToken* and *RequestSecurityTokenResponse* protocol messages, for issuing security tokens based on an information card. An identity provider can, however, issue any type of security token that is acceptable to a relying party since the identity selector on the service requester is token agnostic. This is described in Section 4, and the message exchanges are detailed in Section 5.
- Although not required, it is highly recommended that the identity provider should support the specific extensions to WS-Trust protocol elements defined by the Information Card model to support the identity metasystem goals of user control and privacy. This is described in Sections 4.3.
- Support expressing the security requirements of its security token service using the policy assertions described in [WS-SecurityPolicy], and support the retrieval of that policy using the mechanism described in [WS-MetadataExchange]. This is described in Section 5.1.
- Support one or more of the credential mechanisms described in Section 5 to allow users to authenticate to the security token service. When appropriate, instead of passwords, support self-issued security tokens and the strong cryptographic keys in such tokens as user credentials. This is described in Section 5.5.

For brevity of the examples used for illustration in this document, we list here in Table 1 the XML namespaces and corresponding prefixes used throughout the document.

Table 1: Prefixes and XML namespaces used in this document

Prefix	XML Namespace	Reference(s)
S	http://www.w3.org/2003/05/soap-envelope	SOAP 1.2 [SOAP 1.2]
xs	http://www.w3.org/2001/XMLSchema	XML Schema [Part 1, 2]
ds	http://www.w3.org/2000/09/xmldsig#	XML Digital Signatures
ic	http://schemas.xmlsoap.org/ws/2005/05/identity	Information Card Technical Reference [CardSpace-Ref]
wsid	http://schemas.xmlsoap.org/ws/2006/02/addressingidentity	Identity Extension for Web Services Addressing [Addressing-Ext]

wsx	http://schemas.xmlsoap.org/ws/2004/09/mex	WS-MetadataExchange [WS-MetadataExchange]
wsa	http://www.w3.org/2005/08/addressing	WS-Addressing [WS-Addressing]
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	WS-SecurityUtility
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd	Web Services Security 1.0 [WS-Security]
wsse11	http://docs.oasis-open.org/wss/oasis-wsswssecurity-secext-1.1.xsd	Web Services Security 1.1
wst	http://schemas.xmlsoap.org/ws/2005/02/trust	WS-Trust [WS-Trust]
wsp	http://schemas.xmlsoap.org/ws/2004/09/policy	WS-Policy [WS-Policy]
sp	http://schemas.xmlsoap.org/ws/2005/07/securitypolicy	WS-SecurityPolicy [WS-SecurityPolicy]

3. Relying Party

This section describes the general framework used by a relying party Web service for specifying and conveying its security token requirements as well as its own identity to a service requester. The relying party Web service may be an application service or a delegate STS supporting the application service as shown in Figure 1. For brevity, the relying party STS shown as “Authentication/Authorization STS” in the figure will be referred to as “RP/STS” in the remainder of this section.

At a high-level, a relying party service or a RP/STS specifies its security policy, including its token requirements and security binding, as described in [WS-SecurityPolicy]. A service requester will obtain the security policy from the relying party using the mechanisms specified in [WS-MetadataExchange] before sending any application request messages. The service requester must obtain the required security tokens from the appropriate issuing authorities to satisfy the relying party policy, and submit each token along with proof-of-possession by binding the tokens to application messages.

3.1. Identifying the Relying Party

One of the driving requirements of the Information Card model, and thus of the Windows CardSpace system, is to support cryptographically verifiable but human-friendly identification of the recipient of a user’s digital identities. When a relying party requires that a user’s digital identity be submitted in the form of a security token containing claims, the user needs to have a reliable means to identify the relying party to make a trust decision of whether to release her digital identity or not. This requires that the identity of the relying party be conveyed to the service requester in a form that can be authenticated by the requester, yet presented to the user in a human-friendly manner for making trust decisions. Furthermore, it is important that the conveyed identity of the relying party be that of the organization or enterprise represented by the relying party Web service. Users make a conscious choice of whether or not to trust the actual organization or enterprise behind the Web service with their digital identities, not a specific service end-point.

Given the motivation described above, we recommend using “extended validation” X.509 certificates (as opposed to regular SSL server certificates) to identify the organization. In these certificates, the organization’s name and location information (if present) in the subject identifier should be marked with the extended validation quality (*i.e.*, asserted with high assurance). Furthermore, we recommend using certificates with extended validation logotypes for the issuer organization and subject organization [RFC 3709] for the purpose of visually identifying the relying party. Much of the information contained in digital certificates is appropriate and effective for machine processing; however, this information is not suitable for a corresponding human recognition and trust process. The use of extended validation logotypes is aimed at simplifying the human interpretation of the certificate content and helping the user’s decision to trust the subject organization.

The next question is how is this organizational identity conveyed to the service requester and surfaced to the user? Endpoint references, defined in WS-Addressing, convey the information needed to reference a Web service endpoint. The Information Card model uses the `Identity` element defined in [Addressing-Ext] to add identity information to an endpoint reference. This identity extension for an endpoint reference should be used to convey the identity of the organization behind that endpoint.

Here is an example of an endpoint reference augmented with identity data in the form of an X.509 certificate:

```
<wsa:EndpointReference>
  <wsa:Address>http://wh1.fabrikam123.com/Purchasing</wsa:Address>
  <wsid:Identity>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>...</ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </wsid:Identity>
</wsa:EndpointReference>
```

Security tokens returned by Windows CardSpace to the service requester application for submission to the target service will typically be encrypted to the key in the organizational certificate conveyed in the endpoint reference. Since the user evaluates and approves the identity of the organization in the endpoint reference as the recipient of his digital identity, encrypting the tokens this way guarantees that only the entity approved by the user can examine the content of the security tokens.

Other forms of organizational identity and reputations for organizations are possible, and can easily be accomodated in the Information Card model in the future.

3.1.1. Characteristics of certificate identifying the organization

Although ordinary SSL server certificates can be employed for identifying the relying party organization, it is HIGHLY RECOMMENDED that extended validation certificates be used as described above. Regardless of which type of X.509 certificate is used, it should satisfy the following characteristics [RFC2459]:

- The “Subject” field of the certificate must contain a non-empty X.500 distinguished name (DN) as the subject name. Further, the “CN” and “O” attributes must be present in the subject name. Optionally, zero or more of the locational attributes “L”, “S” and “C” may also be present.
- The “Key Usage” field of the certificate must assert at least the “digitalSignature” and “keyEncipherment” usage bits.

- If the certificate is also to be used for SSL based server authentication, then the "Extended Key Usage" field of the certificate must also assert at least the "Server Authentication" usage OID (1.3.6.1.5.5.7.3.1).

Windows CardSpace uses the subject name in the certficate presented by a RP to construct the RP-specific *private personal identifier* (PPID) claim for the user in issued security tokens (see [CardSpace-Ref] for description of PPID). The RP-specific PPID value is computed as a function of the *required* organization name ("O") and any *optional* location ("L", "S" and "C") attributes present in the subject name.

For extended validation certificates, these attributes are used by themselves. The validation and issuance criteria for such certificates provides a higher level of assurance for the asserted values. Moreover, adherence to a stricter issuance policy ensures uniformity of the assurance level across certificate authorities. However, that is not the case with regular certificates. Hence, the above attributes are further qualified by the subject names of the certificate authorities in the issuance hierarchy, and then used to compute any RP-specific user identifiers.

It is therefore important to note that if a relying party employs regular certificates for conveying its organization's identity, then the PPID value of its users may change if any node in the certificate issuance hierarchy changes, or if the relying party switches to using extended validation certificates.

3.2. Expressing Token Requirements of Relying Party

This section describes the mechanisms available to a relying party for specifying its security token (i.e. user identity) requirements as a prerequisite to providing service. The policy assertions and parameters described here are those already defined in [WS-SecurityPolicy] or extended by [CardSpace-Ref] where needed.

A relying party specifies its security token requirements as part of its security policy using the primitives and assertions described in [WS-SecurityPolicy]. The primary construct in the security policy of the relying party used to specify the type and claims content of security tokens issued by an identity provider is the <sp:IssuedToken> policy assertion. The basic form of the issued token policy assertion as defined in [WS-SecurityPolicy] is as follows.

```
<sp:IssuedToken sp:Usage="xs:anyURI" sp:IncludeToken="xs:anyURI" ...>
  <sp:Issuer>
    ...
  </sp:Issuer>
  <sp:RequestSecurityTokenTemplate>
    ...
  </sp:RequestSecurityTokenTemplate>
  <wsp:Policy>
    ...
  </wsp:Policy>
  ...
</sp:IssuedToken>
```

The following subsections describe the use of special parameters and policy assertion elements added by [CardSpace-Ref] as extensions to the sp:IssuedToken policy assertion that convey additional instructions to the service requester.

3.2.1. Issuer of tokens

The sp:IssuedToken/sp:Issuer element in an issued token policy specifies the issuer for the required token. More specifically, it should contain the endpoint reference of an identity provider STS that can issue the required token.

A relying party can specify the issuer for a required token in the following ways:

- Indicate a *specific* issuer by specifying the issuer's endpoint as the value of the `sp:Issuer/wsa:Address` element.
- Indicate that the issuer is *unspecified* by omitting the `sp:Issuer` element, which typically means that the service requester should determine the appropriate issuer for the required token with help from the user if necessary.

The ability to leave the issuer unspecified is useful in circumstances where the relying party cannot publicize, for confidentiality reasons, which identity providers it is willing to accept. For example, an enterprise that federates identities with other business partners may have a need to keep confidential who its business partners are. The relying party, however, makes the final determination of whether a presented token is acceptable.

When requiring a specific issuer, a relying party can specify that it will accept self-issued security tokens from the user by using the special URI below (defined in [CardSpace-Ref]).

```
http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self
```

Following is an example of using this URI within an issued token policy to specify that self-issued tokens will be accepted.

Example:

```
<sp:IssuedToken ...>
  <sp:Issuer>
    <wsa:Address>
      http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self
    </wsa:Address>
  </sp:Issuer>
  ...
</sp:IssuedToken>
```

When requiring a specific token issuer in policy, a relying party must specify the location of issuer metadata using the mechanism defined in [WS-Addressing] for embedding metadata within an endpoint reference. The following example shows a token policy with a specific issuer and its corresponding metadata location.

Example:

```
<sp:IssuedToken ...>
  <sp:Issuer>
    <wsa:Address>http://contoso.com/sts</wsa:Address>
    <wsa:Metadata>
      <wsx:Metadata>
        <wsx:MetadataSection
            Dialect="http://schemas.xmlsoap.org/ws/2004/09/mex">
          <wsx:MetadataReference>
            <wsa:Address>https://contoso.com/sts/mex</wsa:Address>
          </wsx:MetadataReference>
        </wsx:MetadataSection>
      </wsx:Metadata>
    </wsa:Metadata>
  </sp:Issuer>
  ...
</sp:IssuedToken>
```

In many circumstances, it is useful for a relying party to specify the issuer of a token as a logical name instead of an actual network address where the token is issued. A client can then resolve the logical name to an appropriate token issuing endpoint by means at its

disposal. A relying party can specify a *logical name* of the issuer, instead of its endpoint, as the value of the sp:Issuer/wsa:Address element in policy. Windows CardSpace selects information cards matching the logical issuer name to present to the user.

3.2.2. Type of token

A relying party can specify the type of required token by using the wst:TokenType element within its issued token policy assertion. The URI for a token type may be defined in token-specific profiles. The following example illustrates the use of this element in the relying party's security policy to request a SAML 1.1 token.

Example:

```
<sp:IssuedToken>
  <sp:RequestSecurityTokenTemplate>
    <wst:TokenType>
      urn:oasis:names:tc:SAML:1.0:assertion
    </wst:TokenType>
  </sp:RequestSecurityTokenTemplate>
</sp:IssuedToken>
```

Windows CardSpace is token type agnostic, and acts as a conduit for any token type requested by a relying party and issued by an identity provider.

3.2.3. Type of proof key in issued tokens

A relying party can explicitly request the use of an *asymmetric* or *symmetric* key in the required token by using the wst:KeyType element within its issued token policy assertion. The key type URIs are defined in [WS-Trust]. The following example illustrates the use of this element in security policy to request an asymmetric key in the issued token.

Example:

```
<sp:IssuedToken>
  <sp:RequestSecurityTokenTemplate>
    <wst:KeyType>
      http://schemas.xmlsoap.org/ws/2005/02/trust/PublicKey
    </wst:KeyType>
  </sp:RequestSecurityTokenTemplate>
</sp:IssuedToken>
```

The default bahavior of Windows CardSpace is to request an asymmetric key token from the identity provider if no explicit key type is specified by the relying party.

The choice of using symmetric or asymmetric key tokens can depend on a variety of technical and business factors. For example, symmetric keys provide token processing speed and efficiency, whereas asymmetric keys may be needed to meet legal business requirements of non-repudiation of submitted tokens.

It should be noted that the default behavior of Windows CardSpace is different for the special case of Browser based client interactions with a Web site, in which case it requests "bearer" tokens (see Section 4.3.5). Since a Browser can only submit a token to a Web site passively over HTTP without any proof-of-possession, bearer tokens with no proof keys are appropriate.

3.2.4. Claims in issued tokens

A relying party can ask for one or more claims in the token issued by an identity provider. If any required claims are missing in the token submitted, it can accept or reject that token at its own discretion.

The claims requirement of a relying party can be expressed in its token policy by using the optional `wst:Claims` parameter defined in [WS-Trust]. The `ic:ClaimType` element defined in [CardSpace-Ref] can be used, as a child of the `wst:Claims` element, to specify an individual claim type required. Further, each required claim can be specified as being *mandatory* or *optional*. Multiple `ic:ClaimType` elements can be included to specify multiple claim types required.

When using the Information Card model, the `wst:Dialect` attribute on the `wst:Claims` element should have the URI value shown below (defined in [CardSpace-Ref]). This value indicates that the claim type elements are to be processed as per the semantics of the Information Card model.

```
http://schemas.xmlsoap.org/ws/2005/05/identity
```

Following is an example of using this element within an issued token policy to require two claim types, where one claim type is optional.

Example:

```
<sp:IssuedToken ...>
  ...
  <sp:RequestSecurityTokenTemplate>
    ...
    <wst:Claims
        wst:Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity">
      <ic:ClaimType
        Uri="http://.../ws/2005/05/identity/claims/givenname"/>
      <ic:ClaimType
        Uri="http://.../ws/2005/05/identity/claims/surname"
        Optional="true" />
    </wst:Claims>
  </sp:RequestSecurityTokenTemplate>
  ...
</sp:IssuedToken>
```

The Windows CardSpace Technical Reference [CardSpace-Ref] defines a small set of claim types for common personal information about users that is supported for self-issued tokens. These claim types may be used by relying parties to specify their claim requirements. They may also be used by other 3rd party identity providers for the security tokens they issue. Other claim types may be defined and used by relying parties and identity providers for other specialized needs. Neither the Information Card model nor the Windows CardSpace system places any constraint on the claim types that can be used in tokens.

