OASIS SSTC SAML Assertion Schema Discussion

draft-sstc-core-discussion-00.doc

23 July 2001

Authors:

Chris McLaren, Netegrity

Prateek Mishra, Netegrity

The Design Principles section is largely word-for-word from Dave Orchard and Eve Mahler’s draft.

1 Document Scope

This document and a companion document (draft-sstc-protocols-discussion-00) provide discussion and examples of schema elements and types given in draft-assertion-schema-10 and draft-protocol-10. A normative specification document describing draft-assertion-schema-10 and draft-protocol-10 will be published separately.

2 Design Principles

The proposed design adheres to the following principles for XML structure design:

1. Strong-typing of elements: Use XML Schema complex typing and inheritance to isolate commonalities. This allows XML validators to function as "free error checkers" on assertions and improves performance of streaming tools. Extension points can be created by adding some abstract "base types" to the design.
2. Resist typing of data: The contents of leaf nodes have been set to either string or uriReference. This does not reflect a rejection of the notion that some of these elements need additional restrictions on their contents, but rather indicates a desire to avoid getting drawn into the mire of "identifier religion". Once the first-order questions of what the structure of assertions and request/response pairs looks like are answered then the TC can address what, if any, restrictions need to be placed on the contents of the leaf nodes.
3. Isolate extensions: Use XML Namespaces and XML Schema to isolate extensibility features where possible, so that schema modules can be used to ensure compliance with extensions and so that extensions can be uniquely referred to with XML namespace names. This makes it easier to describe conformance to extensions.
4. Existing vocabularies: Existing XML vocabularies that are well supported, and that directly address a SAML need should be used, where they exist, in preference to new semantics. For example, if SAML needed a facility for marking up error messages, it should prefer XHTML to a new SAML-specific vocabulary. This is illustrated in the used of the XML-DSIG types for handling public key information.
5. Elements vs. attributes: Tend towards attributes for metadata and "single-field" information, and elements for any content that has distinguishable subparts.
6. Distinguish clearly between required elements/attributes and optional elements/attributes. Justify clearly rich cardinalities of the type "zero/one or more" instances of an element.

3 General Architecture

3.1 Discussion and Issues

3.1.1 Aggregating Assertions

Following the discussion at the third f2f no element has been provided for explicitly aggregating or collecting multiple assertions into a single object. Various SAML elements do provide context-dependent containers for assertions (e.g., <Evidence>) as needed in SAML messages.

3.1.1.1 ISSUE:[CONS-01] Aggregation

Do we need an explicit element for aggregating multiple assertions into a single object as part of the SAML specification? If so, what is the type of this element?

3.1.2 ID Types

There are a variety of places throughout the specification where objects are required to have an identifier: assertions, requests, and responses all have (unique) identifiers, and the identifiers of the initiating requests are also quoted back as part of responses.

These identifiers are all typed as instances of the “IDType”, which is in turn defined as an XML Schema simple type. At present the only restriction on this type is that it must be a string.

Should additional constraints on the form of the identifier be deemed necessary this type’s definition can be altered. Should it be deemed necessary that the form of assertion IDs needs to differ from the form of, for example, request IDs then the IDType can be extended into the relevant number of descendant IDTypes.

This issue corresponds to ISSUE:[F2F#3-8] from [f2f3-minutes] which should be consulted at this point.

3.1.2.1 ISSUE:[CONS-02] IDType

Does the specification need additional specification for the types of assertion, request, and response IDs? If so, what are these requirements?

3.1.2.2 ISSUE:[CONS-03] Final Types vs Extensible types

Does the TC plan to restrict certain types in the SAML schema to be final? If so, which types are to be so restricted?

4 Assertion Specification

4.1 Discussion and Issues

4.1.1 Inheritance Structure

The specification defines three different types of assertion: authentication assertions, attribute assertions, and authorization decision assertions. All of these assertion types are extensions of the abstract base “subject assertion”, which is in turn an extension of the abstract base assertion type.

This means that all three of the defined assertion types share the structure of a “subject assertion”. Furthermore, since this common structure is contained within the abstract base class it is available for extension, allowing new assertion types that share this structure to be defined in the future.

The assertion base is also defined and exposed, allowing for possible future extension to create assertions that do not refer to a subject.

