

A Technical Reference for InfoCard v1.0 in Windows

August, 2005

Authors

Microsoft Corporation

Copyright Notice

(c) 2005 [Microsoft Corporation](#). All rights reserved.

Abstract

The InfoCard system in the Windows Communications Foundation (WCF) of WinFX allows users to manage their digital identities from various identity providers, and employ them in different contexts where they are accepted to access online services. This document provides a technical reference for the mechanisms implemented in the Windows InfoCard system and the schema employed by that implementation. This document is intended to be read alongside the InfoCard Guide [[InfoCard-Guide](#)] which provides a non-normative description of the overall InfoCard model to allow digital identity to be integrated into a user-centric identity framework that promotes interoperability between identity providers and relying parties with the user in control.

Status

This draft of the InfoCard Technical Reference reflects what is implemented by the InfoCard system in WCF in the Beta2 release of WinFX. The documented behavior and schema described here are subject to change in the final release of the product.

Table of Contents

1. Introduction

2. Terminology and Notation

- 2.1. XML Namespaces
- 2.2. Notational Conventions

3. Simple Identity Provider and Self-Issued Tokens

4. Relying Party Interactions

- 4.1. Expressing Token Requirements of Relying Party
 - 4.1.1. Issuer of issued tokens
 - 4.1.2. Type of proof key in issued tokens
 - 4.1.3. Freshness of issued tokens
 - 4.1.4. Claims in issued tokens

5. Identity Provider Interactions

- 5.1. Defining InfoCard
 - 5.1.1. XML schema
 - 5.1.2. Obtaining InfoCards
- 5.2. Token Issuance Policy of Identity Provider
 - 5.2.1. Supported token types
 - 5.2.2. Supported claim types
 - 5.2.3. Require relying party identity
- 5.3. Security Token Request to Identity Provider
 - 5.3.1. Supplying InfoCard reference
 - 5.3.2. Identifying relying party to identity provider
 - 5.3.3. Requesting display token
 - 5.3.4. Requesting proof key
- 5.4. Security Token Response from Identity Provider
 - 5.4.1. Returning display token
 - 5.4.2. Returning proof key

6. Authenticating to Identity Provider

- 6.1. Using Username and Password
- 6.2. Using KerberosV5 Service Ticket
- 6.3. Using Software Based X509 Certificate
- 6.4. Using Smartcard Based X509 Certificate
- 6.5. Using Self-issued Token

7. Faults

- 7.1. Relying Party
- 7.2. Identity Provider

8. References

Appendix A – Claims Defined by InfoCard Profile

- A.1. First Name
- A.2. Last Name
- A.3. Email Address

- A.4. Street Address
- A.5. Locality Name or City
- A.6. State or Province
- A.7. Postal Code
- A.8. Country
- A.9. Primary or Home Telephone Number
- A.10. Secondary or Work Telephone Number
- A.11. Mobile Telephone Number
- A.12. Date of Birth
- A.13. Gender
- A.14. Private Personal Identifier

Appendix B – XSD

1. Introduction

The InfoCard model described in [[InfoCard-Guide](#)] builds on the mechanisms described in [[WS-Trust](#)], [[WS-SecurityPolicy](#)] and [[WS-MetadataExchange](#)] to allow digital identity to be integrated into a token issuance and consumption framework that promotes interoperability between identity providers and relying parties with the user in control. In this document, details are provided on the implementation specifics and the schema for new elements/extensions used for interactions between the InfoCard system on a service requester (client) and a relying party or an identity provider. It is useful to refer to the overall model described in the InfoCard Guide [[InfoCard-Guide](#)] when reading this technical reference. The term IP/STS used in this document refers to the security token service run by an identity provider.

2. Terminology and Notation

2.1. XML Namespaces

The base XML namespace URI that is used by the schema and mechanisms described in this technical reference is:

```
http://schemas.microsoft.com/ws/2005/05/identity
```

The schema for new elements or extensions defined in this document can be found at:

```
http://schemas.microsoft.com/ws/2005/05/identity/InfoCard.xsd
```

Table 1 lists the XML namespaces that are used in this document. The current SOAP 1.2 namespace URI is used to provide detailed examples, not to limit the applicability of this profile to a single version of SOAP.

Table 1: Prefixes and XML namespaces used in this document

Prefix	XML Namespace	Specification(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.microsoft.com/ws/2005/05/identity	This document
wsid	http://schemas.microsoft.com/windows/wcf/2005/09/addressingidentityextension	Identity Extension for Web Services Addressing [Addressing-Ext]
wsx	http://schemas.xmlsoap.org/ws/2004/08/mex	WS-MetadataExchange [WS-MetadataExchange]
wsa	http://schemas.xmlsoap.org/ws/2004/08/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-1.1.xsd	WS-Security Extensions [WS-Security]
wst	http://schemas.xmlsoap.org/ws/2004/04/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]

2.2. Notational Conventions

This technical reference uses the following syntax to describe outlines for messages and XML fragments:

- The syntax appears as an XML instance, but values in italics indicate data types instead of values.
- Characters are appended to elements and attributes to indicate cardinality:
 - "?" (0 or 1)
 - "*" (0 or more)
 - "+" (1 or more)
- The character "|" is used to indicate a choice between alternatives.
- The characters "(" and ")" are used to indicate that contained items are to be treated as a group with respect to cardinality or choice.
- The characters "[" and "]" are used to call out references and property names.
- An ellipsis (i.e. "...") indicates a point of extensibility that allows other child or attribute content. Additional children or attributes can be added at the indicated extension points. The InfoCard system in Windows will ignore any extensions it does not recognize.
- XML namespace prefixes (see Table 1) are used to indicate the namespace of the element being defined.

3. Simple Identity Provider and Self-Issued Tokens

The self-issued tokens issued by the simple identity provider in the InfoCard system have the following characteristics:

- The token type of the issued token is SAML 1.1 and is identified by either of the token type URIs “urn:oasis:names:tc:SAML:1.0:assertion” or “http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1”. Note that these are the token types supported in the initial release, but it is expected that the support will be extended for other token types (e.g., SAML 2.0) in future releases.
- The signature key used in the issued token is a 2048-bit asymmetric RSA key which identifies the issuer.
- The `saml:NameIdentifier` element is *not* used to specify the token subject.
- The subject confirmation method is always specified as “holder of key” and is identified by the URI “urn:oasis:names:tc:SAML:1.0:cm:holder-of-key”. The subject confirmation key (also referred to as the *proof key*) included in the issued token is either a 128-bit symmetric key or a 1024-bit RSA public key depending on what the relying party needs. The `ds:KeyInfo` child element is always present under the `saml:SubjectConfirmation` element in the token and contains the proof key as follows:
 - For symmetric proof keys, the symmetric key value is encrypted to the intended recipient of the token in the form of a `xenc:EncryptedKey` child element.
 - For asymmetric proof keys, the public RSA key value is present in the token in the form of a `ds:RSAKeyValue` child element under `ds:KeyValue` element.
- The issued token always contains a single `saml:AttributeStatement` element containing both the subject confirmation information as well as the required claims (called *attributes* in a SAML token). The attribute claim types supported in the self-issued token are those listed in [Appendix A](#).
- The attributes asserted in the `saml:AttributeStatement` element of the issued token are named using the attribute type definitions in the XML schema for this document. For each attribute represented by a `saml:Attribute` element, the `AttributeName` XML attribute is set as the NCname of the corresponding attribute type defined in the schema and the `AttributeNamespace` XML attribute is set to the namespace URI `http://schemas.microsoft.com/ws/2005/05/identity` for the claim types defined in this document.
- The issued token always contains the `saml:Conditions` element to indicate the token validity interval using the `NotBefore` and `NotOnOrAfter` attributes.

The XML signature [[XMLDSIG](#)] profile that is used for the digital signature in a self-issued token is as follows:

- Uses *enveloped signature* format for the token signature. The signature contains a single `ds:Reference` containing a URI reference to the `AssertionID` attribute value of the root element of the SAML token.
- Uses RSA signing with the algorithm identifier given by the URI `http://www.w3.org/2000/09/xmlsig#rsa-sha1`.
- Uses exclusive canonicalization with the algorithm identifier given by the URI `http://www.w3.org/2001/10/xml-exc-c14n#`.
- Uses SHA1 digest method for the data elements being signed with the algorithm identifier `http://www.w3.org/2000/09/xmlsig#sha1`.

- No other transforms, other than the ones listed above, are used in the enveloped token signature.
- The `ds:KeyInfo` element is always present in the signature carrying the signing RSA key value in the form of a `ds:RSAKeyValue` child element.

Following is an example of a self-issued signed security token containing three attributes (or claims). For convenience, the symmetric and asymmetric proof key cases are both shown together in the same token as alternatives although only one would actually be present.