3.3. Expressing Privacy Policy of Relying Party

A relying party Web service should publish its "privacy policy" for its clients to retrieve and possibly display in user interfaces. Users may decide to release tokens and interact further with that service based on its privacy policy. No assumptions are made regarding the format and content of the privacy policy, and the Windows CardSpace system does not attempt to programmatically parse, interpret or act on the privacy policy.

A Web service can express the location of its privacy statement using the optional policy assertion element `ic:PrivacyNotice` defined in [CardSpace-Ref]. The XML attribute `Version` allows expressing changes in the version of the privacy statement when its content changes. Following is an example of using this policy element to express the location of the privacy statement of a Web service.

Example:

```
<wsp:Policy>
  ...
  <ic:PrivacyNotice Version=”1”>
    http://www.contoso.com/privacynotice
  </ic:PrivacyNotice>
  <sp:SymmetricBinding>
    ...
  </sp:SymmetricBinding>
  ...
</wsp:Policy>
```

Windows CardSpace can only accept the privacy statement location as an URL as illustrated above.

When a client system can only render the privacy statement document in a limited number of document formats (media types), it may use the HTTP request-header field “Accept” in its HTTP GET request to specify the media-types it can accept. For example, the following request-header specifies that the client will accept the privacy policy only as a plain text or a HTML document.

```
Accept: text/plain, text/html
```

Similarly, if a client system wants to obtain the privacy statement in a specific language, it may use the HTTP request-header field “Accept-Language” in its HTTP GET request to specify the languages it is willing to accept. For example, the following request-header specifies that the client will accept the privacy policy only in Danish.

```
Accept-Language: da
```

3.4. Example of Relying Party Security Policy

This section shows a complete example of policy for a relying party service containing policy assertions defined in [WS-SecurityPolicy] and in [CardSpace-Ref]. The first policy example is for a service endpoint and applies to all message interactions with that endpoint. It specifies the SOAP message security and token requirements of the service. The second policy example is for individual messages and can be attached to specific messages sent to the service endpoint. It specifies the message integrity and confidentiality requirements.

Example:

Policy attached to the service endpoint:

```
<wsp:Policy
 wsu:Id="ServiceEndpoint_policy"
 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
 xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
 xmlns:wsa="http://www.w3.org/2005/08/addressing"
 xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
 xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity">

 <wsp:ExactlyOne>
  <wsp:All>
   <ic:PrivacyNotice Version="2">
      http://www.contoso.com/privacypolicy
   </ic:PrivacyNotice>
   <sp:SymmetricBinding>
    <wsp:Policy>
     <sp:ProtectionToken>
```

```
      <wsp:Policy>
       <sp:X509Token
         sp:IncludeToken=".../ws/2005/07/securitypolicy/IncludeToken/Never">
        <wsp:Policy>
         <sp:RequireThumbprintReference />
         <sp:WssX509V3Token10 />
        </wsp:Policy>
       </sp:X509Token>
      </wsp:Policy>
     </sp:ProtectionToken>
     <sp:AlgorithmSuite>
      <wsp:Policy>
       <sp:Basic256 />
      </wsp:Policy>
     </sp:AlgorithmSuite>
     <sp:Layout>
      <wsp:Policy>
       <sp:Strict />
      </wsp:Policy>
     </sp:Layout>
     <sp:IncludeTimestamp />
     <sp:OnlySignEntireHeadersAndBody />
    </wsp:Policy>
   </sp:SymmetricBinding>
   <sp:EndorsingSupportingTokens>
    <wsp:Policy>
     <sp:IssuedToken sp:IncludeToken=”.../IncludeToken/AlwaysToRecipient”>
      <sp:Issuer>
       <wsa:Address>
         http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self
       </wsa:Address>
      </sp:Issuer>
      <sp:RequestSecurityTokenTemplate>
       <wst:TokenType>
         urn:oasis:names:tc:SAML:1.0:assertion
       </wst:TokenType>
       <wst:KeyType>
         http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey
       </wst:KeyType>
       <wst:Claims
         wst:Dialect=”http://schemas.xmlsoap.org/ws/2005/05/identity”>
        <ic:ClaimType
          Uri=”http://... /identity/claims/privatepersonalidentifier” />
       </wst:Claims
      </sp:RequestSecurityTokenTemplate>
     </sp:IssuedToken>
    </wsp:Policy>
   </sp:EndorsingSupportingTokens>
  </wsp:All>
 </wsp:ExactlyOne>
</wsp:Policy>
```

Policy attached to individual messages sent to the service endpoint:

```
<wsp:Policy
 wsu:Id="Service_message_policy"
 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
```

```
 xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">

 <wsp:ExactlyOne>
  <wsp:All>
    <sp:SignedParts>
     <sp:Body />
     <sp:Header Name="To" Namespace="http://.../2005/08/addressing" />
     <sp:Header Name="From" Namespace="http://.../2005/08/addressing" />
     <sp:Header Name="FaultTo" Namespace="http://.../2005/08/addressing" />
     <sp:Header Name="ReplyTo" Namespace="http://.../2005/08/addressing" />
     <sp:Header Name="MessageID" Namespace="http://.../2005/08/addressing" />
     <sp:Header Name="RelatesTo" Namespace="http://.../2005/08/addressing" />
     <sp:Header Name="Action" Namespace="http://.../2005/08/addressing" />
    </sp:SignedParts>
    <sp:EncryptedParts>
     <sp:Body />
    </sp:EncryptedParts>
  </wsp:All>
 </wsp:ExactlyOne>
</wsp:Policy>
```

3.5. Retrieving Relying Party Policy

The security policy of a relying party specifies its essage security and token requirements. A service requester can obtain the policy of the relying party using the mechanism specified in [WS-MetadataExchange].

It is highly recommended that the retrieval of the relying party policy should be a secured exchange using a secure transport mechanism like TLS/SSL to prevent tampering or security downgrade attacks. Windows CardSpace requires that the policy metadata of a relying party must be available at an endpoint using the HTTPS transport.

The following example illustrates the request and response messages for retrieving policy metadata.

Metadata request from service requester to relying party:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action S:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
    </wsa:Action>
    <wsa:MessageID>
      urn:uuid:ab9e1c77-0cea-4f2f-a586-78c15536137d
    </wsa:MessageID>
    <wsa:To S:mustUnderstand="1">
      https://www.contoso.com/sts/mex
    </wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>
        http://www.w3.org/2005/08/addressing/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
  </S:Header>
  <S:Body />
</S:Envelope>
```

Metadata response from relying party to service requester:

```
<S:Envelope ...>
 <S:Header>
  <wsa:Action S:mustUnderstand="1">
    http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
  </wsa:Action>
  <wsa:RelatesTo>
    uuid:ab9e1c77-0cea-4f2f-a586-78c15536137d
  </wsa:RelatesTo>
 </S:Header>
 <S:Body>
  <wsx:Metadata>
   <wsx:MetadataSection
       Dialect="http://schemas.xmlsoap.org/wsdl/" Identifier="...">
      ...
   </wsx:MetadataSection>
   <wsx:MetadataSection
       Dialect=" http://schemas.xmlsoap.org/wsdl/" Identifier="...">
     ...
   </wsx:MetadataSection>
   ...
  </wsx:Metadata>
 </S:Body>
</S:Envelope>
```

Note that since all metadata was requested, several metadata sections may be returned in the response each containing a specific type of metadata. For example, service contract definitions, policy declarations, and bindings including attached policy may each be returned in a separate metadata section.

3.6. Submitting Tokens to Relying Party

A service requester can submit the security token it obtains from Windows CardSpace to a relying party using any application specific mechanism. However, when using the web services SOAP security mechanism defined in [WS-Security], it should bind the token to application messages using the mechanisms described in [WS-SecurityPolicy]. Those mechanisms specify the security header layout for ordering of SOAP message elements, and how signatures must be used to provide proof-of-possession of a token using the proof key carried inside the security token.

4. Identity Provider

This section describes the general framework for identity providers to issue information cards, and for an identity selector like Windows CardSpace to request security tokens.

At a high-level, an information card carries the identity provider's issuance capabilities including the types of tokens and claim types it can issue, the location of its token services, and the authentication credential needed for requesting security tokens. It therefore contains enough information to allow an identity selector to match it with a relying party's requirements. Once a match occurs and the user selects an information card, the identity selector requests and obtains the appropriate security token from the identity provider using the mechanisms described in [WS-Trust].

An identity provider runs one or more instances of security token services as shown in Figure 1 to handle security token requests. For brevity, the identity provider STS will be referred to as "IP/STS" in the remainder of this section.

4.1. Information Card

An information card symbolically represents the digital identity of a user issued by an identity provider. As a concrete artifact, it is a container of identity metadata. Furthermore, being a concrete entity, it is portable and can be carried by a user to be used from any computer or device through which Web services are accessed.

4.1.1. Information card format

An information card is concretely represented as a XML document that can be issued by an identity provider and stored by an user on any storage device of their choice. It does not inherently contain any confidential data.

The XML schema for an information card is described in [CardSpace-Ref] and is represented by the `ic:InformationCard` element. The `xml:lang` attribute can be used, either at the root element or at any of the child elements, to specify the language in which the content of those elements in the information card has been localized.

NOTE: Windows CardSpace has the additional restriction that there must not be any "whitespace" characters between XML element tags, or between an element tag and the element content (unless the whitespace is explicitly part of the element content) in the information card.

The following example illustrates an information card issued by "Contoso, Inc." that supports the SAML token type, two claim types, requires that the relying party identity be conveyed in token requests, and requires authentication based on username/password when requesting tokens. Note that whitespace (newline and space character) is included in the example shown only to improve readability; they must not be present in an actual implementation.

Example:

```
<InformationCard
    xmlns="http://schemas.xmlsoap.org/ws/2005/05/identity"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
    xml:lang="en-us">
  <InformationCardReference>
    <CardId>http://contoso.com/CardId/d795621fa01d454285f9</CardId>
    <CardVersion>1</CardVersion>
  </InformationCardReference>
  <CardName>Contoso Membership Card</CardName>
  <CardImage MimeType="image/gif"> ... </CardImage>
  <Issuer>http://contoso.com</Issuer>
  <TimeIssued>2003-08-24T00:30:05Z</TimeIssued>
  <TokenServiceList>
    <TokenService>
      <wsa:EndpointReference>
        <wsa:Address>http://contoso.com/sts</wsa:Address>
        <wsa:Metadata>
          <wsx:Metadata>
            <wsx:MetadataSection
                Dialect="http://schemas.xmlsoap.org/ws/2004/09/mex">
              <wsx:MetadataReference>
                <wsa:Address>https://contoso.com/sts/mex</wsa:Address>
              </wsx:MetadataReference>
            </wsx:MetadataSection>
          </wsx:Metadata>
```

```
        </wsa:Metadata>
      </wsa:EndpointReference>
      <UserCredential>
        <UsernamePasswordCredential>
          <Username>Zoe</Username>
        </UsernamePasswordCredential>
      </UserCredential>
    </TokenService>
  </TokenServiceList>
  <SupportedTokenTypeList>
    <wst:TokenType>urn:oasis:names:tc:SAML:1.0:assertion</wst:TokenType>
  </SupportedTokenTypeList>
  <SupportedClaimTypeList>
    <SupportedClaimType Uri="…/ws/2005/05/identity/claims/givenname">
      <DisplayTag>Given Name</DisplayTag>
    </SupportedClaimType>
    <SupportedClaimType Uri="…/ws/2005/05/identity/claims/surname">
      <DisplayTag>Last Name</DisplayTag>
    </SupportedClaimType>
  </SupportedClaimTypeList>
  <RequireAppliesTo />
  <PrivacyNotice Version="1">
    http://contoso.com/privacynotice
  </PrivacyNotice>
</InformationCard>
```

The subset of schema elements in an information card used to express the token issuance capabilities and requirements of the identity provider are briefly discussed below.

4.1.1.1. Expressing logical name of token issuer

An identity provider can express an URI as a logical name for itself acting as the token issuer using the `ic:Issuer` element in an information card. When a relying party specifies a logical name as the issuer of a required token (in the `sp:Issuer/wsa:Address` field of its issued token policy), Windows CardSpace selects information cards with a matching `ic:Issuer` element value. The following example illustrates the use of this element.

Example:

```
<ic:Issuer>http://contoso.com</ic:Issuer>
```

4.1.1.2. Expressing token service endpoints and authentication mechanisms

An identity provider can publish a prioritized list of endpoints for its IP/STS and a descriptor of the corresponding user credential required for each endpoint using the element `ic:TokenServiceList` in an information card. For each endpoint, the required credential type implicitly determines the authentication mechanism to be used. Each credential descriptor is personalized for the user to allow Windows CardSpace to automatically locate the credential once the user has selected an information card.

Further, each IP/STS endpoint reference in the information card must also include a metadata endpoint that responds to WS-Transfer/Get based metadata requests for the WSDL and policy for the IP/STS endpoint [WS-MetadataExchange]. Windows CardSpace retrieves the WSDL from that metadata endpoint to find the policy for communicating securely with the IP/STS. The IP/STS metadata endpoint must support the secure HTTPS transport mechanism to prevent policy tampering attacks.

The following example illustrates an identity provider with two endpoints for its IP/STS, one requiring Kerberos (higher priority) and the other requiring username/password (lower priority) as its authentication mechanism.

Example:

```
<ic:TokenServiceList>
  <ic:TokenService>
    <wsa:EndpointReference>
      <wsa:Address>http://contoso.com/sts/kerb</wsa:Address>
      <wsa:Metadata>
        <wsx:Metadata>
          <wsx:MetadataSection
              Dialect="http://schemas.xmlsoap.org/ws/2004/09/mex">
            <wsx:MetadataReference>
              <wsa:Address>https://contoso.com/sts/kerb/mex</wsa:Address>
            </wsx:MetadataReference>
          </wsx:MetadataSection>
        </wsx:Metadata>
      </wsa:Metadata>
    </wsa:EndpointReference>
    <ic:UserCredential>
      <ic:KerberosV5Credential />
    </ic:UserCredential>
  </ic:TokenService>
  <ic:TokenService>
    <wsa:EndpointReference>
      <wsa:Address>http://contoso.com/sts/pwd</wsa:Address>
      <wsa:Metadata>
        <wsx:Metadata>
          <wsx:MetadataSection
              Dialect="http://schemas.xmlsoap.org/ws/2004/09/mex">
            <wsx:MetadataReference>
              <wsa:Address>https://contoso.com/sts/pwd/mex</wsa:Address>
            </wsx:MetadataReference>
          </wsx:MetadataSection>
        </wsx:Metadata>
      </wsa:Metadata>
    </wsa:EndpointReference>
    <ic:UserCredential>
      <ic:UsernamePasswordCredential>
        <ic:Username>Zoe</ic:Username>
      </ic:UsernamePasswordCredential>
    </ic:UserCredential>
  </ic:TokenService>
</ic:TokenServiceList>
```

4.1.1.3. Expressing token types offered

An identity provider can express the list of security token types it issues by using the `ic:SupportedTokenTypeList` element in an information card. The following example illustrates that an identity provider can issue both SAML 1.1 and SAML 2.0 tokens.

Example:

```
<ic:SupportedTokenTypeList>
  <wst:TokenType>urn:oasis:names:tc:SAML:1.0:assertion</wst:TokenType>
  <wst:TokenType>urn:oasis:names:tc:SAML:2.0:assertion</wst:TokenType>
</ic:SupportedTokenTypeList>
```

4.1.1.4. Expressing claim types offered

An identity provider can express the list of claim types it can assert by using the `ic:SupportedClaimTypeList` element in an information card. The following example illustrates that an identity provider can assert two claim types.

Example:

```
<ic:SupportedClaimTypeList>
  <ic:SupportedClaimType Uri="…/ws/2005/05/identity/claims/givenname">
    <ic:DisplayTag xml:lang="en-us">Given Name</DisplayTag>
  </ic:SupportedClaimType>
  <ic:SupportedClaimType Uri="…/ws/2005/05/identity/claims/surname">
    <ic:DisplayTag xml:lang="en-us">Last Name</DisplayTag>
  </ic:SupportedClaimType>
</ic:SupportedClaimTypeList>
```

4.1.1.5. Requiring token scope information

Windows CardSpace, by default, does not convey information about the relying party where an issued token will be used (i.e., target scope) when requesting security tokens. This helps safeguard user privacy. However, an identity provider can override that behavior if there are justifiable reasons to do so (e.g. audit requirements for compliance). The element `ic:RequireAppliesTo` can be used for this purpose.

Example:

```
<ic:RequireAppliesTo />
```

The presence of this element in an information card dictates that an identity selector, like Windows CardSpace, must convey the relying party information in a `wsp:AppliesTo` element in its token request message.

4.1.1.6. Expressing privacy policy location

An identity provider can express the location of its privacy statement using the element `ic:PrivacyNotice` in an information card. The XML attribute `Version` allows expressing changes in the version of the privacy statement when its content changes. Following is an example of using this element to express the privacy statement location.

Example:

```
<ic:PrivacyNotice Version="1">
  http://www.contoso.com/privacynotice
</ic:PrivacyNotice>
```

Windows CardSpace can only accept URL-based privacy statement location as shown above.

4.1.2. Issuing information cards

An identity provider can issue information cards to its users using any out-of-band mechanism that is mutually suitable. For example, a user may log on to a Web site provided by the identity provider and download the information card over the HTTP connection. Alternatively, an identity provider may send the information card through email to the user's email address on file. Remember that the information card is not the security token; it only contains metadata about the relationship between the user and the identity provider.

In order to provide the assurance that an information card is indeed issued by the identity provider expected by the user, the information card should be carried inside a digitally signed envelope that is signed by the identity provider. For this, the "enveloping signature" construct (see [XMLDSIG]) should be used where the information card is included in the

ds:Object element. This is illustrated in the example below. The specific details of the XML digital signature profile that should be used to sign the envelope is described in [CardSpace-Ref]. The signature on the digitally signed envelope provides data origin authentication assuring the user that it came from the right identity provider.

It is highly recommend that an extended validation X.509 certificate for the identity provider, preferably with extended validation logotypes, be used to sign the envelope. Windows CardSpace uses this certificate to show the identity provider in its user interface to enable the user to visually identify it.

The following example shows an information card within an enveloping signature container using that prescribed format.

Example:

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    <SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference URI="#_Object_InformationCard">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <DigestValue> ... </DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue> ... </SignatureValue>
  <KeyInfo>
    <X509Data>
      <X509Certificate> ... </X509Certificate>
    </X509Data>
  </KeyInfo>
  <Object Id=”_Object_InformationCard”>
    <ic:InformationCard
        xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
        xml:lang=”en-us”>
      [Actual information card content]
    </ic:InformationCard>
  </Object>
</Signature>
```

Windows CardSpace verifies the enveloping signature and visually identifies the identity provider to the user in its user interface. Upon user approval, it extracts the ic:InformationCard element and stores it in the user’s information card collection.

Windows CardSpace recognizes the special file extension .CRD for information cards. A file with that extension is recognized and interpreted as a signed XML document representing an issued information card. A file named with the .CRD file extension and containing the document shown in the example above would be treated as an information card.

4.2. Identity Provider Policy

An identity provider uses the policy assertions defined in [WS-SecurityPolicy] to specify the authentication and communication security requirements of its IP/STS. Policy assertions are

attached to endpoints or messages in the WSDL for the IP/STS. This section describes any additional policy elements or requirements introduced by the Information Card model.

4.2.1. Require information card provisioning

In the Information Card model, an identity provider requires provisioning in the form of an an information card issued by it which represents the provisioned identity of the user. In order to enable a token requester to learn that such pre-provisioning is necessary before token requests can be made, the identity provider must provide an indication in its policy.

An identity provider issuing information cards must specify its provisioning requirement in its policy using the optional policy element `ic:RequireFederatedIdentityProvisioning` defined in [CardSpace-Ref]. Following is an example of using this policy element.

Example:

```
<wsp:Policy>
  ...
  <ic:RequireFederatedIdentityProvisioning />
  <sp:SymmetricBinding>
    ...
  </sp:SymmetricBinding>
  ...
</wsp:Policy>
```

Further, to allow the identity provider to verify that its provisioning requirement has been satisfied, a token requester must include a reference to the provisioned entity in its token requests to the IP/STS. Windows CardSpace always includes a reference to the specific information card used in its token request.

4.2.2. Secure policy metadata

The IP/STS must provide a metadata endpoint that responds to WS-Transfer/Get based metadata requests for its WSDL and policy (see Section 4.1.1.2). Windows CardSpace retrieves the WSDL from the metadata endpoint for the IP/STS to find the policy for communicating securely with it. The metadata endpoint must support the secure HTTPS transport mechanism to prevent policy tampering attacks.

Section 0 illustrates the request and response messages of a retrieval of WSDL with policy metadata.

4.3. Token Request and Response

When the user selects an information card on a service requester machine to send to a relying party, Windows CardSpace on that system obtains a security token from the IP/STS for that information card. Security tokens are requested using the issuance binding mechanism described in [WS-Trust]. Specifically, tokens are requested by submitting a RequestSecuriyToken (RST) message to the IP/STS.

This section describes the specific extensions to the token request message introduced by the Information Card model (defined in [CardSpace-Ref]), and the specific behavior of Windows CardSpace when requesting tokens. Note that the extension elements introduced by the Information Card model are all optional, and they can be ignored by an IP/STS if present in a token request.

4.3.1. Information card reference

Each information card has a unique identifier and version by which it can be referenced, given by the `ic:InformationCardReference` element in an information card. When

requesting tokens from the IP/STS, Windows CardSpace includes the information card reference in the RST message as a top-level element information item.

Following is an example of the information card reference included in a RST message.

Example:

```
<wst:RequestSecurityToken>
  ...
  <ic:InformationCardReference>
    <ic:CardId>http://xyz.com/CardId/d795621fa01d454285f9</ic:CardId>
    <ic:CardVersion>1</ic:CardVersion>
  </ic:InformationCardReference>
  ...
</wst:RequestSecurityToken>
```

The card reference is only meaningful to the IP/STS. It may use that information to ensure that a valid provisioning action had occurred earlier, or to determine if the corresponding information card is stale or out-of-date for whatever reason. The IP/STS may fault with `ic:InformationCardRefreshRequired` (defined in [CardSpace-Ref]) to signal to the service requester that the information card needs to be refreshed.

4.3.2. Claims and other token parameters

A relying party may require a specific set of claims and other token parameters that must be communicated to the IP/STS. These are expressed in the policy of the relying party using the `sp:RequestSecurityTokenTemplate` parameter within the `sp:IssuedToken` policy assertion (see Section 3.1.1). The content of this element (i.e. all of its [children] elements) are directly copied by Windows CardSapce into the RST message sent to the IP/STS.

For example, if the relying party asks for an issued token in its policy as follows:

```
<sp:IssuedToken>
  <sp:RequestSecurityTokenTemplate>
    <wst:KeyType>
      http://schemas.xmlsoap.org/ws/2005/02/trust/PublicKey
    </wst:KeyType>
    <wst:Claims
        wst:Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity">
      <ic:ClaimType
        Uri="http://.../ws/2005/05/identity/claims/givenname"/>
      <ic:ClaimType
        Uri="http://.../ws/2005/05/identity/claims/surname"
        Optional="true" />
    </wst:Claims>
  </sp:RequestSecurityTokenTemplate>
</sp:IssuedToken>
```

Windows CardSapce on the service requester copies the entire content of the element `sp:RequestSecurityTokenTemplate` into the RST message as follows.

Example:

```
<wst:RequestSecurityToken>
  <wst:KeyType>
    http://schemas.xmlsoap.org/ws/2005/02/trust/PublicKey
  </wst:KeyType>
  <wst:Claims
      wst:Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity">
    <ic:ClaimType
      Uri="http://.../ws/2005/05/identity/claims/givenname"/>
```

```
    <ic:ClaimType
      Uri="http://.../ws/2005/05/identity/claims/surname"
      Optional="true" />
  </wst:Claims>
  ...
</wst:RequestSecurityToken>
```

4.3.3. Token scope

The [WS-Trust] protocol allows a token requester to indicate the target where the issued token will be used (*i.e.*, token scope) by using the optional element `wsp:AppliesTo` in the RST message. When included in a token request message, this element typically contains the endpoint reference of the relying party.

To protect user privacy Windows CardSpace does not, by default, reveal information about the relying party to the identity provider in token requests. In other words, the element `wsp:AppliesTo` is absent in token request RST messages. However, if the identity provider includes the `ic:RequireAppliesTo` element in the information card, then the token scope information may be included in the token request. The actual behavior of Windows CardSpace with respect to when and how the `wsp:AppliesTo` element is included in the token request is described in [CardSpace-Ref].

The following example illustrates the token scope information included in a RST message.

Example:

```
<wst:RequestSecurityToken>
  <wsp:AppliesTo>
    <wsa:EndpointReference>
      <wsa:Address>http://ip.fabrikam.com</wsa:Address>
      <wsid:Identity>
        <ds:KeyInfo>
          <ds:X509Data>
            <ds:X509Certificate>...</ds:X509Certificate>
          </ds:X509Data>
        </ds:KeyInfo>
      </wsid:Identity>
    </wsa:EndpointReference>
  </wsp:AppliesTo>
  ...
</wst:RequestSecurityToken>
```

4.3.4. Client pseudonym

The claim type “private personal identifier” (or PPID) defined in [CardSpace-Ref] and identified by the following URI represents a pseudonym for a user at a given relying party.

http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier

It has the privacy property that the PPID for a user at two different relying parties is guaranteed to be different such that they cannot be used as the basis for collusion.

An identity provider issuing this claim must do so using data present in the RST request. If the target scope information is present in the token request, then it can be used for constructing a RP-specific PPID claim value. However, Windows CardSpace does not always include target scope information in its request. To enable an identity provider that supports the PPID claim type to be able to always produce a consistent RP-specific claim value, the extension element `ic:ClientPseudonym/ic:PPID` is included in the RST request when token scope information is absent. It contains the result of applying a hash function to a

relying party identity and optional user-supplied entropy to produce an opaque yet consistent reference for the relying party. The IP/STS may use this value as is or as an input seed to a custom function to derive a value for the PPID claim.

An opaque reference for the relying party included in a RST message is shown in the following example.

Example:

```
<wst:RequestSecurityToken>
  <ic:ClientPseudonym>
    <ic:PPID>MIIEZzCCA9CgAwIBAgIQEmtJZc0=</ic:PPID>
  </ic:ClientPseudonym >
  ...
</wst:RequestSecurityToken>
```

4.3.5. Proof key

A security token asserts claims which can be coupled with digital signatures to provide mechanisms for demonstrating evidence of the sender's knowledge of the keys described by the security token. The key described by a security token is called the “proof key”, and the data used to demonstrate the sender's knowledge of that key is called “proof-of-possession” of the security token.

The optional `wst:KeyType` element in the RST request indicates the type of proof key desired in the issued security token. An issued token may have a *symmetric* proof key (symmetric key token), an *asymmetric* proof key (asymmetric key token), or *no* proof key (bearer token). A relying party can specify the desired key type in its policy within the `sp:RequestSecurityTokenTemplate` parameter of its required token assertion. If no key type is specified in the relying party policy, then Windows CardSpace requests an asymmetric key token from the IP/STS by default.

The IP/STS can return the proof key in a `wst:RequestedProofToken` element in the RSTR response along with the issued token. Note that the token response is always carried over a confidential channel wherein either an encrypted transport (transport security) or SOAP message confidentiality (messsage security) is used.

The actual behavior of Windows CardSpace with respect to how each proof key type is requested, who contributes entropy, how the proof key is computed and returned is described in [CardSpace-Ref].

4.3.6. Display token

Windows CardSpace is agnostic of specific token types that may be requested by a relying party and issued by an identity provider. The token returned by an IP/STS may be completely opaque to Windows CardSpace which simply provides a conduit. However, to allow informed user consent and release, the Information Card model introduces the notion of a *display token*. It is an informational token associated with the actual security token that essentially contains a friendly representation of the claims carried in the security token. Its friendly content can be displayed to the user in user interfaces.

The optional `ic:RequestDisplayToken` element defined in [CardSpace-Ref] can be used in the RST message to request a display token corresponding to the issued token from the IP/STS. It is optional for an IP/STS to process display token requests. However, it is highly recommended that when requested display tokens be returned along with issued tokens to enable informed participation by the user. Windows CardSpace always requests a display token with every token request.

The following example shows a token request including a request for display token localized in “US English”.

Example:

```
<wst:RequestSecurityToken>
  ...
  <ic:RequestDisplayToken xml:lang=”en-us” />
</wst:RequestSecurityToken>
```

To return a display token, the IP/STS can use the optional `ic:RequestedDisplayToken` element defined in [CardSpace-Ref] in the RSTR response message. The `xml:lang` attribute is used to specify the language in which the returned display token is localized.

The following example illustrates a token response that includes a display token localized in “US English” for a security token carrying two claims.

Example:

```
<wst:RequestSecurityTokenResponse>
  ...
  <ic:RequestedDisplayToken>
    <ic:DisplayToken xml:lang=”en-us”>
      <ic:DisplayClaim Uri=”http://.../ws/2005/05/identity/claims/givenname”>
        <ic:DisplayTag>Given Name</ic:DisplayTag>
        <ic:DisplayValue>John</ic:DisplayValue>
      </ic:DisplayClaim>
      <ic:DisplayClaim Uri=”http://.../ws/2005/05/identity/claims/surname”>
        <ic:DisplayTag>Last Name</ic:DisplayTag>
        <ic:DisplayValue>Doe</ic:DisplayValue>
      </ic:DisplayClaim>
    <ic:DisplayToken>
  </ic:RequestedDisplayToken>
</wst:RequestSecurityTokenResponse>
```

5. Message Exchanges with Identity Provider

The information card includes a descriptor for the user credential needed to authenticate the user to the IP/STS when requesting tokens. For each supported credential type, this section describes in detail:

- the format of the credential descriptor used,
- the security policy that the IP/STS should use, and
- the SOAP messages exchanged between the IP/STS and token requester.