4.1.1.1 ISSUE:[CONS-04] Extension Schema Structure

One of the goals of the f2f3 “whiteboard draft” was to use strong typing to differentiate between the three assertion types and between the three different query forms. This has been achieved through the use of ``abstract'' schema and schema inheritance. One implication is that any concrete assertion instance MUST utilize the xsi:type attribute to specifically describe its type even as all assertions will continue to use a single <Assertion> element as their container. XML processors can key off this attribute during assertion processing.

Is this an acceptable approach? Other approaches, such as the use of substitution groups, are also available. Using substition groups, each concrete assertion type would receive its own distinguished top-level element (e.g., <AuthenticationAssertion>) and there would be no need for the use of xsi:type attribute in any assertion instance. At the same time the SAML schema would be made somewhat more complex through the use of substitution groups.

Should the TC investigate these other approaches? Most important: what is the problem with the current approach?

4.1.2 Abstract Assertion type

```
<element name="Assertion" type="saml:AssertionType"/>
<complexType name="AssertionType" abstract="true">
      <sequence>
            <element name="Conditions" type="saml:ConditionsType"
             minOccurs="0"/>
            <element name="Advice" type="saml:AdviceType" minOccurs="0"/>
      </sequence>
      <attribute name="Version" type="string" use="required"/>
      <attribute name="AssertionID" type="saml:IDType" use="required"/>
      <attribute name="Issuer" type="string" use="required"/>
      <attribute name="IssueInstant" type="timeInstant" use="required"/>
</complexType>
```

The abstract assertion base type contains the common "header" information that is required in an assertion as well as optionally containing a collection of optional conditions and advice. Note that AssertionType is an **abstract** type; it can not be instantiated, it is only useful as a base for inheritance.

Version: This required attribute holds the string that uniquely identifies the version of the SAML specification within which this assertion was defined.

AssertionID: This required attribute is a string which identifies this assertion.

Issuer: This required attribute is the string the issuer provided at creation of the assertion. At present this is defined simply as a string. Additional requirements for this attribute's form may be defined by the committee.

IssueInstant: This required attribute specifies the instant at which the assertion was issued.

4.1.2.1 ISSUE:[CONS-05] Issuer

Does the specification need to further specify the Issuer element? Is a string type adequate for its use in SAML? Discussion [F1] from [f2f3-minutes] points to the relevant thread on the list.

4.1.2.2 ISSUE:[CONS-06] Version

Does the specification need to define to further specify the version element? If so, what are these requirements? Should this be a string? Or is an `unsignedint` enough?

4.1.3 Conditions

```
<complexType name="ConditionsType">
      <sequence>
            <element name="Condition" type="saml:AbstractConditionType"
             minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="NotBefore" type="timeInstant" use="optional"/>
      <attribute name="NotOnOrAfter" type="timeInstant" use="optional"/>
</complexType>
```

```
<xsd:complexType name="AbstractConditionType" abstract="true"/>

<xsd:complexType name="AudienceRestrictionConditionType">
      <xsd:complexContent>
            <xsd:extension base="saml:AbstractConditionType">
                  <xsd:sequence>
                        <xsd:element name="Audience" type="xsd:anyURI"
                         minOccurs="0" maxOccurs="unbounded"/>
                  </xsd:sequence>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

The <Conditions> element contains zero or more <Condition> elements, as well as optionally containing attributes that define the validity period over which the assertion is valid.

From the perspective of an RP the validity of a <Conditions> element is defined by:

(a) validity period as defined by the **NotBefore** and **NotOnOrAfter** attributes, AND

(b) the validity of the conjunction of the all of the <AbstractCondition> elements contained within it.

The only concrete condition type that is defined is the <AudienceRestrictionCondition>. This is a container for a sequence of <Audience> elements, each of which is a URI reference that specifies an audience to which this assertion is addressed. From the perspective of an RP which belongs to one or more audiences A_1,…,An, an assertion is addressed to the RP if at least one of the A_i occur within the <AudienceRestrictionElement>.

NotBefore: This optional attribute identifies the instant in time at which this assertion's validity begins.

NotOnOrAfter: This optional attribute identifies the instant in time at which this assertion's validity becomes false.

4.1.3.1 [ISSUE:CONS-06] Condition Types

The minutes of the F2F call for a reworking of the conditions structure to present a general conditions framework if it can be defended as "well-thought-out". The structure presented here has a clear semantics and allows for future extensibility, via extension of the AbstractConditionType into new types of conditions. It also defines one condition type, audiences; which was the only type specifically required by the F2F minutes.

Does the ConditionsType meet the TC's requirements? If not, why not? Please read ISSUE:[F2F#3-17] and ISSUE:[F2F#3-18] at this point.

4.1.4 Advice