Example:

```
<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
  AssertionID="uuid-08301dba-d8d5-462f-85db-dec08c5e4e17"
  Issuer="http://schemas.microsoft.com/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"/>
<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 specified by the relying party -->
        <!-- symmetric proof key encrypted to recipient -->
        <xenc:EncryptedKey>
          <xenc:EncryptionMethod
            Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
          <ds:KeyInfo>
            <ds:X509Data>
              <wsse:KeyIdentifier
                ValueType="http://docs.oasis-open.org/wss/2004/xx/oasis-
2004xx-wss-soap-message-security-1.1#ThumbprintSHA1">
                EdFoIaAeja85201XTzjNMVWy7532jUYtrx=
              </wsse:KeyIdentifier>
            </ds:X509Data>
          </ds:KeyInfo>
          <xenc:CipherData>
            <xenc:CipherValue>
              AuFhiu72+1kaJiAuFhiu72+1kaJi=
            </xenc:CipherValue>
          </xenc:CipherData>
        </xenc:EncryptedKey>
        <!-- OR -->
        <!-- asymmetric RSA public key as proof key -->
        <KeyValue>
          <RSAKeyValue>
            <Modulus>...</Modulus>
            <Exponent>AQAB</Exponent>
          </RSAKeyValue>
```

```

</KeyValue>
  </ds:KeyInfo>
  </SubjectConfirmation>
</Subject>
<Attribute AttributeName="privatpersonalidentifier"
  AttributeNamespace="http://schemas.microsoft.com/ws/2005/05/identity">
  <AttributeValue>
    f8301dba-d8d5a904-462f0027-85dbdec0
  </AttributeValue>
</Attribute>
<Attribute AttributeName="givenname"
  AttributeNamespace="http://schemas.microsoft.com/ws/2005/05/identity">
  <AttributeValue>dasf</AttributeValue>
</Attribute>
<Attribute AttributeName="emailaddress"
  AttributeNamespace="http://schemas.microsoft.com/ws/2005/05/identity">
  <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+lvEH4gwQ9iHsY=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>
    <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.microsoft.com/ws/2005/05/identity/issuer/self` defined later in Section 4.1.1 representing the issuer "self".
- The subject confirmation key (or proof key) included in the issued token is specified as a base64 encoded value that is either an encrypted symmetric key in the form of a

xenc:EncryptedKey element OR a RSA public key in the form of a ds:KeyValue element within the saml:SubjectConfirmation element (see the first block of highlighted and boxed text in the token above).

- The signature key included in the issued token is specified as a base64 encoded RSA key in the form of a ds:RSAKeyValue element as a child of the ds:KeyInfo element in the token signature (see the second block of highlighted text in the token above).

For stronger guarantee that the self-issued security token issued by the InfoCard system can only be seen and evaluated by the relying party that the user approved for release, the entire token is encrypted to the relying party. The encryption construct involves encrypting the SAML token with a randomly generated symmetric key which in turn is encrypted to the relying party's public key. The encrypted symmetric key is placed in an xenc:EncryptedKey element immediately followed by an xenc:EncryptedData element carrying the encrypted security token.

The XML encryption [XMLENC] profile that is used for encrypted key and encrypted self-issued token content is as follows:

- For encrypting raw keys, it uses the RSA-OAEP algorithm with the encryption method identifier <http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p>.
- For encrypting the token content, it uses the AES CBC algorithm with the encryption method identifier <http://www.w3.org/2001/04/xmlenc#aes128-cbc>.
- The ds:KeyInfo element is always present in the encrypted key or data element carrying the encryption key information in the form of a security token reference.

Following is an illustration of the self-issued token shown in the earlier example encrypted to the public key of a relying party taken from an X509v3 certificate.

Example:

```
<!-- Encryption construct uses a symmetric key to encrypt the SAML token
and encrypts the symmetric key to the relying party's public key -->
<!-- Start encrypted Content
101110010011101001111010100011010101
End encrypted content -->
<xenc:EncryptedKey>
  <xenc:EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p" />
  <ds:KeyInfo>
    <ds:X509Data>
      <wsse:KeyIdentifier
        ValueType="http://docs.oasis-open.org/wss/2004/xx/oasis-2004xx-wss-
soap-message-security-1.1#ThumbprintSHA1">
        EdFoIaAeja85201XTzjNMVWy7532jUYtrx=
      </wsse:KeyIdentifier>
    </ds:X509Data>
  </ds:KeyInfo>
  <xenc:CipherData>
    <xenc:CipherValue>Ukasdj8257Fjwf=</xenc:CipherValue>
  </xenc:CipherData>
</xenc:EncryptedKey>

<!-- Start encrypted Content
<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
  AssertionID="uuid-08301dba-d8d5-462f-85db-dec08c5e4e17" ...>
...
```

```

</Assertion>
End encrypted content -->
<xenc:EncryptedData>
  <xenc:EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
  <ds:KeyInfo>
    <wsse:SecurityTokenReference>
      <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/2004/xx/
oasis-2004xx-wss-soap-message-security-1.1#EncryptedKey">
        SdFoIaAeja8520=
      </wsse:KeyIdentifier>
    </wsse:SecurityTokenReference>
  </ds:KeyInfo>
  <xenc:CipherData>
    <xenc:CipherValue>aKlh4817JerpZoDofy90=</xenc:CipherValue>
  </xenc:CipherData>
</xenc:EncryptedData>

```

Note the following in the encryption construct for the self-issued token shown above:

- The symmetric key used to encrypt the self-issued token is in turn encrypted to the public key of the relying party from an X509v3 certificate which is referenced by a `ds:X509Data` element (see the first block of bold text in the example above).
- There is no `xenc:ReferenceList` element present within the `xenc:EncryptedKey` element providing a manifest of the data encrypted by the encrypted key.
- The encrypted data containing the self-issued token references the encrypted key as its encryption token using the `wsse:KeyIdentifier` child element of the `wsse:SecurityTokenReference` element (see the second block of bold text in the example above) as defined in [\[WS-Security\]](#).

4. Relying Party Interactions

This section defines the constructs used by a relying party Web service for specifying and conveying its security token requirements to the service requester.

4.1. Expressing Token Requirements of Relying Party

A relying party specifies its security token requirements as part of its security policy using the primitives and assertions defined 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.

Syntax:

```

<sp:IssuedToken sp:Usage="xs:anyURI" sp:IncludeToken="xs:anyURI" ...>
  <sp:Issuer>
    <wsa:EndpointReference>...</wsa:EndpointReference> | xs:any
  </sp:Issuer>
  <sp:RequestSecurityTokenTemplate>
    ...
  </sp:RequestSecurityTokenTemplate>
  <wsp:Policy>
    ...
  </wsp:Policy>
  ...
</sp:IssuedToken>

```

The attributes and elements listed in the schema fragment above are described in [[WS-SecurityPolicy](#)].

Note: The issued token may have a symmetric, an asymmetric or no key material to use as proof. The binding assertion will imply the type of proof key associated with this token. Relying parties may also include information in the `sp:RequestSecurityTokenTemplate` element to explicitly specify the expected proof key type.

The ensuing subsections describe special parameters and policy assertion elements added by the InfoCard model as extensions to the `sp:IssuedToken` policy assertion that convey additional instructions to the service requester.

4.1.1. Issuer of issued tokens

The optional `sp:IssuedToken/sp:Issuer` element contains an endpoint reference for the issuer for the required token. A relying party can indicate a specific issuer for the required token. It can also omit the `sp:Issuer` element or specify it as an empty element to indicate that the service requester should determine the appropriate issuer for the required token with help from the user if necessary.

In order to specify that the required token can be a self-issued security token, the special URI below should be specified as the value of the `wsa:Address` element within the endpoint reference for the issuer.

URI:

```
http://schemas.microsoft.com/ws/2005/05/identity/issuer/self
```

Following is an example of using this URI within an issued token policy.

Example:

```
<sp:IssuedToken ...>
  <sp:Issuer>
    <wsa:EndpointReference>
      <wsa:Address>
        http://schemas.microsoft.com/ws/2005/05/identity/issuer/self
      </wsa:Address>
    </wsa:EndpointReference>
  </sp:Issuer>
  ...
</sp:IssuedToken>
```

4.1.2. Type of proof key in issued tokens

By default, a required token is assumed to be a *symmetric key* token if no explicit key type is specified in the issued token policy of a relying party. It can specifically request the use of an *asymmetric key* in the issued token by using the `wst:KeyType` element within its issued token policy assertion as follows:

Syntax:

```
<sp:IssuedToken>
  <sp:RequestSecurityTokenTemplate>
    <wst:KeyType>
      http://schemas.xmlsoap.org/ws/2004/04/trust/PublicKey
    </wst:KeyType>
  </sp:RequestSecurityTokenTemplate>
</sp:IssuedToken>
```

4.1.3. Freshness of issued tokens

A relying party can limit the maximum token age of a security token that it is willing to accept by using the optional assertion below within an issued token policy assertion.

Syntax:

```
<ic:MaxTokenAge> xs:unsignedLong </ic:MaxTokenAge> ?
```

The following describes additional constraints on the outline listed above:

/ic:MaxTokenAge

The maximum elapsed time since the instant the token was created. It acts as an instruction to the service requester (InfoCard system) to not submit any tokens that are older than the age specified.

Following is an example of using this assertion within an issued token policy which indicates that the relying party will only accept a token if it is less than 5 minutes (30,000 milliseconds) old since the time it was created.

Example:

```
<sp:IssuedToken ...>
  ...
  <wsp:Policy>
    <ic:MaxTokenAge>30000</ic:MaxTokenAge>
  </wsp:Policy>
</sp:IssuedToken>
```

4.1.4. Claims in issued tokens

A relying party can specify its requirement for claims in issued tokens by using the optional parameter below within an issued token policy assertion. The outline for the parameter is:

Syntax:

```
<ic:Claim URI="xs:anyURI" Optional="xs:boolean"? /> *
```

The following describes additional constraints on the outline listed above:

/ic:Claim

Indicates the required claim.

/ic:Claim/@URI

The unique identifier of the required claim.

/ic:Claim/@Optional

Indicates if the claim can have a null value or be absent in the security token. By default, any specified claim is a mandatory claim and must not have a null value in the security token.

Two `<ic:Claim>` elements refer to the same claim if and only if the values of their XML attribute named `URI` are equal in a case-sensitive string comparison.

The `ic:Claim` element can appear inside the `wst:Claims` optional parameter element within an issued token policy assertion. The `wst:Dialect` attribute on the `wst:Claims` element should be qualified with the URI value below to indicate that the specified claim elements are to be processed using the InfoCard model.

```
wst:Dialect="http://schemas.microsoft.com/ws/2005/05/identity"
```

Following is an example of using this assertion within an issued token policy to require two claim types in the issued security token with one claim type being optional.

Example:

```
<sp:IssuedToken ...>
...
<sp:RequestSecurityTokenTemplate>
...
<wst:Claims
  wst:Dialect="http://schemas.microsoft.com/ws/2005/05/identity">
  <ic:Claim
    URI="http://.../ws/2005/05/identity/claims/givenname"/>
  <ic:Claim
    URI="http://.../ws/2005/05/identity/claims/surname"
    Optional="true" />
  </wst:Claims> </sp:RequestSecurityTokenTemplate>
...
</sp:IssuedToken>
```

The InfoCard model defines a standard set of claims for typical personal information about users that may be used by relying party Web services and identity providers in issued security tokens. The list of the standard claims defined by the InfoCard model and the corresponding URIs is given in [Appendix A](#).

5. Identity Provider Interactions

This section defines the constructs used for interacting with an identity provider.

5.1. Defining InfoCard

Users may obtain an artifact with a unique identifier from an identity provider, called an "InfoCard," using which users can visualize their digital relationship with the identity provider in user interfaces and request security tokens with claims from the identity provider.

5.1.1. XML schema

InfoCards are concretely represented as XML documents that can be issued by identity providers. The XML schema for an InfoCard is defined below:

Syntax:

```
<ic:InfoCard xml:lang="xs:language"? ...>
  <ic:InfoCardReference>
    <ic:CardId> xs:anyURI </ic:CardId>
    <ic:CardVersion> xs:unsignedInt </ic:CardVersion> ?
  </ic:InfoCardReference>
  <ic:CardName> xs:string </ic:CardName> ?
  <ic:CardImage MimeType="xs:string"> xs:base64Binary </ic:CardImage> ?
  <ic:IssuerName> xs:string </ic:IssuerName>
  <ic:TimeIssued> xs:dateTime </ic:TimeIssued>
  <ic:TimeExpires> xs:dateTime </ic:TimeExpires> ?
  <ic:TokenServiceReference>
    (<ic:TokenService>
      <wsa:EndpointReference ...> ... </wsa:EndpointReference>
      <ic:CredentialHint>xs:string</ic:CredentialHint> ?
    )
    <ic:UserNamePasswordAuthenticate>...</ic:UserNamePasswordAuthenticate> |
    <ic:KerberosV5Authenticate>...</ic:KerberosV5Authenticate> |
    <ic:X509V3Authenticate>...</ic:X509V3Authenticate> |
    <ic:SelfIssuedAuthenticate>...</ic:SelfIssuedAuthenticate>
```

```

    )
    </ic:TokenService>) +
</ic:TokenServiceReference>
<ic:InfoCardPolicy>
  <ic:SupportedTokenTypes>
    <ic:TokenType URI="xs:anyURI" /> +
  </ic:SupportedTokenTypes>
  <ic:SupportedClaims>
    (<ic:SupportedClaim URI="xs:anyURI">
      <ic:DisplayTag> xs:string </ic:DisplayTag> ?
      <ic:Description> xs:string </ic:Description> ?
    </ic:SupportedClaim>) +
  </ic:SupportedClaims>
  <ic:RequireAppliesTo /> ?
</ic:InfoCardPolicy>
...
</ic:InfoCard>

```

The following describes additional constraints on the outline listed above:

/ic:InfoCard

An InfoCard issued by an identity provider.

/ic:InfoCard/@xml:lang

An optional language identifier, using the language codes specified in [\[RFC 3066\]](#), in which the content of those InfoCard elements that can be localized have been localized.

/ic:InfoCard/ic:InfoCardReference

A specific reference for the InfoCard that should be used in future requests for security tokens from the identity provider based on that InfoCard (see Section 5.3.1).

/ic:InfoCard/ic:InfoCardReference/ic:CardId

This required element provides a globally unique identifier in the form of a URI for the specific InfoCard.

/ic:InfoCard/ic:InfoCardReference/ic:CardVersion

This optional element provides a versioning epoch for the InfoCard issuance infrastructure used by the identity provider. Note that it is possible to include version information in the CardId value as it is a URI which can have hierarchical content. However, it is specified as a separate value to allow the identity provider to change its issuance infrastructure and hence its versioning epoch independently without changing the CardId of all issued InfoCards. For example, when an identity provider makes a change to the supported claims or any other policy pertaining to the issued cards, the version number allows the identity provider to determine if the InfoCard needs to be renewed. The version number is assumed to be monotonically increasing. If two InfoCards have the same CardId value but different CardVersion values, then the one with a higher numerical CardVersion value should be treated as being more up-to-date.

/ic:InfoCard/ic:CardName

This optional element provides a friendly textual name for the issued InfoCard.

/ic:InfoCard/ic:CardImage

This optional element contains a base64 encoded inline image that provides a graphical image for the issued InfoCard that can be displayed in user interfaces. It should contain an image within the size range of 60 pixels wide by 45 pixels high, and 200 pixels wide by 150 pixels high.

/ic:InfoCard/ic:CardImage/@MimeType

This required attribute provides a MIME type specifying the format of the included logo image. The InfoCard Profile supports both the JPEG and GIF image formats (with MIME types of "image/jpeg" and "image/gif").

/ic:InfoCard/ic:IssuerName

This required element provides a friendly name for the issuer of the InfoCard.

/ic:InfoCard/ic:TimeIssued

This required element provides the date and time when the InfoCard was issued.

/ic:InfoCard/ic:TimeExpires

This optional element provides the date and time after which the InfoCard should be treated as expired and invalid.

/ic:InfoCard/ic:TokenServiceReference

This required element provides an ordered list of child elements that specify the security token service (IP/STS) endpoints, the corresponding authentication method and credentials needed to request security tokens. Each service endpoint should be tried in order by a requester when requesting security tokens.

/ic:InfoCard/ic:TokenServiceReference/ic:TokenService

This required element provides an IP/STS reference.

/ic:InfoCard/ic:TokenServiceReference/ic:TokenService/wsa:EndpointReference

This required element provides the endpoint reference for the IP/STS. For a self-issued identity provider, the special address value given in Section 4.1.1 will be used. The identity extension for endpoint reference should be used to convey the protection token for this endpoint to secure communications with it.

/ic:InfoCard/ic:TokenServiceReference/ic:TokenService/ ic:CredentialHint

This optional element provides a hint (string) to be displayed to the user to help provide the right credential (e.g. a hint to insert the right smart card).

/ic:InfoCard/ic:TokenServiceReference/ic:TokenService/<credential selector element>

This required element provides an unambiguous description of the credentials to use for authenticating to the IP/STS. The schema used to describe the credential is specific to each credential type and can be one of *ic:UserNamePasswordAuthenticate*, *ic:KerberosV5Authenticate*, *ic:X509V3Authenticate* or *ic:SelfIssuedAuthenticate*. (see Section 6).

/ic:InfoCard/ic:InfoCardPolicy

This required element provides the token issuance policy of the identity that allows a service requester system to determine if the InfoCard satisfies a relying party's token requirements in a given interaction (see Section 5.2).

/ic:InfoCard/ic:InfoCardPolicy/ic:SupportedTokenTypes

This required element contains the list of token types, as child elements, that the identity provider can issue.

/ic:InfoCard/ic:InfoCardPolicy/ic:SupportedTokenTypes/ic:TokenType (one or more)

This required element indicates an individual token type that is supported.

/ic:InfoCard/ic:InfoCardPolicy/ic:SupportedClaims

This required element contains the list of claim types, as child elements, that the identity provider can provide in security tokens.

/ic:InfoCard/ic:InfoCardPolicy/ic:SupportedClaims/ic:SupportedClaim (one or more)

This required element indicates an individual claim type that is supported.

/ic:InfoCard/ic:InfoCardPolicy/ic:RequireAppliesTo

This optional empty element indicates that the service requester (InfoCard system) must submit the relying party identity to the identity provider.

The following example illustrates an InfoCard issued by the "XYZ Authority" supporting the SAML token type, two claims, and requiring authentication based on username/password.

Example:

```
<InfoCard
  xmlns="http://schemas.microsoft.com/ws/2005/05/identity"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
  xml:lang="en-us">
  <InfoCardReference>
    <CardId>http://xyz.com/CardId/d795621fa01d454285f9bdf50c00cb2e</CardId>
  </InfoCardReference>
  <CardName>XYZ membership card</CardName>
  <CardImage MimeType="image/gif"> ... </CardImage>
  <IssuerName>XYZ Authority</IssuerName>
  <TimeIssued>2003-08-24T00:30:05Z</TimeIssued>
  <TokenServiceReference>
    <TokenService>
      <wsa:EndpointReference>
        <wsa:Address>http://xyz.org/sts</wsa:Address>
        <wsid:Identity>
          <ds:KeyInfo>
            <ds:X509Data>
              <ds:X509Certificate>...</ds:X509Certificate>
            </ds:X509Data>
          </ds:KeyInfo>
        </wsid:Identity>
      </wsa:EndpointReference>
      <UserNamePasswordAuthenticate>
        <Username>Zoe</Username>
      </UserNamePasswordAuthenticate>
    </TokenService>
  </TokenServiceReference>
  <ic:InfoCardPolicy>
    <SupportedTokenTypes>
      <TokenType URI="urn:oasis:names:tc:SAML:1.0:assertion"/>
    </SupportedTokenTypes>
    <SupportedClaims>
      <SupportedClaim URI="http://.../ws/2005/05/identity/claims/givenname">
        <DisplayTag>Given Name</DisplayTag>
      </SupportedClaim>
      <SupportedClaim URI="http://.../ws/2005/05/identity/claims/surname">
        <DisplayTag>Last Name</DisplayTag>
      </SupportedClaim>
    </SupportedClaims>
    <RequireAppliesTo />
  </ic:InfoCardPolicy>
</InfoCard>
```

5.1.2. Obtaining InfoCards

When an InfoCard is delivered out-of-band, it should be delivered inside a digitally signed envelope (i.e. enveloping signature) signed by the identity provider. The digitally signed

envelope is the moral equivalent of a signed message, which provides a signed container, in which the application payload gets delivered to a recipient. In this case, the InfoCard itself is included as the element content of the `ds:Object` element inside the enveloping signature. The signature on the digitally signed envelope makes no implicit statement about the content of the InfoCard itself except providing data origin authentication.

The specific details of the XML digital signature [[XMLDSIG](#)] profile that should be used to sign the envelope carrying the InfoCard is as follows:

- Use *enveloping signature* format when signing the InfoCard XML document.
- Use a single `ds:Object` element within the signature to hold the `ic:InfoCard` element that represents the issued InfoCard. The `ds:Object/@Id` attribute provides a convenient way for referencing the InfoCard from the `ds:SignedInfo/ds:Reference` element within the signature.
- Use RSA signing and verification with the algorithm identifier given by the URI <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.
- Use exclusive canonicalization with the algorithm identifier given by the URI <http://www.w3.org/2001/10/xml-exc-c14n#>.
- Use SHA1 digest method for the data elements being signed with the algorithm identifier <http://www.w3.org/2000/09/xmldsig#sha1>.
- There should be no other transforms used in the enveloping signature for the InfoCard other than the ones listed above.
- The `ds:KeyInfo` element must be present in the signature carrying the signing key information in the form of an X509v3 certificate (or a X509v3 certificate chain) specified as one or more `ds:X509Certificate` elements within a `ds:X509Data` element.

The following example shows the general form of an InfoCard within an enveloping signature signed by the identity provider using the format outlined above.

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_InfoCard">
      <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_InfoCard">
    <ic:InfoCard
```

```

    xmlns:ic="http://schemas.microsoft.com/ws/2005/05/identity"
    xml:lang="en-us">
    [Actual InfoCard content]
  </ic:InfoCard>
</Object>
</Signature>

```

5.2. Token Issuance Policy of Identity Provider

This section describes the various policy assertion elements that can be used by an identity provider to represent its token issuance policy either as metadata or in an issued InfoCard (see InfoCard schema outlined in Section 5.1.1). One or more of these token issuance policy assertions can appear as children of the `ic:InfoCardPolicy` element inside the InfoCard.

5.2.1. Supported token types

The list of different security token types that can be issued by the IP/STS for an InfoCard can be expressed using the policy element below.

Syntax:

```

<ic:SupportedTokenTypes ...>
  <ic:TokenType URI="xs:anyURI" /> +
</ic:SupportedTokenTypes>

```

The following describes the elements listed in the schema outlined above:

/ic:SupportedTokenTypes

This element is used to express the list of token types that the IP/STS is capable of issuing.

/ic:SupportedTokenTypes/ic:TokenType

This required element indicates an individual token type that the IP/STS can issue.

/ic:SupportedTokenTypes/ic:TokenType/@URI

This required attribute provides the unique identifier (URI) of the individual token type that the IP/STS can issue.

The following example illustrates a supported tokens policy of an identity provider that can issue both SAML 1.1 and SAML 2.0 tokens.

Example:

```

<ic:SupportedTokenTypes>
  <ic:TokenType URI="urn:oasis:names:tc:SAML:1.1" />
  <ic:TokenType URI="urn:oasis:names:tc:SAML:2.0" />
</ic:SupportedTokenTypes>

```

5.2.2. Supported claim types

The list of different claim types that can be asserted in security tokens issued by the IP/STS for an InfoCard can be expressed using the policy element below.

Syntax:

```

<ic:SupportedClaims ...>
  (<ic:SupportedClaim URI="xs:anyURI" ...>
    <ic:DisplayTag>xs:string</ic:DisplayTag> ?
    <ic:Description>xs:string</ic:Description> ?
    ...
  </ic:SupportedClaim>) +
</ic:SupportedClaims>

```

The following describes the elements listed in the schema outlined above:

/ic:SupportedClaims

This element is used to express the list of claims that the IP/STS is capable of issuing.

/ic:SupportedClaims/ic:SupportedClaim

This required element indicates an individual claim that the IP/STS can assert.

/ic:SupportedClaims/ic:SupportedClaim/@URI

This required attribute provides the unique identifier (URI) of the individual claim that the IP/STS can assert.

/ic:SupportedClaims/ic:SupportedClaim/ic:DisplayTag

This optional element provides a friendly name for this claim that can be shown in user interfaces.

/ic:SupportedClaims/ic:SupportedClaim/ic:Description

This optional element provides a description of the semantics for this claim.

/ic:SupportedClaims/ic:SupportedClaim/{any}

This is an extensibility mechanism to allow additional elements to be specified.

/ic:SupportedClaims/ic:SupportedClaim/@{any}

This is an extensibility mechanism to allow additional attributes to be specified.

The following example illustrates a supported claim types policy of an identity provider.

Example:

```
<ic:SupportedClaims>
  <ic:SupportedClaim URI="http://.../ws/2005/05/identity/claims/givename">
    <ic:DisplayTag>Given Name</ic:DisplayTag>
  </ic:SupportedClaim>
  <ic:SupportedClaim URI="http://.../ws/2005/05/identity/claims/surname">
    <ic:DisplayTag>Last Name</ic:DisplayTag>
    <ic:Description>A person's last name</ic:Description>
  </ic:SupportedClaim>
</ic:SupportedClaims>
```

5.2.3. Require relying party identity

An identity provider can assert that the identity of the relying party be conveyed in the `wsp:AppliesTo` element of the token request message by using the policy element below.