Note that Windows CardSpace retrieves the WSDL containing the security policy of the IP/STS before requesting tokens to determine how messages are to be secured and the type of authentication to use. If the required authentication token type in the retrieved security policy does not match the corresponding credential type specified in the information card, then Windows CardSpace fails without sending the token request.

The security of the messages exchanged between Windows CardSpace and the IP/STS is governed by the security binding assertions specified in the IP/STS policy. For the binding assertions, [WS-SecurityPolicy] specifies the SOAP security header layout for ordering of elements and signatures in messages. The XML signature [XMLDSIG] profile used for signatures and the XML encryption [XMLENC] profile used for encryption of keys and other elements in the messages is governed by the value of the `sp:AlgorithmSuite` assertion in the security binding. These profiles are also described in [WS-SecurityPolicy].

NOTE: Windows CardSpace has the restriction that there must not be any “whitespace” characters between XML element tags, or between an element tag and the element content (unless the whitespace is explicitly part of the element content) in the metadata response message (*i.e.*, the WSDL) and the token response message (*i.e.*, the RSTR) in the SOAP body. Although whitespace (newline and space character) is included in the examples shown to improve readability, they must not be present in an actual implementation.

When the AES with CBC encryption method is used for message confidentiality, the padding method used is as per the PKCS-7 standard in which the number of octets remaining in the last block is used as the padding octet value

5.1. Retrieving Identity Provider Policy

When an information card is selected by the user, Windows CardSpace prepares to request a security token from the corresponding IP/STS by first fetching its WSDL containing its security policy. The WSDL is retrieved as metadata by using the WS-Transfer/Get based retrieval method defined in [WS-MetadataExchange] and illustrated in this section. The IP/STS endpoint specified in an information card issued by the IP must include an endpoint that responds to WS-Transfer/Get based metadata requests from a client.

5.1.1. WSDL and security policy

The security policy of the IP/STS indicates endpoint behavior over a token request/response sequence, and specifies policy for client credential requirements and how messages should be secured on the channel. In the WSDL, policy meant for an STS endpoint should be attached to the `wsdl:binding` element whereas policy meant for token request/response messages should be attached to the `wsdl:operation` element (or the `wsdl:input` and `wsdl:output` elements).

This section illustrates the complete metadata that can be used by an IP/STS to specify its WSDL and security policy. The metadata illustrations show the attachment of policy to the appropriate WSDL elements. The security policy for two separate cases are discussed below, one using transport security and the other using message security for securing the token request and response exchanges between the IP/STS and the client.

The examples use a target namespace of *http://constoso.com* that must be replaced with the actual namespace representing the IP/STS. Further, the required token assertion in the security policy for authenticating the client varies with the type of client credential required. The credential specific token assertion in the security policy is shown as a placeholder in the examples, and more specifically described in the credential specific sections that follow.

5.1.1.1. Using transport binding

This section illustrates the metadata of an IP/STS containing its WSDL and security policy when transport security (**transport binding**) is used to secure its SOAP message exchanges with a client. For this security binding, message protection and security correlation for the request and response legs of the message exchange is provided by the secure HTTPS transport. There is no message level encryption required. This is described in [WS-SecurityPolicy].

Example:

Metadata containing WSDL and policy when using transport security:

```
<Metadata xmlns="http://schemas.xmlsoap.org/ws/2004/09/mex">
  <MetadataSection
      Dialect="http://schemas.xmlsoap.org/wsdl/"
      Identifier="http://schemas.xmlsoap.org/ws/2005/02/trust">
```

```
   <wsdl:definitions name="STS_wsdl" targetNamespace="http://contoso.com"
      xmlns:tns="http://contoso.com"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:wsa="http://www.w3.org/2005/08/addressing"
      xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
      xmlns:wsid="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity"
      xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
      xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
      xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
      xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
      xmlns:q1="http://contoso.com/schemas">
      <wsdl:types>
        <xs:schema
targetNamespace="http://schemas.xmlsoap.org/ws/2005/02/trust/Imports">
          <xs:import schemaLocation="" namespace="http://contoso.com/schemas"
/>
        </xs:schema>
      </wsdl:types>
      <wsdl:message name="RequestSecurityTokenMsg">
        <wsdl:part name="request" type="q1:MessageBody" />
      </wsdl:message>
      <wsdl:message name="RequestSecurityTokenResponseMsg">
        <wsdl:part name="response" type="q1:MessageBody" />
      </wsdl:message>

      <wsdl:portType name="SecurityTokenService">
        <wsdl:operation name="Issue">
          <wsdl:input
wsaw:Action="http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue"
            message="tns:RequestSecurityTokenMsg">
          </wsdl:input>
          <wsdl:output
wsaw:Action="http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue"
            message="tns:RequestSecurityTokenResponseMsg">
          </wsdl:output>
        </wsdl:operation>
      </wsdl:portType>

      <wsp:Policy wsu:Id="STS_endpoint_policy">
        <wsp:ExactlyOne>
          <wsp:All>
            <ic:RequireFederatedIdentityProvisioning />
            <sp:TransportBinding>
              <wsp:Policy>
                <sp:TransportToken>
                  <wsp:Policy>
                    <sp:HttpsToken RequireClientCertificate="false"/>
                  </wsp:Policy>
                </sp:TransportToken>
                <sp:AlgorithmSuite>
                  <wsp:Policy>
                    <sp:Basic256/>
```

```
                </wsp:Policy>
              </sp:AlgorithmSuite>
              <sp:Layout>
                <wsp:Policy>
                  <sp:Strict/>
                </wsp:Policy>
              </sp:Layout>
              <sp:IncludeTimestamp/>
            </wsp:Policy>
          </sp:TransportBinding>
          [Authentication token assertion]
          <sp:Wss11>
            <wsp:Policy>
              <sp:MustSupportRefThumbprint/>
              <sp:MustSupportRefEncryptedKey/>
            </wsp:Policy>
          </sp:Wss11>
          <sp:Trust10>
            <wsp:Policy>
              <sp:RequireClientEntropy/>
              <sp:RequireServerEntropy/>
            </wsp:Policy>
          </sp:Trust10>
          <wsaw:UsingAddressing wsdl:required="true" />
        </wsp:All>
      </wsp:ExactlyOne>
    </wsp:Policy>

    <wsdl:binding name="Transport_binding" type="tns:SecurityTokenService">
      <wsp:PolicyReference URI="#STS_endpoint_policy"/>
      <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="Issue">
        <soap12:operation
soapAction="http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue"
          style="document"/>
        <wsdl:input>
          <soap12:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
          <soap12:body use="literal"/>
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="STS_0">
      <wsdl:port name="STS_0_port" binding="tns:Transport_binding">
        <soap12:address location="https://contoso.com/sts"/>
        <wsa:EndpointReference>
          <wsa:Address>https://contoso.com/sts</wsa:Address>
          <wsid:Identity>
            <ds:KeyInfo>
              <ds:X509Data>
                <ds:X509Certificate>
                  [base64 encoded certificate value]
                </ds:X509Certificate>
              </ds:X509Data>
            </ds:KeyInfo>
```

```
            </wsid:Identity>
          </wsa:EndpointReference>
        </wsdl:port>
      </wsdl:service>
    </wsdl:definitions>
  </MetadataSection>

  <MetadataSection
      Dialect="http://www.w3.org/2001/XMLSchema"
      Identifier="http://contoso.com/schemas">
    <xs:schema xmlns:tns="http://contoso.com/schemas"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified"
        targetNamespace="http://contoso.com/schemas">
      <xs:complexType name="MessageBody">
        <xs:sequence>
          <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any"/>
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
  </MetadataSection>
</Metadata>
```

Note that the token assertion required for client authentication is shown as a placeholder with the text *"[Authentication Token Assertion]"* inside the security policy and highlighted in the metadata example above. The authentication token assertion for each type of required client credential is described in later sections. Other metadata content that would need to be substituted when used with a real IP/STS is also highlighted.

5.1.1.2. Using symmetric binding

This section illustrates the metadata of an IP/STS containing its WSDL and security policy when symmetric message security (**symmetric binding**) is used to secure its SOAP message exchanges with a client. For this security binding, message protection and security correlation for the request and response legs of the message exchange is provided by an ephemeral symmetric session key. Message integrity and confidentiality is governed by the policy attached to individual messages in the WSDL. This is described in [WS-SecurityPolicy].

Example:

Metadata containing WSDL and policy when using message security with symmetric binding:

```
<Metadata xmlns="http://schemas.xmlsoap.org/ws/2004/09/mex">
  <MetadataSection
      Dialect="http://schemas.xmlsoap.org/wsdl/"
      Identifier="http://schemas.xmlsoap.org/ws/2005/02/trust">
   <wsdl:definitions name="STS_wsdl" targetNamespace="http://contoso.com"
      xmlns:tns="http://contoso.com"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:wsa="http://www.w3.org/2005/08/addressing"
      xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
      xmlns:wsid="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity"
      xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
      xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
      xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
```

```
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
      xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      xmlns:ic="http://schemas.xmlsoap.org/ws/2005/05/identity"
      xmlns:q1="http://contoso.com/schemas">
      <wsdl:types>
        <xs:schema
targetNamespace="http://schemas.xmlsoap.org/ws/2005/02/trust/Imports">
          <xs:import schemaLocation="" namespace="http://contoso.com/schemas"
/>
        </xs:schema>
      </wsdl:types>
      <wsdl:message name="RequestSecurityTokenMsg">
        <wsdl:part name="request" type="q1:MessageBody" />
      </wsdl:message>
      <wsdl:message name="RequestSecurityTokenResponseMsg">
        <wsdl:part name="response" type="q1:MessageBody" />
      </wsdl:message>

      <wsdl:portType name="SecurityTokenService">
        <wsdl:operation name="Issue">
          <wsdl:input
wsaw:Action="http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue"
            message="tns:RequestSecurityTokenMsg">
          </wsdl:input>
          <wsdl:output
wsaw:Action="http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue"
            message="tns:RequestSecurityTokenResponseMsg">
          </wsdl:output>
        </wsdl:operation>
      </wsdl:portType>

      <wsp:Policy wsu:Id="STS_endpoint_policy">
        <wsp:ExactlyOne>
          <wsp:All>
            <ic:RequireFederatedIdentityProvisioning />
            <sp:SymmetricBinding>
              <wsp:Policy>
                <sp:ProtectionToken>
                  <wsp:Policy>
                    <sp:X509Token sp:IncludeToken="http://schemas.xmlsoap.org
/ws/2005/07/securitypolicy/IncludeToken/Never">
                      <wsp:Policy>
                        <sp:RequireThumbprintReference/>
                        <sp:WssX509V3Token10/>
                      </wsp:Policy>
                    </sp:X509Token>
                  </wsp:Policy>
                </sp:ProtectionToken>
                <sp:AlgorithmSuite>
                  <wsp:Policy>
                    <sp:Basic256/>
                  </wsp:Policy>
                </sp:AlgorithmSuite>
                <sp:Layout>
                  <wsp:Policy>
```

```
                  <sp:Strict/>
                </wsp:Policy>
              </sp:Layout>
              <sp:IncludeTimestamp/>
              <sp:OnlySignEntireHeadersAndBody/>
            </wsp:Policy>
          </sp:SymmetricBinding>
          [Authentication token assertion]
          <sp:Wss11>
            <wsp:Policy>
              <sp:MustSupportRefThumbprint/>
              <sp:MustSupportRefEncryptedKey/>
            </wsp:Policy>
          </sp:Wss11>
          <sp:Trust10>
            <wsp:Policy>
              <sp:RequireClientEntropy/>
              <sp:RequireServerEntropy/>
            </wsp:Policy>
          </sp:Trust10>
          <wsaw:UsingAddressing wsdl:required="true" />
        </wsp:All>
      </wsp:ExactlyOne>
    </wsp:Policy>

    <wsp:Policy wsu:Id="STS_message_policy">
      <wsp:ExactlyOne>
        <wsp:All>
          <sp:SignedParts>
            <sp:Body />
            <sp:Header Name="To"
              Namespace="http://www.w3.org/2005/08/addressing"/>
            <sp:Header Name="From"
              Namespace="http://www.w3.org/2005/08/addressing"/>
            <sp:Header Name="FaultTo"
              Namespace="http://www.w3.org/2005/08/addressing"/>
            <sp:Header Name="ReplyTo"
              Namespace="http://www.w3.org/2005/08/addressing"/>
            <sp:Header Name="MessageID"
              Namespace="http://www.w3.org/2005/08/addressing"/>
            <sp:Header Name="RelatesTo"
              Namespace="http://www.w3.org/2005/08/addressing"/>
            <sp:Header Name="Action"
              Namespace="http://www.w3.org/2005/08/addressing"/>
          </sp:SignedParts>
          <sp:EncryptedParts>
            <sp:Body />
          </sp:EncryptedParts>
        </wsp:All>
      </wsp:ExactlyOne>
    </wsp:Policy>

    <wsdl:binding name="Symmetric_binding" type="tns:SecurityTokenService">
      <wsp:PolicyReference URI="#STS_endpoint_policy"/>
      <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="Issue">
        <soap12:operation
```

```
soapAction="http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue"
            style="document"/>
          <wsdl:input>
            <wsp:PolicyReference URI="#STS_message_policy"/>
            <soap12:body use="literal"/>
          </wsdl:input>
          <wsdl:output>
            <wsp:PolicyReference URI="#STS_message_policy"/>
            <soap12:body use="literal"/>
          </wsdl:output>
        </wsdl:operation>
      </wsdl:binding>

      <wsdl:service name="STS_0">
        <wsdl:port name="STS_0_port" binding="tns:Symmetric_binding">
          <soap12:address location="http://contoso.com/sts"/>
          <wsa:EndpointReference>
            <wsa:Address>http://contoso.com/sts</wsa:Address>
            <wsid:Identity>
              <ds:KeyInfo>
                <ds:X509Data>
                  <ds:X509Certificate>
                    [base64 encoded certificate value]
                  </ds:X509Certificate>
                </ds:X509Data>
              </ds:KeyInfo>
            </wsid:Identity>
          </wsa:EndpointReference>
        </wsdl:port>
      </wsdl:service>
    </wsdl:definitions>
  </MetadataSection>

  <MetadataSection
      Dialect="http://www.w3.org/2001/XMLSchema"
      Identifier="http://contoso.com/schemas">
    <xs:schema xmlns:tns="http://contoso.com/schemas"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified"
        targetNamespace="http://contoso.com/schemas">
      <xs:complexType name="MessageBody">
        <xs:sequence>
          <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##any"/>
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
  </MetadataSection>
</Metadata>
```

Note that the token assertion required for client authentication is shown as a placeholder with the text *"[Authentication Token Assertion]"* inside the security policy and highlighted in the metadata example above. The authentication token assertion for each type of required client credential is described in later sections. Other metadata content that would need to be substituted when used with a real IP/STS is also highlighted.