```
<xsd:complexType name="AdviceType">
      <xsd:sequence>
            <xsd:any namespace="##any" processContents="lax" minOccurs="0"
             maxOccurs="unbounded"/>
      </xsd:sequence>
```

```
</xsd:complexType>
```

The optional <Advice> element is an “any” container. Basically you can put any number of arbitrary well-formed XML documents into this container.

4.1.5 Subject Assertion

```
<xsd:complexType name="SubjectAssertionType" abstract="true">
      <xsd:complexContent>
            <xsd:extension base="saml:AssertionType">
                  <xsd:sequence>
                        <xsd:element name="Subject" type="saml:SubjectType"
                         minOccurs="1" maxOccurs="1"/>
                  </xsd:sequence>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>
```

The SubjectAssertionType extends the AssertionType with the addition of a single required element: the <Subject>. Note that SubjectAssertionType is an **abstract** type; it can not be instantiated, it is only useful as a base for inheritance.

4.1.6 Subject

```
<xsd:complexType name="SubjectType">
      <xsd:choice minOccurs="1" maxOccurs="unbounded">
            <xsd:element ref="saml:NameIdentifier" minOccurs="0"
             maxOccurs="unbounded"/>
            <xsd:element ref="saml:Authenticator" minOccurs="0"
             maxOccurs="unbounded"/>
            <xsd:element ref="saml:AssertionSpecifier" minOccurs="0"
             maxOccurs="unbounded"/>
      </xsd:choice>
</xsd:complexType>
```

The <Subject> is a collection of one or more means of identifying the subject of an assertion. The possible means are a <NameIdentifier> element, a <HolderOfKey> element or an <AssertionSpecifier> element. Each element may occur one or more times and should be understood as providing a ``principal’’ or ``description’’ for the subject.

4.1.7 NameIdentifier

```
<xsd:complexType name="NameIdentifierType">
      <xsd:sequence>
            <xsd:element name="SecurityDomain" type="string" minOccurs="1"
             maxOccurs="1"/>
            <xsd:element name="Name" type="string" minOccurs="1"
             maxOccurs="1"/>
      </xsd:sequence>
```

```
</xsd:complexType>
```

The NameIdentifier type represents the identification of a subject as a combination of a name and a security domain.

4.1.7.1 [ISSUE:CONS-07] NameIdentifier Strings

Should the type of the <SecurityDomain> element of a <NameIdentifier> have additional or different structure? This is also addressed in ISSUE:[F2F#3-11] of the [f2f3-minutes].

Should the type of the <Name> element have additional or different structure?

4.1.8 HolderOfKey

```
<complexType name="HolderOfKeyType">
      <sequence>
            <element name="Protocol" type="uriReference"
             maxOccurs="unbounded"/>
            <element name="Authdata" type="string" minOccurs="0"/>
            <element ref="ds:KeyInfo" minOccurs="0"/>
      </sequence>
</complexType>
```

This element specifies one or more <Protocol> elements together an (optional) XML-DSIG <KeyInfo> and/or an (optional) <AuthData> element. The intention here is that the <Protocol> element would describe one or more acceptable authentication techniques such as “urn:protocol:UNIX_PASSWORD_HASH”, “urn:protocol:SSL”, “urn:protocol:XML-DSIG”, etc. The <KeyInfo> element would hold information about the public key (or certificate)—using the structure specified by the XML-DSIG standard—and the <AuthData> element would hold data such as the hash of a password.

4.1.8.1 [ISSUE:CONS-08] Protocol Profile

The TC will develop a namespace identifier (e.g., protocol above) and set of standard namespace specific strings for the <Protocol> element above. If not, what approach should be taken here?

4.1.8.2 [ISSUE:CONS-09] “Bearer” Type

The following proposal has been made for identifying a ``bearer’’ assertion: a distinguished URI urn:protocol:bearer be used as the value of the <Protocol> element in <HolderOfKey> with no other sub-elements. Is this an acceptable design?

4.1.9 AssertionSpecifier

```
<element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>
<xsd:complexType name="AssertionSpecifierType">
      <xsd:choice>
            <xsd:element name="AssertionID" type="saml:IDType" minOccurs="1"
```

```
             maxOccurs="1"/>
            <xsd:element name="Assertion" type="saml:AssertionType"
             minOccurs="1" maxOccurs="1"/>
      </xsd:choice>
</xsd:complexType>
```

This type is used when you want to identify the subject of an assertion by saying “The subject of this assertion is whoever the subject of **the included** assertion is.” You specify the other assertion either by its AssertionID, or by including the other assertion completely. Note that a global element of this type has been declared, so this element can be referenced in other definitions.

4.1.10 Authentication Assertion