Syntax:

```
<ic:RequireAppliesTo ... /> ?
```

The following describes the elements listed in the schema element outlined above:

/ic:RequireAppliesTo

This optional empty element asserts that a service requester (InfoCard system) must submit the relying party identity to the IP/STS during security token requests.

5.3. Security Token Request to Identity Provider

When the user selects an InfoCard on a service requester system, the InfoCard system obtains a suitable security token from the IP/STS for that InfoCard. Security tokens are requested using the issuance mechanism described in [\[WS-Trust\]](#). This section describes specific constraints and/or extensions to the token request messages sent to the IP/STS.

5.3.1. Supplying InfoCard reference

Each InfoCard has a unique identifier and version, given by the `ic:InfoCardReference` element in an InfoCard, by which it can be referenced. When requesting a security token from the IP/STS, the service requester MUST include the InfoCard reference in the body of the “request security token” (RST) message as a top-level element information item. The `ic:InfoCardReference` element information item including all of its [children], [attributes] and [in-scope namespaces] is copied as an immediate child of the RST element in the message as follows.

Syntax:

```
<wst:RequestSecurityToken>
  ...
  <ic:InfoCardReference>
    <ic:CardId> ... </ic:CardId>
    <ic:CardVersion> ... </ic:CardVersion>
  </ic:InfoCardReference>
  ...
</wst:RequestSecurityToken>
```

The IP/STS MAY fault with `ic:InfoCardRefreshRequired` to signal to the service requester that the InfoCard needs to be refreshed.

5.3.2. Identifying relying party to identity provider

The `ic:OpaqueEndpoint` element defined below is used to send an opaque reference for a relying party identity to an IP/STS.

Syntax:

```
<ic:OpaqueEndpoint> xs:base64Binary </ic:OpaqueEndpoint>
```

The following describes the attributes and elements listed in the schema overview above:

/ic:OpaqueEndpoint

A base64 encoded opaque reference to an endpoint.

An opaque reference for the relying party identity is submitted to the IP/STS by including a `wsp:AppliesTo` element containing the opaque handle for the relying party endpoint in the RST request message as shown in the following example.

Example:

```
<wst:RequestSecurityToken>
  <wsp:AppliesTo>
    <ic:OpaqueEndpoint>
      MIIEZzCCA9CgAwIBAgIQEmtJZc0==
    </ic:OpaqueEndpoint>
  </wsp:AppliesTo>
  ...
</wst:RequestSecurityToken>
```

When required, the actual relying party identity is submitted to the IP/STS by including a `wsp:AppliesTo` element containing the endpoint reference of the relying party in the RST request message as shown in the following example.

Example:

```
<wst:RequestSecurityToken>
  <wsp:AppliesTo>
    <wsa:EndpointReference>
```

```

    <wsa:Address>http://ip.fabrikam.com/STS</wsa:Address>
    ...
  </wsa:EndpointReference>
</wsp:AppliesTo>
...
</wst:RequestSecurityToken>

```

5.3.3. Requesting display token

A display token, containing a textual representation of the claims carried in the issued security token, can be requested from an IP/STS by including the following optional element in the RST request message from the service requester.

Syntax:

```
<ic:RequestDisplayToken xml:lang="xs:language"? ... />
```

The following describes the attributes and elements listed in the schema outlined above:

/ic:RequestDisplayToken

This optional element is used to request an identity provider to return a display token for the security token returned in the response.

/ic:RequestDisplayToken/@xml:lang

This optional attribute indicates language identifier, using the language codes specified in [RFC 3066], in which the display token content should be localized.

/ic:RequestDisplayToken/@{any}

This is an extensibility mechanism to allow additional attributes to be specified.

The format of the display token returned by the IP/STS in a response RSTR message is described later in Section 5.4.1.

5.3.4. Requesting proof key

When requesting a *symmetric key* token from the IP/STS, the identity selector on the service requester does not supply any key material. The IP/STS must generate the symmetric key for use in the token and return it in the response.

When requesting an *asymmetric key* token from the IP/STS, the identity selector on the service requester generates an **ephemeral 1024-bit RSA key pair** for use as the proof key and submits to the IP/STS by augmenting the RST message as follows:

- It includes a `wst:KeyType` element with the following URI value in the RST message to indicate that an asymmetric key based token is requested.

http://schemas.xmlsoap.org/ws/2004/04/security/trust/PublicKey

- It includes a `wst:UseKey` element in the RST message to indicate the public key to be used in the returned security token.
- It includes a supporting signature to prove ownership of the corresponding private key. The `ds:KeyInfo` element within the signature includes the actual public key in the form of a `ds:RSAKeyValue` child element of a `ds:KeyValue` element.
- The supporting signature is placed in the security header of the RST message where the signature for an endorsing supporting token would be placed as per the security header layout defined in [WS-SecurityPolicy].

5.4. Security Token Response from Identity Provider

This section defines the constructs used by an identity provider (IP/STS) to respond to security token requests from the identity selector on a service requester.

5.4.1. Returning display token

An IP/STS can use the following optional element to return a display token for the issued security token in the RSTR response message to the service requester.

Syntax:

```
<ic:RequestedDisplayToken ...>
  <ic:DisplayToken xml:lang="xs:language" ... >
    [ <ic:DisplayClaim URI="xs:anyURI" ...>
      <ic:DisplayTag> xs:string </ic:DisplayTag>
      <ic:Description> xs:string </ic:Description> ?
      <ic:DisplayValue> xs:string </ic:DisplayValue> *
    </ic:DisplayClaim> ] +
    |
    [ <ic:DisplayTokenText> xs:string </ic:DisplayTokenText> ]
    ...
  </ic:DisplayToken>
</ic:RequestedDisplayToken>
```

The following describes the attributes and elements listed in the schema outlined above:

/ic:RequestedDisplayToken

This optional element is used to return a display token for the security token returned in the response.

/ic:RequestedDisplayToken/ic:DisplayToken

The returned display token.

/ic:RequestedDisplayToken/ic:DisplayToken/@xml:lang

This required attribute indicates a language identifier, using the language codes specified in [\[RFC 3066\]](#), in which the display token content is localized.

/ic:RequestedDisplayToken/ic:DisplayToken/ic:DisplayClaim

This required element indicates an individual claim returned in the security token.

/ic:RequestedDisplayToken/ic:DisplayToken/ic:DisplayClaim/@URI

This required attribute provides the unique identifier (URI) of the individual claim returned in the security token.

/ic:RequestedDisplayToken/ic:DisplayToken/ic:DisplayClaim/ic:DisplayTag

This optional element provides a friendly name for the claim returned in the security token.

/ic:RequestedDisplayToken/ic:DisplayToken/ic:DisplayClaim/ic:Description

This optional element provides a description of the semantics for the claim returned in the security token.

/ic:RequestedDisplayToken/ic:DisplayToken/ic:DisplayClaim/ic:DisplayValue (one or more)

This optional element provides one or more displayable values for the claim returned in the security token.

/ic:RequestedDisplayToken/ic:DisplayToken/ic:DisplayTokenText

This element provides an alternative textual representation of the entire token as a whole when the token content is not suitable for display as individual claims.

/ic:RequestedDisplayToken/ic:DisplayToken/{any}

This is an extensibility mechanism to allow additional elements to be specified.

/ic:RequestedDisplayToken/ic:DisplayToken/@{any}

This is an extensibility mechanism to allow additional attributes to be specified.

The following example illustrates a returned display token corresponding to a security token with two claims.

Example:

```
<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>
```

5.4.2. Returning proof key

When a *symmetric key* token is requested, the IP/STS should return the symmetric proof key in the response by including a `wst:RequestedProofToken` element with:

- a `wst:BinarySecret` child element carrying the base64 encoded symmetric key (if transport security is used), or
- a `xenc:EncryptedKey` child element carrying the symmetric key encrypted to the service requester's authentication key (if message security is used).

When an *asymmetric key* token is requested, the IP/STS should not return a proof key if it accepts the proof key submitted in the token request. The IP/STS can **override** the submitted proof key by specifying a public key token of its choice (for which the requester already has the private key) as the proof token. In that case, a `wst:RequestedProofToken` element must be returned in the response containing a `ds:KeyInfo` child element that provides the selection criteria for the specific proof token. The Windows InfoCard system can only accept an X509v3 certificate (either software or hardware token based) as the overriding proof token. The X509v3 certificate to be used as proof token can be indicated by specifying the SHA1 hash of the entire certificate content as a "thumbprint" in the manner below:

Syntax:

```
<!-- Select certificate based on certificate thumbprint -->
<wst:RequestedProofToken>
  <ds:KeyInfo>
    <wsse:SecurityTokenReference>
      <wsse:KeyIdentifier Value="http://docs.oasis-open.org/wss/2004/xx/
oasis-2004xx-wss-soap-message-security-1.1#ThumbprintSHA1">
        <xs:base64binary>
          </xs:base64binary>
        </wsse:KeyIdentifier>
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </wst:RequestedProofToken>
```

6. Authenticating to Identity Provider

The InfoCard schema includes the element content necessary for an identity provider to express what credential the user must use in order to authenticate to the IP/STS when requesting tokens. This section defines the schema used to express the credential description for each supported credential type.