5.1.2. Message exchange

Windows CardSpace retrieves the WSDL of the IP/STS including its policy using the WS-Transfer/Get request mechanism specified in [WS-MetadataExchange].

The following SOAP request/response messages illustrate this exchange.

Metadata request from service requester to IP/STS:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action S:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
    </wsa:Action>
    <wsa:MessageID>
      urn:uuid:ab9e1c77-0cea-4f2f-a586-78c15536137d
    </wsa:MessageID>
    <wsa:To S:mustUnderstand="1">https://contoso.com/sts/mex</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>
        http://www.w3.org/2005/08/addressing/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
  </S:Header>
  <S:Body />
</S:Envelope>
```

Note the following in the metadata request message:

- The request is directed at an endpoint secured using the HTTPS transport.
- The request does not specify any specific metadata dialect causing all available metadata at that endpoint to be returned.

Metadata response from IP/STS to service requester:

```
<S:Envelope ...>
 <S:Header>
  <wsa:Action S:mustUnderstand="1">
    http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
  </wsa:Action>
  <wsa:RelatesTo>
    urn:uuid:ab9e1c77-0cea-4f2f-a586-78c15536137d
  </wsa:RelatesTo>
 </S:Header>
 <S:Body>
  <Metadata xmlns="http://schemas.xmlsoap.org/ws/2004/08/mex">
    [The metadata containing a WSDL metadata section and a
     XML schema metadata section as shown in the previous
     subsection goes here]
  </Metadata>
 </S:Body>
</S:Envelope>
```

Note the following in the metadata response message:

- One or more metadata sections may be returned in the response, each section containing a different type of metadata or part of a metadata type, *e.g.*, WSDL with message and port type definitions, policy declarations, bindings with policy attachments, or an XML schema for type definitions used.

5.2. Authenticating with Username and Password

The identity provider requires that the service requester submit a *username* and *password* as the credential to authenticate to the IP/STS when requesting tokens.

5.2.1. Credential format

The credential descriptor format for username/password defined in [CardSpace-Ref] has the following form:

```
<ic:UserCredential>
  <ic:UsernamePasswordCredential>
    <ic:Username>zoe</ic:Username>
  </ic:UsernamePasswordCredential>
</ic:UserCredential>
```

For convenience of the user, the “username” value can be optionally included in the information card in the `ic:Username` element of the credential descriptor as shown in the example above. The user will be prompted to supply the “password” when the information card is selected for use.

5.2.2. Security policy

Transport security using the “transport binding” should be used for token requests using this authentication method. As an alternative, message security using the “symmetric binding” may also be used for token requests using this authentication method.

The authentication token assertion in security policy that should be used inside the WSDL of the IP/STS, as described in Section 5.1.1, is shown below. This token assertion can be used regardless of whether transport binding (Section 5.1.1.1) or symmetric binding (Section 5.1.1.2) is used.

Authentication token assertion in security policy:

```
<sp:SignedSupportingTokens>
  <wsp:Policy>
    <sp:UsernameToken
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/Include
Token/AlwaysToRecipient">
      <wsp:Policy>
        <sp:WssUsernameToken10/>
      </wsp:Policy>
    </sp:UsernameToken>
  </wsp:Policy>
</sp:SignedSupportingTokens>
```

5.2.3. Message exchange

This section provides the SOAP message exchanges when transport security with “transport binding” is used by the IP/STS. For this security binding, message protection and security correlation for the request and response legs of the message exchange is provided by the secure HTTPS transport. There is no message level encryption required.

The following SOAP messages show the request/response exchange when transport security is used (see WSDL and policy for transport security in Section 5.1.1.1). The exchange when message security is used is shown in later sections for other credential types.

Token request from service requester to IP/STS:

```
<S:Envelope ...>
   <S:Header>
```

```
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue
    </wsa:Action>
    <wsa:MessageID wsu:Id="_2">
      uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:MessageID>
    <wsa:To wsu:Id="_3">
      https://contoso.com/sts
    </wsa:To>
    <wsa:ReplyTo wsu:Id="_4">
      <wsa:Address>
        http://www.w3.org/2005/08/addressing/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>
      <!-- Username w/ cleartext password as authentication token -->
      <wsse:UsernameToken wsu:Id="_6">
        <wsse:Username>Zoe</wsse:Username>
        <wsse:Password
            Type="http:// http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">
          ILoveDogs
        </wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id=”_10”>
    <wst:RequestSecurityToken>
      <wst:TokenType>urn:oasis:names:tc:SAML:1.0:assertion</wst:TokenType>
      <wst:RequestType>
          http://schemas.xmlsoap.org/ws/2005/02/trust/Issue
      </wst:RequestType>
      <wst:KeyType>
        http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey
      </wst:KeyType>
      <wst:KeySize>256</wst:KeySize>
      <wst:Entropy>
        <wst:BinarySecret>mQlxWxEifnHgQpylcD7LYSkJplpE=</wst:BinarySecret>
      </wst:Entropy>
      <wsp:AppliesTo>
        <wsa:EndpointReference>
          <wsa:Address>http://www.relying-party.com</wsa:Address>
          <wsid:Identity>...</wsid:Identity>
        </wsa:EndpointReference>
      </wsp:AppliesTo>
      <ic:InformationCardReference>
        <ic:CardId>http://contoso.com/id/d795621fa01d454285f9</ic:CardId>
      </ic:InformationCardReference>
      <wst:Claims
          wst:Dialect=”http://schemas.xmlsoap.org/ws/2005/05/identity”>
        <ic:ClaimType Uri=”http://.../ws/2005/05/identity/claims/givenname”/>
        <ic:ClaimType Uri=”http://.../ws/2005/05/identity/claims/surname”/>
      </wst:Claims>
```

```
      <ic:RequestDisplayToken xml:lang="en-us" />
    </wst:RequestSecurityToken>
  </S:Body>
</S:Envelope>
```

Note the following in the request message:

- The request is sent over HTTPS since a username/password token is used for authentication.
- A symmetric proof key is requested for which client-entropy is also included.
- Relying party information in the form of an endpoint reference and its identity token is communicated to the IP/STS via the `wsp:AppliesTo` element (the example shown assumes that the IP/STS specified the `ic:RequireAppliesTo` assertion in the information card).
- The information card reference (`CardId`) is included.
- A display token localized in "US English" is requested.

Token response from IP/STS to service requester:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue
    </wsa:Action>
    <wsa:RelatesTo wsu:Id="_2">
      uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:RelatesTo>
    <wsa:To wsu:Id="_3">
      http://www.w3.org/2005/08/addressing/anonymous
    </wsa:To>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id=”_10”>
    <wst:RequestSecurityTokenResponse>
      <wst:TokenType>urn:oasis:names:tc:SAML:1.0:assertion</wst:TokenType>
      <wst:Lifetime>
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wst:Lifetime>
      <wst:RequestedSecurityToken>
        <!-- Start encrypted token
        <saml:Assertion xmlns="urn:oasis:names:tc:SAML:1.1:assertion"
          AssertionID=”uuid:17e2007e-f959-4624-85ef-ae00df6fe071” ...>
          ...
        </saml:Assertion>
        End encrypted token -->
        <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element">
          <xenc:EncryptionMethod
            Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
          <ds:KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <xenc:EncryptedKey>
```

```
              <xenc:EncryptionMethod
                Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
                <ds:DigestMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
              </xenc:EncryptionMethod>
              <ds:KeyInfo>
                <!-- token encryption key is encrypted to certificate
                     of relying party -->
                <wsse:SecurityTokenReference>
                  <wsse:KeyIdentifier
                    ValueType="http://docs.oasis-open.org/wss/2004/xx/oasis-
2004xx-wss-soap-message-security-1.1#ThumbprintSHA1"
                    EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis200401-wss-soap-message-security-1.0#Base64Binary">
                    +PYbznDaB/dlhjIfqCQ458E72wA=
                  </wsse:KeyIdentifier>
                </wsse:SecurityTokenReference>
              </ds:KeyInfo>
              <xenc:CipherData>
                <xenc:CipherValue>...</xenc:CipherValue>
              </xenc:CipherData>
            </xenc:EncryptedKey>
          </ds:KeyInfo>
          <xenc:CipherData>
            <xenc:CipherValue>...</xenc:CipherValue>
          </CipherData>
        </EncryptedData>
      </wst:RequestedSecurityToken>
      <wst:RequestedAttachedReference>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/
oasis-wss-saml-token-profile-1.0#SAMLAssertionID">
             uuid:17e2007e-f959-4624-85ef-ae00df6fe071
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </wst:RequestedAttachedReference>
      <wst:RequestedUnattachedReference>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/
oasis-wss-saml-token-profile-1.0#SAMLAssertionID">
             uuid:17e2007e-f959-4624-85ef-ae00df6fe071
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </wst:RequestedUnattachedReference>
      <wst:RequestedProofToken>
        <wst:ComputedKey>
          http://schemas.xmlsoap.org/ws/2005/02/trust/CK/PSHA1
        </wst:ComputedKey>
      </wst:RequestedProofToken>
      <wst:Entropy>
        <wst:BinarySecret Type="http://.../ws/2005/02/trust/Nonce>
          u+Qe3WdkFYqZsfwT9ZU6qTu9LqIYtwNz
        </wst:BinarySecret>
      </wst:Entropy>
      <wst:KeyType>
        http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey
      </wst:KeyType>
```

```
      <wst:KeySize>256</wst:KeySize>
      <ic:RequestedDisplayToken>
       <ic:DisplayToken xml:lang="en-us">
        <ic:DisplayClaim Uri="http://.../identity/claims/givenname">
          <ic:DisplayTag>Given Name</ic:DisplayTag>
          <ic:DisplayValue>John</ic:DisplayValue>
        </ic:DisplayClaim>
        <ic:DisplayClaim Uri="http://.../identity/claims/surname">
          <ic:DisplayTag>Last Name</ic:DisplayTag>
          <ic:DisplayValue>Doe</ic:DisplayValue>
        </ic:DisplayClaim>
       </ic:DisplayToken>
      </ic:RequestedDisplayToken>
    </wst:RequestSecurityTokenResponse>
  </S:Body>
</S:Envelope>
```

Note the following in the response message:

- The response is sent over HTTPS.
- The issued security token is encrypted to the relying party since information about the relying party and its identity token were conveyed in the request.
- Since the SAML token doesn't support references using URI fragments (XML Id), attached and unattached references are returned whose element content can be used *verbatim* within a `wsse:SecurityTokenReference` element to reference the token when it is placed inside a message.
- A symmetric proof key, based on client and server entropies, is returned.
- A display token containing textual representation of the actual token is returned.

5.3. Authenticating with KerberosV5 Service Ticket

The identity provider requires that the service requester submit a *Kerberos v5 service ticket* as the credential to authenticate to the IP/STS when requesting tokens.

5.3.1. Credential format

No explicit user credential needs to be specified in this case as it is implied by the Kerberos realm that the user logs into. The credential descriptor format for Kerberos v5 defined in [CardSpace-Ref] has the following form:

```
<ic:UserCredential>
  <ic:KerberosV5Credential />
</ic:UserCredential>
```

To enable the service requester to obtain a Kerberos v5 service ticket for the IP/STS, the endpoint reference of the IP/STS in the information card or in the metadata retrieved from it must include a "service principal name" identity claim under the `wsid:Identity` tag as defined in [Addressing-Ext]. An example is shown below.

```
<wsa:EndpointReference>
  <wsa:Address>http://contoso.com/sts</wsa:Address>
  <wsid:Identity>
    <wsid:SpnClaim>host/corp-sts.contoso.com</wsid:SpnClaim>
  </wsid:Identity>
</wsa:EndpointReference>
```

The KDC in the appropriate domain/realm can identify the IP/STS service account based on the service principal name information and issue the required service ticket. This would typically be used in enterprise intranet scenarios.

5.3.2. Security policy

Message security using the “symmetric binding” should be used for token requests using this authentication method. The content of the sp:ProtectionToken assertion in security policy shown in Section 5.1.1.2 should be replaced by the partial policy fragment shown below.

Protection token assertion in security policy:

```
<sp:ProtectionToken>
  <wsp:Policy>
    <sp:KerberosToken
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/Include
Token/Once">
      <wsp:Policy>
        <sp:WssGssKerberosV5ApReqToken11/>
      </wsp:Policy>
    </sp:KerberosToken>
  </wsp:Policy>
</sp:ProtectionToken>
```

Since the the Kerberos token already carries a symmetric session key that can be used as the basis for message security, no separate authentication token assertion in security policy is required in this case.

5.3.3. Message exchange

This section provides the SOAP message exchanges when message security with “symmetric binding” is used by the IP/STS. For this security binding, message protection and security correlation for the request and response legs of the message exchange is provided by the symmetric session key in the attached KerberosV5 service ticket. Message integrity and confidentiality is governed by the policy attached to individual messages as described in Section 5.1.1.2.

The following SOAP messages show the request/response exchange when message security is used. The exchange when transport security is used is shown in the earlier section for the username/password credential type.