```
<complexType name="AuthenticationAssertionType">
      <complexContent>
            <extension base="saml:SubjectAssertionType">
                  <sequence>
                        <element ref="saml:AuthenticationCode"/>
                        <element name="AuthenticationInstant"
                         type="timeInstant"/>
                        <element name="AuthLocale" type="saml:AuthLocaleType"
                         minOccurs="0"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
```

The AuthenticationAssertionType extends the SubjectAssertionType with the addition of two required elements, and an optional one. Note that AuthenticationAssertionType is a **concrete** type and can be instantiated.

The extensions that make up this type are a string that identifies the type of authentication that was used to create the assertion (“AuthenticationCode”), an identifier of the time at which the authentication took place (“AuthenticationInstant”), and an optional advisory element that identifies the DNS domain name and IP address for system entity the authentication (“AuthLocale”).

AuthenticationCode: This is a string that identifies the type of Authentication used to generate the assertion.

AuthenticationInstant: This is the time at which the authentication took place.

4.1.10.1 [ISSUE:CONS-10] AuthenticationCode Profile

What restrictions, if any, should be placed on the format of the contents of the AuthenticationCode element? Should this be a closed list of possible values? Should the list be open, but with some “well-known” values? Should we refer to another list already in existence?

Are the set of values supported for the <Protocol> element ([ISSUE:CONS-08]) essentially the same as those requred for the <AuthenticationCode> element?

4.1.11 AuthLocale

```
<xsd:complexType name="AuthLocaleType">
      <xsd:sequence>
            <xsd:element name="IP" type="string" minOccurs="0"
             maxOccurs="1"/>
            <xsd:element name="Domain" type="string" minOccurs="0"
             maxOccurs="1"/>
      </xsd:sequence>
</xsd:complexType>
```

This optional element contains two optional elements: an identifier of the IP address and DNS domain name of the authenticated system entity. This element is entirely advisory, since both these fields are quite easily "spoofed" but current practice appears to require its inclusion.

4.1.12 Attribute Assertion

```
<complexType name="AttributeAssertionType">
      <complexContent>
            <extension base="saml:SubjectAssertionType">
                  <sequence>
                        <element ref="saml:Attribute" maxOccurs="unbounded"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
```

The AttributeAssertionType extends the SubjectAssertionType with the addition of one or more attributes. Note that AttributeAssertionType is a **concrete** type and can be instantiated.

4.1.13 Attributes

```
<complexType name="AttributeValueType">
      <sequence>
            <any namespace="##any" processContents="lax" minOccurs="0"
             maxOccurs="unbounded"/>
      </sequence>
</complexType>

<element name="Attribute" type="saml:AttributeType"/>

<complexType name="AttributeType">
      <sequence>
            <element name="AttributeName" type="string"/>
            <element name="AttributeNamespace" type="uriReference"
             minOccurs="0"/>
            <element name="AttributeValue" type="saml:AttributeValueType"
             minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
</complexType>
```

The attributes are combinations of an attribute name, and optionally a namespace and one or more values. The <AttributeNamespace> elements qualifies the <AttributeName>. The values are "any" aggregates so that an arbitrary number of well-formed XML documents (one or more) can make up a value.

4.1.14 Authorization Decision Assertions

```
<complexType name="AuthorizationDecisionAssertionType">
      <complexContent>
            <extension base="saml:SubjectAssertionType">
                  <sequence>
                        <element ref="saml:Object"/>
                        <element name="Answer" type="saml:DecisionType"/>
                        <element ref="saml:Evidence" minOccurs="0"
                         maxOccurs="unbounded"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
```

The AuthorizationDecisionAssertionType extends the SubjectAssertionType with the addition of two required elements, and an optional one. Note that AuthorizationDecisionAssertionType is a **concrete** type and can be instantiated.

The required elements are the <Object> of the authorization decision, and the <Answer> (which represents the decision part of the authorization decision). The optional element, <**Evidence>**, is a container of zero or more AssertionSpecifiers (either AssertionIDs, or complete Assertions—see §4.1.3.1.3) that describe assertions provided as evidence for the decision. These evidence assertions can also be interpreted as “This decision is made subject to the assertions in the Evidence element”.

One of the required elements is the <**Answer>**, which is a string of the DecisionType. This type is an enumeration of valid answers to Authorization questions. At this time the set of possible answers is limited to “Permit”, “Deny”, and “Indeterminate” as defined below.