6.1. Using Username and Password

When the identity provider requires that the service requester submit a *username* and corresponding *password* as the credential to authenticate to the IP/STS, the following credential format should be used in the InfoCard to specify the required credential.

Syntax:

```
<ic:UserNamePasswordAuthenticate>  
  <ic:Username>xs:string</ic:Username> ?  
</ic:UserNamePasswordAuthenticate>
```

The following describes the attributes and elements listed in the schema outlined above:

/ic:UserNamePasswordAuthenticate/ic:Username

This optional element provides the “username” part of the credential; the service requester must prompt the user for the “password.” If the username is specified, then it must be used “as is” in the username token employed by the service requester to authenticate to the IP/STS; else the service requester must prompt the user for the “username” as well.

For user convenience, the “username” may be included in the credential description present in the InfoCard, but the “password” **must not** be included in the InfoCard.

6.2. Using KerberosV5 Service Ticket

When the identity provider requires that the service requester submit a *Kerberosv5 service ticket* for the IP/STS as the credential to authenticate to the IP/STS, the following credential format should be used in the InfoCard to specify the required credential.

Syntax:

```
<ic:KerberosV5Authenticate>  
  <ic:UserPrincipalName>xs:string</ic:UserPrincipalName> ?  
</ic:KerberosV5Authenticate>
```

The following describes the attributes and elements listed in the schema outlined above:

/ic:KerberosV5Authenticate/ic:UserPrincipalName

This optional element provides a “user principal name” that identifies the user as a specific principal in some domain using the format *user@domain*. Absence of this element implies the credential and Kerberos realm that the user is currently logged into.

To enable the service requester to obtain a Kerberosv5 service ticket for the IP/STS, the endpoint reference of the IP/STS specified in the InfoCard must include a *service principal name* claim as identity information as described in [[Addressing-Ext](#)]. The KDC in the appropriate domain/realm can identify the service account based on this information and issue the required Kerberosv5 service ticket. This would typically be used in enterprise intranet scenarios.

No explicit user credential needs to be specified in this case as it is implied by the Kerberos realm that the user is currently logged into. Optionally, a user’s credential may be specified as a *user principal name* that identifies the user as a specific principal in some domain using

the format *user@domain*. The user would prove this credential by supplying the password for the account to be submitted to the KDC of that domain. The InfoCard system in Windows will only support the *implicit* credential in this case given by an empty credential selector element. Any *explicitly* specified user principal name in the credential selector will be ignored.

6.3. Using Software Based X509 Certificate

When the identity provider requires that the service requester submit an *X509v3 certificate* for the user, where the certificate and keys are in a software-based store (e.g. Microsoft base cryptographic storage provider), as the credential to authenticate to the IP/STS, the following credential format should be used in the InfoCard to specify the required credential.

Syntax:

```
<ic:X509V3Authenticate>
  <ds:X509Data>
    <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/2004/xx/
oasis-2004xx-wss-soap-message-security-1.1#ThumbprintSHA1">
      xs:base64binary
    </wsse:KeyIdentifier>
  </ds:X509Data>
</ic:X509V3Authenticate>
```

The following describes the attributes and elements listed in the schema outlined above:

/ic:X509V3Authenticate/ds:X509Data/wsse:KeyIdentifier

This element provides a key identifier for the X509 certificate based on the SHA1 hash of the entire certificate content expressed as a "thumbprint." Note that the extensibility point in the *ds:X509Data* element is used to add *wsse:KeyIdentifier* as a child element.

The certificate thumbprint is used as a search value with the appropriate platform-specific APIs (e.g. CAPI2 on Windows) to locate the right certificate.

6.4. Using Smartcard Based X509 Certificate

When the identity provider requires that the service requester submit an *X509v3 certificate* for the user, where the certificate and keys are in a hardware-based smart card, as the credential to authenticate to the IP/STS, the following credential format should be used in the InfoCard to specify the required credential.

Syntax:

```
<ic:CredentialHint> xs:string </ic:CredentialHint> ?
<ic:X509V3Authenticate>
  <ds:X509Data>
    <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/2004/xx/
oasis-2004xx-wss-soap-message-security-1.1#ThumbprintSHA1">
      xs:base64binary
    </wsse:KeyIdentifier>
  </ds:X509Data>
</ic:X509V3Authenticate>
```

The following describes the attributes and elements listed in the schema outlined above:

/ic:CredentialHint

This optional element provides a user hint string which can be used to prompt the user for inserting the appropriate smart card into the reader.

/ic:X509V3Authenticate/ds:X509Data/wsse:KeyIdentifier

This element provides a key identifier for the X509 certificate based on the SHA1 hash of the entire certificate content expressed as a “thumbprint.” Note that the extensibility point in the `ds:X509Data` element is used to add `wsse:KeyIdentifier` as a child element.

The certificate thumbprint is used as a search value with the appropriate platform-specific APIs (e.g. CAPI2 on Windows) to locate the right certificate.

6.5. Using Self-issued Token

When the identity provider requires that the service requester submit a *self-issued SAML token*, carrying the asymmetric public key of the user known to the IP/STS, as the credential to authenticate to the IP/STS, the following credential format should be used in the InfoCard to specify the required credential.

Syntax:

```
<ic:SelfIssuedAuthenticate>  
  <ic:PrivatePersonalIdentifier>...</ic:PrivatePersonalIdentifier>  
</ic:SelfIssuedAuthenticate>
```

The following describes the attributes and elements listed in the schema outlined above:

/ic:SelfIssuedAuthenticate/ic:PrivatePersonalIdentifier

This required element provides the value of the private personal identifier claim asserted in the self-issued token used previously to register with the IP/STS.

The identity selector on the service requester can locate the right self-issued InfoCard based on the `ic:PrivatePersonalIdentifier` claim used to identify the user at the IP/STS.

7. Faults

In addition to the standard faults described in WS-Addressing, WS-Security and WS-Trust, the InfoCard Profile defines the following additional faults that may occur when interacting with the relying party or the identity provider. The binding of the fault properties (listed below) to a SOAP 1.1 or SOAP 1.2 fault message is described in [\[WS-Addressing\]](#). If the optional **[Detail]** property for a fault includes any specified content, then the corresponding schema fragment is included in the listing below.

7.1. Relying Party

The following faults may occur when submitting security tokens carrying claims to a relying party per its security policy.

[action]	http://schemas.microsoft.com/ws/2005/05/identity/fault
[Code]	S:Sender
[Subcode]	ic:RequiredClaimMissing
[Reason]	A required claim is missing from the security token.
[Detail]	[URI of missing claim] <ic:Claim URI="[claim URI]" />

[action]	http://schemas.microsoft.com/ws/2005/05/identity/fault
[Code]	S:Sender
[Subcode]	ic:InvalidClaimValue
[Reason]	A claim value asserted in the security token is invalid.
[Detail]	[URI of invalid claim] <ic:Claim URI="[claim URI]" />

7.2. Identity Provider

The following faults may occur when requesting security tokens from an identity provider using InfoCards.

[action]	http://schemas.microsoft.com/ws/2005/05/identity/fault
[Code]	S:Sender
[Subcode]	ic:MissingAppliesTo
[Reason]	The request is missing relying party identity information.
[Detail]	(None defined.)

[action]	http://schemas.microsoft.com/ws/2005/05/identity/fault
[Code]	S:Sender
[Subcode]	ic:InvalidProofKey
[Reason]	Invalid proof key specified in request.
[Detail]	(None defined.)

[action]	http://schemas.microsoft.com/ws/2005/05/identity/fault
[Code]	S:Sender
[Subcode]	ic:UnknownInfoCardReference
[Reason]	Unknown InfoCard reference specified in request.
[Detail]	[Unknown InfoCard reference] <ic:InfoCardReference> <ic:CardId>[card ID]</ic:CardId> <ic:CardVersion>[version]</ic:CardVersion> </ic:InfoCardReference>

[action]	http://schemas.microsoft.com/ws/2005/05/identity/fault
[Code]	S:Sender
[Subcode]	ic:FailedRequiredClaims
[Reason]	Could not satisfy required claims in request; construction of token failed
[Detail]	[URIs of claims that could not be satisfied] <ic:Claim URI="[claim URI]" /> <ic:Claim URI="[claim URI]" /> ...