Token request from service requester to IP/STS:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue
    </wsa:Action>
    <wsa:MessageID wsu:Id="_2">
      urn:uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:MessageID>
    <wsa:To wsu:Id="_3">http://contoso.com/sts</wsa:To>
    <wsa:ReplyTo wsu:Id="_4">
      <wsa:Address>
        http://www.w3.org/2005/08/addressing/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
    <wsse:Security S:mustUnderstand="1">
```

```
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>
      <!-- Kerberosv5 service ticket as authentication token -->
      <wsse:BinarySecurityToken wsu:Id="_30"
        ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-
profile-1.1#GSS_Kerberosv5_AP_REQ"
        EncodingType=" http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
          MIIEZzCCA9CgAwIBAgIQEmtJZc0==
      </wsse:BinarySecrityToken>
      <!-- List of encrypted elements in the message per
           message confidentiality policy -->
      <xenc:ReferenceList>
         <xenc:DataReference URI="#_20" />
      </xenc:ReferenceList>
      <!—Signature using the Kerberosv5 service ticket  -->
      <ds:Signature wsu:Id=”_33”>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
          <ds:Reference URI="#_6">...</ds:Reference>
          <ds:Reference URI="#_1">...</ds:Reference>
          <ds:Reference URI="#_2">...</ds:Reference>
          <ds:Reference URI="#_3">...</ds:Reference>
          <ds:Reference URI="#_4">...</ds:Reference>
          <ds:Reference URI="#_10">...</ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#_30"
              ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-
token-profile-1.1#GSS_Kerberosv5_AP_REQ"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id=”_10”>
    <!-- Start encrypted Content
    <wst:RequestSecurityToken>
      <wst:TokenType>urn:oasis:names:tc:SAML:1.0:assertion</wst:TokenType>
      <wst:RequestType>
          http://schemas.xmlsoap.org/ws/2005/02/trust/Issue
      </wst:RequestType>
      <wst:KeyType>
        http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey
      </wst:KeyType>
      <wst:KeySize>256</wst:KeySize>
      <wst:Entropy>
        <wst:BinarySecret>mQlxWxEifnHgQpylcD7LYSkJplpE=</wst:BinarySecret>
      </wst:Entropy>
      <wsp:AppliesTo>
```

```
        <wsa:EndpointReference>
          <wsa:Address>http://www.relying-party.com</wsa:Address>
          <wsid:Identity>...</wsid:Identity>
        </wsa:EndpointReference>
      </wsp:AppliesTo>
      <ic:InformationCardReference>
        <ic:CardId> http://contoso.com/id/d795621fa01d454285f9</ic:CardId>
      </ic:InformationCardReference>
      <wst:Claims
          wst:Dialect=”http://schemas.xmlsoap.org/ws/2005/05/identity”>
        <ic:ClaimType Uri=”http://.../identity/claims/givenname”/>
        <ic:ClaimType Uri=”http://.../identity/claims/surname”/>
      </wst:Claims>
      <ic:RequestDisplayToken xml:lang="en-us" />
    </wst:RequestSecurityToken>
    End encrypted content -->
    <xenc:EncryptedData Id="_20">
      <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#_30"
            ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-
token-profile-1.1#GSS_Kerberosv5_AP_REQ"/>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>...</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

Note the following in the request message:

- The ordering of items in the security header follows the strict layout as prescribed by [WS-SecurityPolicy].
- A symmetric proof key is requested for which client-entropy is included.
- Relying party information in the form of an endpoint reference and its identity token is communicated to the IP/STS via the `wsp:AppliesTo` element (the example shown assumes that the IP/STS specified the `ic:RequireAppliesTo` assertion in the information card).
- The information card reference (CardId) is included.
- A display token localized in “US English” is requested.
- The Kerberos service ticket is included as a binary security token in the SOAP security header.
- The message is signed with the session key in the Kerberos service ticket; but the service ticket itself is NOT included within the scope of the message signature.
- Encrypted message elements are encrypted with the session key in the Kerberos service ticket.
- References to the Kerberos service ticket included in the message is made using the `wsse:Reference` based direct reference.

Token response from IP/STS to service requester:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue
    </wsa:Action>
    <wsa:RelatesTo wsu:Id="_2">
      urn:uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:RelatesTo>
    <wsa:To wsu:Id="_3">
      http://www.w3.org/2005/08/addressing/anonymous
    </wsa:To>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>
      <!-- List of encrypted elements in the message per
           message confidentiality policy -->
      <xenc:ReferenceList>
         <xenc:DataReference URI="#_20" />
      </xenc:ReferenceList>
      <!-- Signature using the Kerberosv5 service ticket  -->
      <ds:Signature wsu:Id=”_33”>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
          <ds:Reference URI="#_6">...</ds:Reference>
          <ds:Reference URI="#_1">...</ds:Reference>
          <ds:Reference URI="#_2">...</ds:Reference>
          <ds:Reference URI="#_3">...</ds:Reference>
          <ds:Reference URI="#_10">...</ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentfier
              ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-
token-profile-1.1#Kerberosv5APREQSHA1">
                xqBw9N99tkxs4UH2TvyD06Ikj5k=
            </wsse:KeyIdentfier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id=”_10”>
    <!-- Start encrypted Content
    <wst:RequestSecurityTokenResponse>
      <wst:TokenType>
        urn:oasis:names:tc:SAML:1.0:assertion
      </wst:TokenType>
      <wst:Lifetime>
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
```

```
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wst:Lifetime>
      <wst:RequestedSecurityToken>
        <!-- Start encrypted token
        <saml:Assertion xmlns="urn:oasis:names:tc:SAML:1.1:assertion"
          AssertionID="uuid:17e2007e-f959-4624-85ef-ae00df6fe071" ...>
          ...
        </saml:Assertion>
        End encrypted token -->
        <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element">
          <xenc:EncryptionMethod
            Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
          <ds:KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <xenc:EncryptedKey>
              <xenc:EncryptionMethod
                Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
                <ds:DigestMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
              </xenc:EncryptionMethod>
              <ds:KeyInfo>
                <!-- token encryption key is encrypted to certificate
                     of relying party -->
                <wsse:SecurityTokenReference>
                  <wsse:KeyIdentifier
                    ValueType="http://docs.oasis-open.org/wss/2004/xx/oasis-
2004xx-wss-soap-message-security-1.1#ThumbprintSHA1"
                    EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis200401-wss-soap-message-security-1.0#Base64Binary">
                    +PYbznDaB/dlhjIfqCQ458E72wA=
                  </wsse:KeyIdentifier>
                </wsse:SecurityTokenReference>
              </ds:KeyInfo>
              <xenc:CipherData>
                <xenc:CipherValue>...</xenc:CipherValue>
              </xenc:CipherData>
            </xenc:EncryptedKey>
          </ds:KeyInfo>
          <xenc:CipherData>
            <xenc:CipherValue>...</xenc:CipherValue>
          </CipherData>
        </EncryptedData>
      </wst:RequestedSecurityToken>
      <wst:RequestedAttachedReference>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/
oasis-wss-saml-token-profile-1.0#SAMLAssertionID">
             uuid:17e2007e-f959-4624-85ef-ae00df6fe071
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </wst:RequestedAttachedReference>
      <wst:RequestedUnattachedReference>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/
oasis-wss-saml-token-profile-1.0#SAMLAssertionID">
             uuid:17e2007e-f959-4624-85ef-ae00df6fe071
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
```

```
      </wst:RequestedUnattachedReference>
      <wst:RequestedProofToken>
        <wst:ComputedKey>
          http://schemas.xmlsoap.org/ws/2005/02/trust/CK/PSHA1
        </wst:ComputedKey>
      </wst:RequestedProofToken>
      <wst:Entropy>
        <wst:BinarySecret Type="http://.../ws/2005/02/trust/Nonce>
          u+Qe3WdkFYqZsfwT9ZU6qTu9LqIYtwNz
        </wst:BinarySecret>
      </wst:Entropy>
      <wst:KeyType>
        http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey
      </wst:KeyType>
      <wst:KeySize>256</wst:KeySize>
      <ic:RequestedDisplayToken>
       <ic:DisplayToken xml:lang="en-us">
        <ic:DisplayClaim Uri=”http://.../identity/claims/givenname”>
          <ic:DisplayTag>Given Name</ic:DisplayTag>
          <ic:DisplayValue>John</ic:DisplayValue>
        </ic:DisplayClaim>
        <ic:DisplayClaim Uri=”http://.../identity/claims/surname”>
          <ic:DisplayTag>Last Name</ic:DisplayTag>
          <ic:DisplayValue>Doe</ic:DisplayValue>
        </ic:DisplayClaim>
       </ic:DisplayToken>
      </ic:RequestedDisplayToken>
    </wst:RequestSecurityTokenResponse>
    End encrypted content -->
    <xenc:EncryptedData Id="_20">
      <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentfier
            ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-
token-profile-1.1#Kerberosv5APREQSHA1">
              xqBw9N99tkxs4UH2TvyD06Ikj5k=
          </wsse:KeyIdentfier>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>...</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

Note the following in the response message:

- The ordering of items in the security header follows the strict layout as prescribed by [WS-SecurityPolicy].
- The Kerberos service ticket is NOT included in the message. References to the Kerberos token are made indirectly using a SHA1 thumbprint based key identifier reference using the `wsse:KeyIdentifier` element.
- The message is signed with the session key in the Kerberos token.

- Encrypted message elements are encrypted with the key in the Kerberos token.
- The issued security token is encrypted to the relying party since information about the relying party and its identity token were conveyed in the request.
- Since the SAML token doesn't support references using URI fragments (XML Id), attached and unattached references are returned whose element content can be used *verbatim* within a `wsse:SecurityTokenReference` element to reference the token when it is placed inside a message.
- A symmetric proof key, based on client and server entropy is returned.
- A display token containing textual representation of the actual token is returned.

5.4. Authenticating with X.509v3 Certificate

The identity provider requires that the service requester submit an *X.509 v3 certificate*, where the certificate and keys may be in a hardware-based smart card or a software-based certificate, as the credential to authenticate to the IP/STS when requesting tokens.

5.4.1. Credential format

To enable the service requester to locate the right X.509 certificate for use, the SHA1 hash of the entire certificate (i.e., a certificate thumbprint) should be specified as the credential descriptor in the information card. The thumbprint value can be used with the appropriate platform-specific APIs (e.g. CAPI2 on Windows) to locate the certificate. When using a smart card based credential, a textual hint should be included in the `ic:DisplayCredentialHint` element of the credential type that will be used to prompt the user to insert the appropriate smart card in the reader.

The credential descriptor format for hardware-based X.509 certificate defined in [CardSpace-Ref] has the following form:

```
<ic:UserCredential>
  <ic:DisplayCredentialHint>
    Please insert your corporate smart card
  </ic:DisplayCredentialHint>
  <ic:X509V3Credential>
    <ds:X509Data>
      <wsse:KeyIdentifier
        ValueType="http://docs.oasis-open.org/wss/2004/xx/oasis-2004xx-wss-
soap-message-security-1.1#ThumbprintSHA1"
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis200401-wss-
soap-message-security-1.0#Base64Binary">
          ... Thp8EqU0S+A4Qu+==
      </wsse:KeyIdentifier>
    </ds:X509Data>
  </ic:X509V3Credential>
</ic:UserCredential>
```

IMPORTANT NOTE: Notice that the URI used as the value of the `ValueType` attribute on the `wsse:KeyIdentifier` element to indicate a SHA1 thumbprint based key identifier is a slightly outdated URI value. The new prescribed value as per the WS-Security v1.1 standard should be as follows:

http://docs.oasisopen.org/wss/oasiswss-soap-messagesecurity-1.1#ThumbPrintSHA1

Support for this newer URI will be added in a future release of Windows CardSpace.

5.4.2. Security policy

Message security using the “symmetric binding” should be used for token requests using this authentication method. As an alternative, transport security using the “transport binding” may also be used for token requests using this authentication method.

To enable the service requester to obtain the security token of the IP/STS for securing messages, the endpoint reference of the IP/STS in the information card or in the WSDL retrieved must include its X.509v3 certificate in the `wsid:Identity` tag as defined in [Addressing-Ext].

The authentication token assertion in security policy that should be used inside the WSDL of the IP/STS, as described in Section 5.1.1, is shown below. This token assertion can be used regardless of whether transport binding (Section 5.1.1.1) or symmetric binding (Section 5.1.1.2) is used. The user’s X.509v3 certificate is submitted as an endorsing supporting token in the RST request

Authentication token assertion in security policy:

```
<sp:EndorsingSupportingTokens>
  <wsp:Policy>
    <sp:X509Token
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/Include
Token/AlwaysToRecipient">
      <wsp:Policy>
        <sp:WssX509V3Token10/>
      </wsp:Policy>
    </sp:X509Token>
  </wsp:Policy>
</sp:EndorsingSupportingTokens>
```

5.4.3. Message exchange

This section provides the SOAP message exchanges when message security with “symmetric binding” is used by the IP/STS. For this security binding, message protection and security correlation for the request and response legs of the message exchange is provided by an ephemeral symmetric session key. Message integrity and confidentiality is governed by the policy attached to individual messages as described in Section 5.1.1.2.

The following SOAP messages show the request/response exchange when message security is used. The exchange when transport security is used is shown in the earlier section for the username/password credential type.

Token request from service requester to IP/STS:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue
    </wsa:Action>
    <wsa:MessageID wsu:Id="_2">
      urn:uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:MessageID>
    <wsa:To wsu:Id="_3">http://contoso.com/sts</wsa:To>
    <wsa:ReplyTo wsu:Id="_4">
      <wsa:Address>
        http://www.w3.org/2005/08/addressing/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
    <wsse:Security S:mustUnderstand="1">
```

```
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>
      <!-- Symmetric session key encrypted to the X.509 certificate
           of the IP/STS endpoint  -->
      <xenc:EncryptedKey Id="_30">
        <xenc:EncryptionMethod
            Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
          <ds:DigestMethod Algorithm="http://.../2000/09/xmldsig#sha1"/>
        </xenc:EncryptionMethod>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier
              ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-
message-security-1.1#ThumbprintSHA1"
              EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-soap-message-security-1.0#Base64Binary">
              +PYbznDaB/dlhjIfqCQ458E72wA=
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>...</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedKey>
      <!-- List of encrypted elements in the message per the message
           confidentiality policy -->
      <xenc:ReferenceList>
        <xenc:DataReference URI="#_20" />
      </xenc:ReferenceList>
      <!-- X.509 certificate of the user as the endorsing token -->
      <wsse:BinarySecurityToken wsu:Id="_33"
        ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0#X509v3"
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
          ...
      </wsse:BinarySecrityToken>
      <!-- Primary message signature using the symmetric session key  -->
      <ds:Signature wsu:Id=”_40”>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
          <ds:Reference URI="#_6">...</ds:Reference>
          <ds:Reference URI="#_1">...</ds:Reference>
          <ds:Reference URI="#_2">...</ds:Reference>
          <ds:Reference URI="#_3">...</ds:Reference>
          <ds:Reference URI="#_4">...</ds:Reference>
          <ds:Reference URI="#_10">...</ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#_30"
```

```
              ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-
message-security-1.1#EncryptedKey" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
      <!-- Endorsing signature using the user's X.509 certificate
           endorsing the primary message signature  -->
      <ds:Signature wsu:Id="_43">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <ds:Reference URI="#_40">...</ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#_33"
              ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0#X509v3" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>

  <S:Body wsu:Id="_10">
    <!-- Start encrypted Content
    <wst:RequestSecurityToken>
      <wst:TokenType>
        urn:oasis:names:tc:SAML:1.0:assertion
      </wst:TokenType>
      <wst:RequestType>
          http://schemas.xmlsoap.org/ws/2005/02/trust/Issue
      </wst:RequestType>
      <wst:KeyType>
        http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey
      </wst:KeyType>
      <wst:KeySize>256</wst:KeySize>
      <wst:Entropy>
        <wst:BinarySecret>mQlxWxEifnHgQpylcD7LYSkJplpE=</wst:BinarySecret>
      </wst:Entropy>
      <wsp:AppliesTo>
        <wsa:EndpointReference>
          <wsa:Address>http://www.relying-party.com</wsa:Address>
          <wsid:Identity>...</wsid:Identity>
        </wsa:EndpointReference>
      </wsp:AppliesTo>
      <ic:InformationCardReference>
        <ic:CardId>http://contoso.com/id/d795621fa01d454285f9</ic:CardId>
      </ic:InformationCardReference>
      <wst:Claims
          wst:Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity">
        <ic:ClaimType Uri="http://.../identity/claims/givenname"/>
        <ic:ClaimType Uri="http://.../identity/claims/surname"/>
      </wst:Claims>
```

```
      <ic:RequestDisplayToken xml:lang="en-us" />
    </wst:RequestSecurityToken>
    End encrypted content -->
    <xenc:EncryptedData Id="_20">
      <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
            <wsse:Reference URI="#_30"
              ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-
message-security-1.1#EncryptedKey" />
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>...</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

Note the following in the request message:

- The ordering of items in the security header follows the strict layout as prescribed by [WS-SecurityPolicy].
- A symmetric proof key is requested for which client-entropy is included.
- Relying party information in the form of an endpoint reference and its identity token is communicated to the IP/STS via the `wsp:AppliesTo` element (the example shown assumes that the IP/STS specified the `ic:RequireAppliesTo` assertion in the information card).
- The information card reference (CardId) is included.
- A display token localized in “US English” is requested.
- The X.509 certificate of the IP/STS is NOT included in the message. References to it are made indirectly using a SHA1 thumbprint based key identifier reference using the `wsse:KeyIdentifier` element since the `sp:ProtectionToken` assertion in the STS policy includes the `sp:RequireThumbprintReference` policy assertion.
- An ephemeral symmetric session key is generated and encrypted to the X.509 certificate of the IP/STS endpoint. The message is signed with this symmetric session key which constitutes the primary message signature.
- The primary message signature is further signed by the key in the user’s X.509 certificate (endorsing signature) which is used to authenticate the user. The X.509 certificate itself is NOT covered by the message signature or the endorsing signature.
- The X.509 client certificate is included in its entirety in the SOAP security header since the `sp:EndorsingSupportingToken` assertion in the IP/STS policy does not include the `sp:RequireThumbprintReference` policy assertion.
- References to the encrypted session key and the X.509 certificates included in the message are made using the `wsse:Reference` based direct references.