[action]	http://schemas.microsoft.com/ws/2005/05/identity/fault
[Code]	S:Sender
[Subcode]	ic:InfoCardRefreshRequired
[Reason]	Stale InfoCard reference specified in request; InfoCard should be refreshed
[Detail]	[InfoCard reference that needs refreshing] <ic:InfoCardReference> <ic:CardId>[card ID]</ic:CardId> <ic:CardVersion>[version]</ic:CardVersion> </ic:InfoCardReference>

8. References

[HTTP]

R. Fielding et al, "[IETF RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1](#)", June 1999.

[HTTPS]

E. Rescorla, "[RFC 2818: HTTP over TLS](#)", May 2000.

[InfoCard-Guide]

"[A Guide to Integrating with InfoCard v1.0](#)," August 2005.

[RFC 2119]

S. Bradner, "[RFC 2119: Key words for use in RFCs to Indicate Requirement Levels](#)", March 1997.

[RFC 3066]

H. Alvestrand, "[Tags for the Identification of Languages](#)", January 2001.

[SOAP 1.2]

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

[WS-Addressing]

D. Box et al, "[Web Services Addressing \(WS-Addressing\)](#)," August 2004.

[Addressing-Ext]

Document to be published in the near future.

[WS-MetadataExchange]

"Web Services Metadata Exchange (WS-MetadataExchange)," August 2004.

[WS-Security]

A. Natalin et al, "[Web Services Security: SOAP Message Security 1.0](#)", May 2004.

[WS-Policy]

"Web Services Policy Framework (WS-Policy)," September 2004.

[WS-SecurityPolicy]

"Web Services Security Policy Language (WS-SecurityPolicy)," March 2005.

[WS-Trust]

"Web Services Trust Language (WS-Trust)," May 2004.

[XMLDSIG]

Eastlake III, D., Reagle, J., and Solo, D., "XML-Signature Syntax and Processing",
<http://www.ietf.org/rfc/rfc3275.txt>, March 2002.

[XMLENC]

Imamura, T., Dillaway, B., and Simon, E., "XML Encryption Syntax and Processing",
<http://www.w3.org/TR/xmlenc-core/>, 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 – Claims Defined by InfoCard Profile

The following set of claim (attribute) types and the corresponding URIs are defined by the InfoCard model. These claim URIs can be used in the token requirements policy of a relying party or the issuance policy of an identity provider. Note that, where possible, the claims included here reuse and refer to the attribute semantics defined in other popular industry standards that deal with personal information about persons.

A.1. First Name

URI: *http://schemas.microsoft.com/ws/2005/05/identity/claims/givenname*

Type: *xs:string*

Definition: (*givenName* in RFC 2256) Preferred name or first name of a subject. According to RFC 2256: "This attribute is used to hold the part of a person's name which is not their surname nor middle name."

A.2. Last Name

URI: *http://schemas.microsoft.com/ws/2005/05/identity/claims/surname*

Type: *xs:string*

Definition: (*sn* in RFC 2256) Surname or family name of a subject. According to RFC 2256: "This is the X.500 surname attribute which contains the family name of a person."

A.3. Email Address

URI: *http://schemas.microsoft.com/ws/2005/05/identity/claims/emailaddress*

Type: *xs:string*

Definition: (*mail* in *inetOrgPerson*) Preferred address for the "To:" field of email to be sent to the subject, usually of the form <user>@<domain>. According to *inetOrgPerson* using RFC 1274: "This attribute type specifies an electronic mailbox attribute following the syntax specified in RFC 822."

A.4. Street Address

URI: *http://schemas.microsoft.com/ws/2005/05/identity/claims/streetaddress*

Type: *xs:string*

Definition: (*street* in RFC 2256) Street address component of a subject's address information. According to RFC 2256: "This attribute contains the physical address of the object to which the entry corresponds, such as an address for package delivery." Its content is arbitrary, but typically given as a PO Box number or apartment/house number followed by a street name, *e.g.* 303 Mulberry St.

A.5. Locality Name or City

URI: *http://schemas.microsoft.com/ws/2005/05/identity/claims/locality*

Type: *xs:string*

Definition: (*l* in RFC 2256) Locality component of a subject's address information. According to RFC 2256: "This attribute contains the name of a locality, such as a city, county or other geographic region." *e.g.* Redmond.

A.6. State or Province

URI: <http://schemas.microsoft.com/ws/2005/05/identity/claims/stateorprovince>

Type: *xs:string*

Definition: (*st* in RFC 2256) Abbreviation for state or province name of a subject's address information. According to RFC 2256: "This attribute contains the full name of a state or province. The values should be coordinated on a national level and if well-known shortcuts exist - like the two-letter state abbreviations in the US - these abbreviations are preferred over longer full names." *e.g.* WA.

A.7. Postal Code

URI: <http://schemas.microsoft.com/ws/2005/05/identity/claims/postalcode>

Type: *xs:string*

Definition: (*postalCode* in X.500) Postal code or zip code component of a subject's address information. According to X.500(2001): "The postal code attribute type specifies the postal code of the named object. If this attribute value is present, it will be part of the object's postal address - zip code in USA, postal code for other countries."

A.8. Country

URI: <http://schemas.microsoft.com/ws/2005/05/identity/claims/country>

Type: *xs:string*

Definition: (*c* in RFC 2256) Country of a subject. According to RFC 2256: "This attribute contains a two-letter ISO 3166 country code."

A.9. Primary or Home Telephone Number

URI: <http://schemas.microsoft.com/ws/2005/05/identity/claims/homephone>

Type: *xs:string*

Definition: (*homePhone* in *inetOrgPerson*) Primary or home telephone number of a subject. According to *inetOrgPerson* using RFC 1274: "This attribute type specifies a home telephone number associated with a person." Attribute values should follow the agreed format for international telephone numbers, *e.g.* +44 71 123 4567.

A.10. Secondary or Work Telephone Number

URI: <http://schemas.microsoft.com/ws/2005/05/identity/claims/otherphone>

Type: *xs:string*

Definition: (*telephoneNumber* in X.500 *Person*) Secondary or work telephone number of a subject. According to X.500(2001): "This attribute type specifies an office/campus telephone number associated with a person." Attribute values should follow the agreed format for international telephone numbers, *e.g.* +44 71 123 4567.

A.11. Mobile Telephone Number

URI: <http://schemas.microsoft.com/ws/2005/05/identity/claims/mobilephone>

Type: *xs:string*

Definition: (*mobile* in *inetOrgPerson*) Mobile telephone number of a subject. According to *inetOrgPerson* using RFC 1274: "This attribute type specifies a mobile telephone number associated with a person." Attribute values should follow the agreed format for international telephone numbers, *e.g.* +44 71 123 4567.

A.12. Date of Birth

URI: <http://schemas.microsoft.com/ws/2005/05/identity/claims/dateofbirth>

Type: *xs:date*

Definition: The date of birth of a subject in a form allowed by the *xs:date* data type.

A.13. Gender

URI: <http://schemas.microsoft.com/ws/2005/05/identity/claims/gender>

Type: *xs:token*

Definition: Gender of a subject that can have any of these exact string values - "Male," "Female" or "Unspecified."

A.14. Private Personal Identifier

URI: <http://schemas.microsoft.com/ws/2005/05/identity/claims/privatepersonalidentifier>

Type: *xs:base64binary*

Definition: Indicates a private identifier that identifies the subject to a relying party. The word "private" is used in the sense that the subject identifier is specific to a given relying party and hence private to that relying party. A subject's private personal identifier (PPI) at one relying party cannot be correlated with the subject's PPI at another relying party. Typically, the PPI for a subject is generated by an identity provider for a given relying party. For a self-issued InfoCard, the self-issued identity provider in the InfoCard system will generate a PPI for each relying party where the InfoCard is used as a function of the card identifier and the relying party's identity.

Appendix B – XSD

An informational copy of the XML schema used by this document is listed below for convenience.

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema
  targetNamespace="http://schemas.microsoft.com/ws/2005/05/identity"
  xmlns:tns="http://schemas.microsoft.com/ws/2005/05/identity"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  elementFormDefault="qualified" blockDefault="#all" version="0.1">

  <xs:import
    namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    schemaLocation=
      "http://schemas.xmlsoap.org/ws/2004/08/addressing/addressing.xsd"/>
  <xs:import
    namespace="http://schemas.xmlsoap.org/ws/2004/09/policy"
    schemaLocation=
      "http://schemas.xmlsoap.org/ws/2004/09/policy/ws-policy.xsd"/>
  <xs:import
    namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation=
      "http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-
schema.xsd"/>
  <xs:import
    namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/xml.xsd"/>

  <!-- Standard claim types defined by the InfoCard model -->
  <xs:simpleType name="StringMaxLength255">
    <xs:restriction base="xs:string">
      <xs:maxLength value="255" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="GivenName" type="tns:StringMaxLength255" />
  <xs:element name="Surname" type="tns:StringMaxLength255" />
  <xs:element name="EmailAddress" type="tns:StringMaxLength255" />
  <xs:element name="StreetAddress" type="tns:StringMaxLength255" />
  <xs:element name="Locality" type="tns:StringMaxLength255" />
  <xs:element name="StateOrProvince" type="tns:StringMaxLength255" />
  <xs:element name="PostalCode" type="tns:StringMaxLength255" />
  <xs:element name="Country" type="tns:StringMaxLength255" />
  <xs:element name="PrimaryPhone" type="tns:StringMaxLength255" />
  <xs:element name="DateOfBirth" type="xs:date" />
  <xs:element name="PrivatePersonalIdentifier" type="xs:base64Binary" />
  <xs:element name="Gender" type="tns:GenderType" />
  <xs:simpleType name="GenderType">
    <xs:restriction base="xs:token">
      <xs:enumeration value="Male" />
      <xs:enumeration value="Female" />
      <xs:enumeration value="Unspecified" />
    </xs:restriction>
  </xs:simpleType>

```

```

    </xs:restriction>
</xs:simpleType>

<!-- General use elements and attributes -->
<xs:complexType name="AttributedEmptyElement">
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
<xs:element name="CardName" type="xs:string" />
<xs:element name="IssuerName" type="xs:string" />
<xs:element name="TimeIssued" type="xs:dateTime" />
<xs:element name="TimeExpires" type="xs:dateTime" />
<xs:element name="RequireAppliesTo" type="tns:AttributedEmptyElement" />
<xs:element name="MaxTokenAge" type="xs:unsignedLong" />
<xs:element name="OpaqueEndpoint" type="xs:base64Binary" />
<xs:simpleType name="LogoImageTypes">
  <xs:restriction base="xs:token">
    <xs:enumeration value="image/jpeg" />
    <xs:enumeration value="image/gif" />
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="BaseClaimType" abstract="true">
  <xs:attribute name="URI" type="xs:anyURI" use="required" />
</xs:complexType>
<xs:element name="Claim" type="tns:ClaimType" />
<xs:complexType name="ClaimType">
  <xs:complexContent>
    <xs:extension base="tns:BaseClaimType">
      <xs:attribute name="Optional" type="xs:boolean" />
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="SupportedClaim" type="tns:SupportedClaimType" />
<xs:complexType name="SupportedClaimType">
  <xs:complexContent>
    <xs:extension base="tns:BaseClaimType">
      <xs:sequence>
        <xs:element name="DisplayTag" type="xs:string" minOccurs="0" />
        <xs:element name="Description" type="xs:string" minOccurs="0" />
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="InfoCardReference" type="tns:InfoCardReferenceType" />
<xs:complexType name="InfoCardReferenceType">
  <xs:sequence>
    <xs:element name="CardId" type="xs:anyURI" />
    <xs:element name="CardVersion" type="xs:unsignedInt" minOccurs="0" />
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
<xs:element name="CardImage" type="tns:CardImageType" />
<xs:complexType name="CardImageType">
  <xs:simpleContent>
    <xs:extension base="xs:base64Binary">

```

```

        <xs:attribute name="MimeType" type="tns:LogoImageTypes"
use="required" />
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:element name="CredentialHint" type="xs:string" />
<xs:element name="Username" type="xs:string" />
<xs:element name="UserPrincipalName" type="xs:string" />
<xs:element name="UserNamePasswordAuthenticate"
    type="tns:UserNamePasswordAuthenticateType" />
<xs:complexType name="UserNamePasswordAuthenticateType">
    <xs:sequence>
        <xs:element ref="tns:Username" minOccurs="0" />
    </xs:sequence>
</xs:complexType>
<xs:element name="KerberosV5Authenticate"
    type="tns:KerberosV5AuthenticateType" />
<xs:complexType name="KerberosV5AuthenticateType">
    <xs:sequence>
        <xs:element ref="tns:UserPrincipalName" minOccurs="0" />
    </xs:sequence>
</xs:complexType>
<xs:element name="X509V3Authenticate" type="tns:X509V3AuthenticateType" />
<xs:complexType name="X509V3AuthenticateType">
    <xs:sequence>
        <xs:element ref="ds:X509Data" />
    </xs:sequence>
</xs:complexType>
<xs:element name="SelfIssuedAuthenticate"
    type="tns:SelfIssuedAuthenticateType" />
<xs:complexType name="SelfIssuedAuthenticateType">
    <xs:sequence>
        <xs:element ref="tns:PrivatePersonalIdentifier" />
    </xs:sequence>
</xs:complexType>
<xs:element name="TokenService" type="tns:TokenServiceType" />
<xs:complexType name="TokenServiceType">
    <xs:sequence>
        <xs:element ref="wsa:EndpointReference" />
        <xs:element ref="tns:CredentialHint" minOccurs="0" />
        <xs:choice>
            <xs:element ref="tns:SelfIssuedAuthenticate" />
            <xs:element ref="tns:X509V3Authenticate" />
            <xs:element ref="tns:UserNamePasswordAuthenticate" />
            <xs:element ref="tns:KerberosV5Authenticate" />
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xs:choice>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
<xs:element name="TokenServiceReference"
    type="tns:TokenServiceReferenceType" />
<xs:complexType name="TokenServiceReferenceType">
    <xs:sequence maxOccurs="unbounded">
        <xs:element ref="tns:TokenService" />
    </xs:sequence>

```

```

        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
    <xs:element name="TokenType" type="tns:TokenType" />
    <xs:complexType name="TokenType">
        <xs:attribute name="URI" type="xs:anyURI" use="required" />
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
    <xs:element name="SupportedTokenTypes" type="tns:SupportedTokenTypes" />
    <xs:complexType name="SupportedTokenTypes">
        <xs:sequence>
            <xs:element ref="tns:TokenType" maxOccurs="unbounded" />
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
    <xs:element name="SupportedClaims" type="tns:SupportedClaimsType" />
    <xs:complexType name="SupportedClaimsType">
        <xs:sequence>
            <xs:element ref="tns:SupportedClaim" maxOccurs="unbounded" />
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>
    <xs:element name="InfoCardPolicy" type="tns:InfoCardPolicyType" />
    <xs:complexType name="InfoCardPolicyType">
        <xs:sequence>
            <xs:element ref="tns:SupportedTokenTypes" />
            <xs:element ref="tns:SupportedClaims" minOccurs="0" />
            <xs:element ref="tns:RequireAppliesTo" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>
    <xs:element name="PrivacyPolicyText" type="xs:string" />
    <xs:element name="PrivacyPolicyLink" type="xs:anyURI" />
    <xs:element name="IssuerPrivacyPolicy" type="tns:IssuerPrivacyPolicyType" />
    <xs:complexType name="IssuerPrivacyPolicyType">
        <xs:sequence>
            <xs:element ref="tns:PrivacyPolicyText" />
            <xs:element ref="tns:PrivacyPolicyLink" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>
    <xs:element name="InfoCard" type="tns:InfoCardType" />
    <xs:complexType name="InfoCardType">
        <xs:sequence>
            <xs:element ref="tns:IssuerPrivacyPolicy" />
            <xs:element ref="tns:InfoCardReference" />
            <xs:element ref="tns:CardName" minOccurs="0" />
            <xs:element ref="tns:CardImage" minOccurs="0" />
            <xs:element ref="tns:IssuerName" />
            <xs:element ref="tns:TimeIssued" />
            <xs:element ref="tns:TimeExpires" minOccurs="0" />
            <xs:element ref="tns:TokenServiceReference" />
            <xs:element ref="tns:InfoCardPolicy" />
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:complexType>

```

```

<xs:element name="RequestDisplayToken">
  <xs:complexType>
    <xs:attribute ref="xml:lang" use="optional" />
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
</xs:element>
<xs:element name="DisplayClaim" type="tns:DisplayClaimType" />
<xs:complexType name="DisplayClaimType">
  <xs:complexContent>
    <xs:extension base="tns:SupportedClaimType">
      <xs:sequence>
        <xs:element name="DisplayValue" type="xs:string" minOccurs="0"
maxOccurs="unbounded" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="DisplayTokenText" type="tns:DisplayTokenTextType" />
<xs:complexType name="DisplayTokenTextType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:anyAttribute namespace="##other" processContents="lax" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:element name="DisplayToken" type="tns:DisplayTokenType" />
<xs:complexType name="DisplayTokenType">
  <xs:choice>
    <xs:element ref="tns:DisplayClaim" maxOccurs="unbounded" />
    <xs:element ref="tns:DisplayTokenText" />
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
  </xs:choice>
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
<xs:element name="RequestedDisplayToken">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="tns:DisplayToken" />
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
</xs:element>
</xs:schema>

```