Token response from IP/STS to service requester:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
```

```
      http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue
    </wsa:Action>
    <wsa:RelatesTo wsu:Id="_2">
      urn:uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:RelatesTo>
    <wsa:To wsu:Id="_3">
      http://www.w3.org/2005/08/addressing/anonymous
    </wsa:To>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>
      <!-- List of encrypted elements in the message per
           message confidentiality policy -->
      <xenc:ReferenceList>
        <xenc:DataReference URI="#_20" />
      </xenc:ReferenceList>
      <!-- Message signature using the symmetric session key -->
      <ds:Signature wsu:Id=”_33”>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
          <ds:Reference URI="#_6">...</ds:Reference>
          <ds:Reference URI="#_1">...</ds:Reference>
          <ds:Reference URI="#_2">...</ds:Reference>
          <ds:Reference URI="#_3">...</ds:Reference>
          <ds:Reference URI="#_10">...</ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier
              ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-
message-security-1.1#EncryptedKeySHA1">
                AcLJ9234LIl2HbBwbpk0qBPhVZ8=
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>

  <S:Body wsu:Id=”_10”>
    <!-- Start encrypted Content
    <wst:RequestSecurityTokenResponse>
      <wst:TokenType>
        urn:oasis:names:tc:SAML:1.0:assertion
      </wst:TokenType>
      <wst:Lifetime>
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wst:Lifetime>
      <wst:RequestedSecurityToken>
        <!-- Start encrypted token
```

```
        <saml:Assertion xmlns="urn:oasis:names:tc:SAML:1.1:assertion"
          AssertionID="uuid:17e2007e-f959-4624-85ef-ae00df6fe071" ...>
          ...
        </saml:Assertion>
        End encrypted token -->
        <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element">
          <xenc:EncryptionMethod
            Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
          <ds:KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <xenc:EncryptedKey>
              <xenc:EncryptionMethod
                Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
                <ds:DigestMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
              </xenc:EncryptionMethod>
              <ds:KeyInfo>
                <!-- token encryption key is encrypted to certificate
                     of relying party -->
                <wsse:SecurityTokenReference>
                  <wsse:KeyIdentifier
                    ValueType="http://docs.oasis-open.org/wss/2004/xx/oasis-
2004xx-wss-soap-message-security-1.1#ThumbprintSHA1"
                    EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis200401-wss-soap-message-security-1.0#Base64Binary">
                    +PYbznDaB/dlhjIfqCQ458E72wA=
                  </wsse:KeyIdentifier>
                </wsse:SecurityTokenReference>
              </ds:KeyInfo>
              <xenc:CipherData>
                <xenc:CipherValue>...</xenc:CipherValue>
              </xenc:CipherData>
            </xenc:EncryptedKey>
          </ds:KeyInfo>
          <xenc:CipherData>
            <xenc:CipherValue>...</xenc:CipherValue>
          </CipherData>
        </EncryptedData>
      </wst:RequestedSecurityToken>
      <wst:RequestedAttachedReference>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/
oasis-wss-saml-token-profile-1.0#SAMLAssertionID">
             uuid:17e2007e-f959-4624-85ef-ae00df6fe071
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </wst:RequestedAttachedReference>
      <wst:RequestedUnattachedReference>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/
oasis-wss-saml-token-profile-1.0#SAMLAssertionID">
             uuid:17e2007e-f959-4624-85ef-ae00df6fe071
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </wst:RequestedUnattachedReference>
      <wst:RequestedProofToken>
        <wst:ComputedKey>
          http://schemas.xmlsoap.org/ws/2005/02/trust/CK/PSHA1
```

```
        </wst:ComputedKey>
      </wst:RequestedProofToken>
      <wst:Entropy>
        <wst:BinarySecret Type="http://.../ws/2005/02/trust/Nonce>
          u+Qe3WdkFYqZsfwT9ZU6qTu9LqIYtwNz
        </wst:BinarySecret>
      </wst:Entropy>
      <wst:KeyType>
        http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey
      </wst:KeyType>
      <wst:KeySize>256</wst:KeySize>
      <ic:RequestedDisplayToken>
       <ic:DisplayToken xml:lang="en-us">
        <ic:DisplayClaim Uri=”http://.../identity/claims/givenname”>
          <ic:DisplayTag>Given Name</ic:DisplayTag>
          <ic:DisplayValue>John</ic:DisplayValue>
        </ic:DisplayClaim>
        <ic:DisplayClaim Uri=”http://.../identity/claims/surname”>
          <ic:DisplayTag>Last Name</ic:DisplayTag>
          <ic:DisplayValue>Doe</ic:DisplayValue>
        </ic:DisplayClaim>
       </ic:DisplayToken>
      </ic:RequestedDisplayToken>
    </wst:RequestSecurityTokenResponse>
    End encrypted content -->
    <xenc:EncryptedData Id="_20">
      <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier
            ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-
security-1.1#EncryptedKeySHA1">
              AcLJ9234LIl2HbBwbpk0qBPhVZ8=
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>...</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

Note the following in the response message:

- The ordering of items in the security header follows the strict layout as prescribed by [WS-SecurityPolicy].
- The message is signed with the symmetric session key that was included in the RST request. Encrypted message elements are encrypted with the same symmetric session key.
- References to the symmetric session key, which is NOT included in the message, are made indirectly using a SHA1 thumbprint based key identifier reference using the `wsse:KeyIdentifier` element.

- The issued security token is encrypted to the relying party since information about the relying party and its identity token were conveyed in the request.
- Since the SAML token doesn't support references using URI fragments (XML Id), attached and unattached references are returned whose element content can be used *verbatim* within a `wsse:SecurityTokenReference` element to reference the token when it is placed inside a message.
- A symmetric proof key, based on client and server entropies, is returned.
- A display token containing textual representation of the actual token is returned.

5.5. Authenticating with Self-issued Token

The identity provider requires that the service requester submit a *self-issued SAML token* as the credential to authenticate to the IP/STS when requesting tokens.

5.5.1. Credential format

To enable the service requester to locate the right self-issued information card as the credential, the PPID value that identifies the user at the IP/STS should be specified as the credential descriptor in the information card. The PPID value can be used to locate the self-issued information card which produces that value for the IP/STS.

The credential descriptor format for self-issued token defined in [CardSpace-Ref] has the following form. The PPID is specified as the value of the `ic:PrivatePersonalIdentifier` element.

```
<ic:UserCredential>
  <ic:SelfIssuedCredential>
    <ic:PrivatePersonalIdentifier>
      xqh78FgyuThp8EqU0S+A4Qu+=
    </ic:PrivatePersonalIdentifier>
  </ic:SelfIssuedCredential>
</ic:UserCredential>
```

5.5.2. Security policy

Message security using the “symmetric binding” should be used for token requests using this authentication method. As an alternative, transport security using the “transport binding” may also be used for token requests using this authentication method.

To enable the service requester to obtain the security token of the IP/STS for securing messages, the endpoint reference of the IP/STS in the information card or in the WSDL retrieved must include its X.509v3 certificate in the `wsid:Identity` tag as defined in [Addressing-Ext].

The authentication token assertion in security policy that should be used inside the WSDL of the IP/STS, as described in Section 5.1.1, is shown below. This token assertion can be used regardless of whether transport binding (Section 5.1.1.1) or symmetric binding (Section 5.1.1.2) is used. The user’s self-issued token is submitted as an endorsing supporting token in the RST request

Authentication token assertion in security policy:

```
<sp:EndorsingSupportingTokens>
  <wsp:Policy>
    <sp:IssuedToken sp:IncludeToken="http://schemas.xmlsoap.org/ws/
2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
      <sp:Issuer>
```

```
        <wsa:Address>
          http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self
        </wsa:Address>
      </sp:Issuer>
      <sp:RequestSecurityTokenTemplate>
        <wst:TokenType>
          urn:oasis:names:tc:SAML:1.0:assertion
        </wst:TokenType>
        <wst:KeyType>
          http://schemas.xmlsoap.org/ws/2005/02/trust/PublicKey
        </wst:KeyType>
        <wst:Claims>
          <ic:ClaimType
Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalide
ntifier"/>
        </wst:Claims>
      </sp:RequestSecurityTokenTemplate>
      <wsp:Policy>
        <sp:RequireInternalReference/>
      </wsp:Policy>
    </sp:IssuedToken>
  </wsp:Policy>
</sp:EndorsingSupportingTokens>
```

5.5.3. Message Exchange

This section provides the SOAP message exchanges when message security with “symmetric binding” is used by the IP/STS. For this security binding, message protection and security correlation for the request and response legs of the message exchange is provided by an ephemeral symmetric session key. Message integrity and confidentiality is governed by the policy attached to individual messages as described in Section 5.1.

The following SOAP messages show the request/response exchange when message security is used (see policy for message security specified in the previous section). The exchange when transport security is used is shown in an earlier section for the username/password credential type.

This exchange also shows the use of the `wst:UseKey` element to request a token with an asymmetric proof key.

Token request from service requester to IP/STS:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue
    </wsa:Action>
    <wsa:MessageID wsu:Id="_2">
      urn:uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:MessageID>
    <wsa:To wsu:Id="_3">http://contoso.com/sts</wsa:To>
    <wsa:ReplyTo wsu:Id="_4">
      <wsa:Address>
        http://www.w3.org/2005/08/addressing/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
```

```
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>
      <!-- Symmetric session key encrypted to the X.509 certificate
           of the IP/STS endpoint  -->
      <xenc:EncryptedKey Id="_30">
        <xenc:EncryptionMethod
            Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
          <ds:DigestMethod Algorithm="http://.../2000/09/xmldsig#sha1"/>
        </xenc:EncryptionMethod>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier
              ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-
message-security-1.1#ThumbprintSHA1"
              EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-soap-message-security-1.0#Base64Binary">
              +PYbznDaB/dlhjIfqCQ458E72wA=
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>...</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedKey>
      <!-- List of encrypted elements in the message per the message
           confidentiality policy -->
      <xenc:ReferenceList>
        <xenc:DataReference URI="#_20" />
      </xenc:ReferenceList>
      <!-- Self-issued SAML token of the user encrypted to the IP/STS
           as the endorsing token -->
      <!-- Start encrypted Content (self-issued SAML token)
      <saml:Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" ...
           AssertionID="uuid:17e2007e-f959-4624-85ef-ae00df6fe071" ...>
        ...
      </saml:Assertion>
      End encrypted content -->
      <xenc:EncryptedData>
        ...
        <xenc:CipherData>
          <xenc:CipherValue>...</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedData>
      <!-- Primary message signature using the symmetric session key  -->
      <ds:Signature wsu:Id=”_40”>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
          <ds:Reference URI="#_6">...</ds:Reference>
          <ds:Reference URI="#_1">...</ds:Reference>
          <ds:Reference URI="#_2">...</ds:Reference>
          <ds:Reference URI="#_3">...</ds:Reference>
          <ds:Reference URI="#_4">...</ds:Reference>
          <ds:Reference URI="#_10">...</ds:Reference>
```

```
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#_30"
              ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-
message-security-1.1#EncryptedKey" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
      <!-- Endorsing signature using the user's self-issued SAML token
           endorsing the primary message signature  -->
      <ds:Signature wsu:Id="_43">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <ds:Reference URI="#_40">...</ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/
oasis-wss-saml-token-profile-1.0#SAMLAssertionID">
              uuid:17e2007e-f959-4624-85ef-ae00df6fe071
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
      <!-- Endorsing signature proving possession of the private key
           corresponding to the public key requested as proof key;
           KeyInfo within the signature contains the public key to be
           used as proof key in the issued token -->
      <ds:Signature wsu:Id="_46">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <ds:Reference URI="#_40">...</ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <ds:KeyValue>
            <ds:RSAKeyValue>
              <ds:Modulus>...</ds:Modulus>
              <ds:Exponent>...</ds:Exponent>
            </ds:RSAKeyValue>
          </ds:KeyValue>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>

  <S:Body wsu:Id="_10">
    <!-- Start encrypted Content
```

```
    <wst:RequestSecurityToken>
      <wst:TokenType>
        urn:oasis:names:tc:SAML:1.0:assertion
      </wst:TokenType>
      <wst:RequestType>
          http://schemas.xmlsoap.org/ws/2005/02/trust/Issue
      </wst:RequestType>
      <wst:KeyType>
        http://schemas.xmlsoap.org/ws/2005/02/trust/PublicKey
      </wst:KeyType>
      <ic:InformationCardReference>
        <ic:CardId> http://contoso.com/id/d795621fa01d454285f9</ic:CardId>
      </ic:InformationCardReference>
      <wst:Claims
          wst:Dialect=”http://schemas.xmlsoap.org/ws/2005/05/identity”>
        <ic:ClaimType Uri=”http://.../identity/claims/givenname”/>
        <ic:ClaimType Uri=”http://.../identity/claims/surname”/>
      </wst:Claims>
      <ic:ClientPseudonym>
        <ic:PPID>NHbuoB4KVKuvUx7b8szaux+bM8Rr0rPTPOXQlQTEBAo=</ic:PPID>
      </ic:ClientPseudonym>
      <wst:UseKey Sig="#_46">
        <ds:KeyInfo>
          <ds:KeyValue>
            <ds:RSAKeyValue>
              <ds:Modulus>...</ds:Modulus>
              <ds:Exponent>...</ds:Exponent>
            </ds:RSAKeyValue>
          </ds:KeyValue>
        </ds:KeyInfo>
      </wst:UseKey>
      <ic:RequestDisplayToken xml:lang="en-us" />
    </wst:RequestSecurityToken>
    End encrypted content -->
    <xenc:EncryptedData Id="_20">
      <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#_30"
            ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-
security-1.1#EncryptedKey" />
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>Thp8EqU0S+A4Qu+==</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

Note the following in the request message:

- The ordering of items in the security header follows the strict layout as prescribed by [WS-SecurityPolicy].
- An asymmetric proof key is requested, and the public key to be used as proof key is included in the wst:UseKey element in the SOAP body as a raw RSA key. Further,

proof-of-possession of the corresponding private key is included via a signature in the SOAP security header (see signature element with wsu:Id="_46"). The signature also includes the same RSA key that is in the wst:UseKey element in the SOAP body. The IP/STS should verify that the RSA key included in the wst:UseKey element and in the proof-of-possession signature are the same before accepting it.

- The information card reference (CardId) is included.
- Information about the relying party is not included (i.e. there is no wsp:AppliesTo element).
- A client generated PPID seed is included in the ic:PPID element for the IP/STS to use in generating any pairwise identifiers.
- A display token localized in "US English" is requested.
- The X.509 certificate of the IP/STS is NOT included in the message. References to it are made indirectly using a SHA1 thumbprint based key identifier reference using the wsse:KeyIdentifier element since the sp:ProtectionToken assertion in the STS policy includes the sp:RequireThumbprintReference policy assertion.
- An ephemeral symmetric session key is generated and encrypted to the X.509 certificate of the IP/STS endpoint. References to the encrypted session key included in the SOAP security header are made using the wsse:Reference based direct references.
- The message is signed with this symmetric session key which constitutes the primary message signature. Encrypted message elements are encrypted with the symmetric session key as well.
- The primary message signature is further signed by the key in the user's self-issued SAML token (endorsing signature) which is used to authenticate the user. The SAML token itself is NOT covered by the message signature or the endorsing signature.
- The self-issued SAML token is included in its entirety in the SOAP security header. References to the self-issued SAML token included in the message are made using the assertion ID using the wsse:KeyIdentifier element.

Token response from IP/STS to service requester:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue
    </wsa:Action>
    <wsa:RelatesTo wsu:Id="_2">
      urn:uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:RelatesTo>
    <wsa:To wsu:Id="_3">
      http://www.w3.org/2005/08/addressing/anonymous
    </wsa:To>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>
      <!-- List of encrypted elements in the message per
           message confidentiality policy -->
      <xenc:ReferenceList>
```

```
        <xenc:DataReference URI="#_20" />
      </xenc:ReferenceList>
      <!-- Message signature using the symmetric session key -->
      <ds:Signature wsu:Id=”_33”>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
          <ds:Reference URI="#_6">...</ds:Reference>
          <ds:Reference URI="#_1">...</ds:Reference>
          <ds:Reference URI="#_2">...</ds:Reference>
          <ds:Reference URI="#_3">...</ds:Reference>
          <ds:Reference URI="#_10">...</ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier
              ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-
message-security-1.1#EncryptedKeySHA1">
                AcLJ9234LIl2HbBwbpk0qBPhVZ8=
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>

  <S:Body wsu:Id=”_10”>
    <!-- Start encrypted Content
    <wst:RequestSecurityTokenResponse>
      <wst:TokenType>
        urn:oasis:names:tc:SAML:1.0:assertion
      </wst:TokenType>
      <wst:Lifetime>
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wst:Lifetime>
      <wst:RequestedSecurityToken>
        <saml:Assertion xmlns="urn:oasis:names:tc:SAML:1.1:assertion"
          AssertionID=”uuid:17e2007e-f959-4624-85ef-ae00df6fe071” ...>
          ...
        </saml:Assertion>
      </wst:RequestedSecurityToken>
      <wst:RequestedAttachedReference>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/
oasis-wss-saml-token-profile-1.0#SAMLAssertionID">
             uuid:17e2007e-f959-4624-85ef-ae00df6fe071
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </wst:RequestedAttachedReference>
      <wst:RequestedUnattachedReference>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/
oasis-wss-saml-token-profile-1.0#SAMLAssertionID">
```

```
            uuid:17e2007e-f959-4624-85ef-ae00df6fe071
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </wst:RequestedUnattachedReference>
      <wst:KeyType>
        http://schemas.xmlsoap.org/ws/2005/02/trust/PublicKey
      </wst:KeyType>
      <wst:KeySize>2048</wst:KeySize>
      <ic:RequestedDisplayToken>
       <ic:DisplayToken xml:lang="en-us">
        <ic:DisplayClaim Uri="http://.../identity/claims/givenname">
          <ic:DisplayTag>Given Name</ic:DisplayTag>
          <ic:DisplayValue>John</ic:DisplayValue>
        </ic:DisplayClaim>
        <ic:DisplayClaim Uri="http://.../identity/claims/surname">
          <ic:DisplayTag>Last Name</ic:DisplayTag>
          <ic:DisplayValue>Doe</ic:DisplayValue>
        </ic:DisplayClaim>
       </ic:DisplayToken>
      </ic:RequestedDisplayToken>
    </wst:RequestSecurityTokenResponse>
    End encrypted content -->
    <xenc:EncryptedData Id="_20">
      <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier
            ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-
security-1.1#EncryptedKeySHA1">
              AcLJ9234LIl2HbBwbpk0qBPhVZ8=
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>...</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

Note the following in the response message:

- The ordering of items in the security header follows the strict layout as prescribed by [WS-SecurityPolicy].
- The message is signed with the symmetric session key that was included in the RST request. Encrypted message elements are encrypted with the same symmetric session key.
- References to the symmetric session key, which is NOT included in the message, are made indirectly using a SHA1 thumbprint based key identifier reference using the `wsse:KeyIdentifier` element.
- The issued security token is NOT encrypted to the relying party since information about the relying party was not conveyed in the request.

- Since the SAML token doesn't support references using URI fragments (XML Id), attached and unattached references are returned whose element content can be used *verbatim* within a `wsse:SecurityTokenReference` element to reference the token when it is placed inside a message.
- Since an asymmetric proof key was requested and an ephemeral public key was supplied as the proof key in the token request, the response message does not include an explicit proof token.
- A display token containing textual representation of the actual token is returned.

6. Faults

In addition to the standard faults described in WS-Addressing, WS-Security and WS-Trust, the Information Card Technical Reference [CardSpace-Ref] defines additional faults that may be generated by the relying party or the identity provider.

7. References

[CardSpace-Ref]
"A Technical Reference for Information Cards in Windows CardSpace v1.0", December 2006.

[CardSpace-Browser]
"A Guide to Supporting Information Cards within Web Applications and Browsers as of Windows CardSpace v1.0", December 2006.

[SOAP 1.2]
M. Gudgin, et al, "SOAP Version 1.2 Part 1: Messaging Framework", June 2003.

[WS-Addressing]
M. Gudgin et al, "Web Services Addressing 1.0 – Core", August 2005.

[Addressing-Ext]
Document to be published in the near future.

[WS-MetadataExchange]
"Web Services Metadata Exchange (WS-MetadataExchange), Version 1.1" August 2006.

[WS-Security]
A. Natalin et al, "Web Services Security: SOAP Message Security 1.0", May 2004.

[WS-Policy]
"Web Services Policy Framework (WS-Policy), Version 1.2", March 2006.

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

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

[XMLDSIG]
Eastlake III, D., Reagle, J., and Solo, D., "XML-Signature Syntax and Processing", March 2002.

[XMLENC]
Imamura, T., Dillaway, B., and Simon, E., "XML Encryption Syntax and Processing", August 2002.

[XML Schema, Part 1]
H. Thompson et al, "XML Schema Part 1: Structures", May 2001.

[XML Schema, Part 2]
P. Biron et al, "XML Schema Part 2: Datatypes", May 2001.

Appendix A – Self-Issued Tokens

The Windows CardSpace system includes a simple identity provider called the “Self-issued Identity Provider” (see Figure 1) which allows users to self-assert identity in the form of self-issued tokens. These tokens may be acceptable, for example, when accessing a retail bookseller Web service to set up an account. The retail service may allow the user to self-assert her own name and address information.

This section describes how a relying party that accepts self-issued tokens can authenticate and use them. Note that an identity provider can also be the relying party for self-issued tokens if it accepts a self-issued token as the credential to authenticate a user. This is described in more detail in Section .

Self-issued Token Characteristics

The characteristics of a self-issued token, including its format, the encryption structure, and the supported claim types are defined in [CardSpace-Ref].

Although a self-issued token is always encrypted to the relying party, following is an example of a decrypted self-issued security token containing three claims (or attributes) with an asymmetric proof key.

Example:

```
<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
    AssertionID="uuid:08301dba-d8d5-462f-85db-dec08c5e4e17"
    Issuer="http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self"
    IssueInstant="2004-10-06T16:44:20.00Z"
    MajorVersion="1" MinorVersion="1">
  <Conditions NotBefore="2004-10-06T16:44:20.00Z"
      NotOnOrAfter="2004-10-06T16:49:20.00Z">
    <AudienceRestrictionCondition>
      <Audience>http://www.relying-party.com</Audience>
    </AudienceRestrictionCondition>
  </Conditions>
  <AttributeStatement>
    <Subject>
      <SubjectConfirmation>
        <ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
        </ConfirmationMethod>
        <ds:KeyInfo>
          <!-- The proof key goes here. The content of this element
               is either a symmetric key or a RSA public key depending
               on what is required by the relying party -->
          <!-- Proof key: an asymmetric RSA public key -->
          <KeyValue>
           <RSAKeyValue>
             <Modulus>...</Modulus>
             <Exponent>AQAB</Exponent>
           </RSAKeyValue>
          </KeyValue>
        </ds:KeyInfo>
      </SubjectConfirmation>
    </Subject>
    <Attribute AttributeName=”privatepersonalidentifier”
      AttributeNamespace=”http://.../ws/2005/05/identity/claims”>
```

```
      <AttributeValue>q65Thp8EqU0S+A4Qu+==</AttributeValue>
    </Attribute>
    <Attribute AttributeName=”givenname”
      AttributeNamespace=”http://.../ws/2005/05/identity/claims”>
      <AttributeValue>dasf</AttributeValue>
    </Attribute>
    <Attribute AttributeName=”emailaddress”
      AttributeNamespace=”http://.../ws/2005/05/identity/claims”>
      <AttributeValue>dasf@mail.com</AttributeValue>
    </Attribute>
  </AttributeStatement>
  <Signature xmlns=”http://www.w3.org/2000/09/xmldsig#”>
    <SignedInfo>
      <CanonicalizationMethod
        Algorithm=”http://www.w3.org/2001/10/xml-exc-c14n#”/>
      <SignatureMethod
        Algorithm=”http://www.w3.org/2000/09/xmldsig#rsa-sha1”/>
      <Reference URI=”uuid:08301dba-d8d5-462f-85db-dec08c5e4e17”>
        <Transforms>
          <Transform
            Algorithm=” http://.../2000/09/xmldsig#enveloped-signature”/>
          <Transform
            Algorithm=”http://www.w3.org/2001/10/xml-exc-c14n#”/>
        </Transforms>
        <DigestMethod
          Algorithm=”http://www.w3.org/2000/09/xmldsig#sha1”/>
        <DigestValue>vpnIyEi4R/S4b+1vEH4gwQ9iHsY=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>...</SignatureValue>
    <KeyInfo>
      <!-- Token signing key: an asymmetric RSA public key -->
      <KeyValue>
        <RSAKeyValue>
          <Modulus>...</Modulus>
          <Exponent>AQAB</Exponent>
        </RSAKeyValue>
      </KeyValue>
    </KeyInfo>
  </Signature>
</Assertion>
```

Note the following in the self-issued token shown above:

- The issuer of the token, indicated by the value of the `saml:Issuer` attribute, is specified as the URI *http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self*.
- The token is issued specifically for use at the relying party (i.e., target scope) by using the `saml:AudienceRestrictionCondition` element.
- The subject confirmation key (or proof key) in the issued token is a RSA public key within the `saml:SubjectConfirmation` element.
- The token signing key in the issued token is a RSA public key within the `ds:Signature` element.

Accepting Self-issued Tokens

A relying party can accept self-issued tokens from users where it is convenient and appropriate. As described earlier, an identity provider can also accept self-issued tokens as authentication credential. When accepting and processing self-issued tokens, one should adhere to the following guidelines.

- The token signing key (a RSA public key) in the self-issued token should be used as the long-term trusted key associated with the user. This key is used to authenticate the user whenever a self-issued token is presented. The proof key in the token may change from one instance of a self-issued token to another from the same user (e.g., two tokens issued at different times).
- The signature of the self-issued token should always be verified.
- When accepting and verifying a self-issued token, ensure that the current time falls within the token's validity interval; otherwise reject the token.
- If an audience restriction condition is included in the self-issued token specifying a target scope, then ensure that the relying party is covered by that scope. Otherwise, reject the token.
- If a subject confirmation key is specified in the self-issued token, it should be treated as a short-term key to demonstrate proof-of-possession of the token. The service requester must be required to provide some proof of its knowledge of the subject confirmation key.
- If a long-term unique identifier for the user is needed (for example, to anchor profile information for the user), then the "Private Personal Identifier" claim should be used and specified as a required claim in the token policy (see description of that claim in [CardSpace-Ref]). This claim provides a privacy friendly identifier for the user that is the subject of the security token.

Appendix B – Glossary

Authentication

Authentication is the process of validating security credentials.

Claim

A claim is a statement made about an entity such as a sender, a service, or other resource (e.g., name, identifier, key, group, privilege, capability, etc.). It is sometimes referred to as an *assertion* in the security literature.

Claims Authority

A claims authority is an entity that can authenticate principals and make specific claims about them which other services may trust. For example, an authority may assert a user's name, address and social security number as claims which another service may trust and accept. A claims authority is typically a security token service.

Confidentiality

Confidentiality is the process by which data is protected such that only authorized actors or security token owners can view the data.

Digest

A digest is a cryptographic checksum of an octet stream.

Digital Identity

A set of claims asserted by a claims authority about a subject. Claims are typically conveyed in signed security tokens.

Integrity

Integrity is the process by which it is guaranteed that information is not modified in transit.

Principal

See *subject*.

Proof-of-Possession

The proof-of-possession information is data that is used to demonstrate the sender's knowledge of information that should only be known to the claiming sender of a security token.

Security Binding

A set of properties that together provide enough information to secure a given message exchange.

Security token

A security token represents a collection of one or more claims.

Security Token Service

A Security Token Service (STS) is a Web service that issues security tokens – that is, the service makes assertions – based on evidence that it trusts, to those services that trust the service or to specific recipients.

Signature

A signature is a cryptographic binding of a proof-of-possession and a digest. This covers both symmetric key-based and public key-based signatures.

Signed security token

A signed security token is a security token that is cryptographically endorsed by a specific authority (e.g., an X.509 certificate, a Kerberos ticket or a SAML assertion).

Subject

A subject is any entity about which claims can be made by an identity provider. These entities include users, services, computers and devices.