```
<xsd:simpleType name="DecisionType">
      <xsd:restriction base="string">
            <xsd:enumeration value="Permit"/>
            <xsd:enumeration value="Deny"/>
            <xsd:enumeration value="Indeterminate"/>
      </xsd:restriction>
</xsd:simpleType>
```

4.1.14.1 [ISSUE:CONS-11] Authentication Decision Strings

Does {Permit, Deny, Indeterminate} cover the range of decision answers we need? See also discussion in ISSUE:[F2f#3-33].

4.1.15 Object

```
<element name="Object" type="saml:ObjectType"/>
<complexType name="ObjectType">
      <sequence>
            <element name="Resource" type="xsd:uriReference"/>
            <element name="Namespace" type="uriReference" minOccurs="0"/>
            <element name="Action" type="string" maxOccurs="unbounded"/>
      </sequence>
```

```
</complexType>
```

The <Object> element is composed of a uriReference that identifies the resource (<Resource>), an optional namespace reference (<Namespace>), and a list of one or more actions that are relevant to the resource (<Action>). The <Namespace> element qualifies the <Action> element.

Example:

Namespace: xmlns:http-action-namespace

Actions: GET, POST, HEAD

4.1.15.1 [ISSUE:CONS-12] <Action> Element Profile

As part of f2f#3, there was a consensus that some kind of registry of actions and namespaces. This issue is also discussed in ISSUE:[F2F#3-32]. Where should this registry be maintained? There is a further question of whether the SAML specification should call components of this registry, either as part of this specification, or parallel to it (e.g., actions for HTTP, SMTP, J2EE etc.).

4.1.15.2 [ISSUE:CONS-13] Multiple Action Semantics

The f2f#3 left it somewhat unclear if multiple actions are supported within an <Object>. There is clear advantage to this type of extension (as defined in the schema above) as it provides a simple way to aggregate actions. Given that actions are strings (as opposed to pieces of XML) this does seem to provide additional flexibility within the SAML framework.

Does the TC support this type of flexibility?

4.2 Examples

4.2.1 Authentication Assertion Example

This example shows an assertion with a 5 minute lifespan that asserts that the subject (identified by both a NameIdentifier and a KeyInfo block) is in fact “SomeUser” of Example Company.

```
<Assertion xsi:type=”saml:AuthenticationAssertionType”
  version=”http://www.oasis.org/tbs/1066-12-25/1.0”
  AssertionID=”{186CB370-5C81-4716-8F65-F0B4FC4B4A0B}”
  Issuer=”www.example.com”
  IssueInstant=”2001-05-31T13:20:00-05:00”>
      <Conditions
        NotBefore=”2001-05-31T13:20:00-05:00”
        NotOnOrAfter=”2001-05-31T13:25:00-05:00”/>
      <Subject>
            <NameIdentifier>
                  <SecurityDomain>www.example.com</SecurityDomain>
                  <Name>SomeUser</Name>
            </NameIdentifier>
            <Authenticator>
                  <ds:KeyInfo>
                        <KeyValue>
                              <DSAKeyValue>
                                    <P>
/X9TgR11EilS30qcLuzk5/YRt1I870QAwx4/gLZRJmlFXUAiUftZPY1Y+r/F9bow9s
ubVWzXgTuAHTRv8mZgt2uZUKWkn5/oBHsQIsJPu6nX/rfGG/g7V+fGqKYVDwT7g/bT
xR7DAjVUE1oWkTL2dfOuK2HXKu/yIgMZndFIAcc=
                                    </P>
                                    <Q>l2BQjxUjC8yykrmCouuEC/BYHPU=</Q>
                                    <G>
9+GghdabPd7LvKtcNrhXuXmUr7v6OuqC+VdMCz0HgmdRWVeOutRZT+ZxBxCBgLRJFn
Ej6EwoFhO3zwkyjMim4TwWeotUfI0o4KOuHiuzpnWRbqN/C/ohNWLx+2J6ASQ7zKTx
vqhRkImog9/hWuWfBpKLZl6Ae1UlZAFMO/7PSSo=
                                    </G>
                                    <Y>
i5/D5JhXm/ZbA+ivdGTdqrrAu/HHkiMDit6J1/KFJLKkTidMzM5xJADzxw6Tj+mKji
+fJee5EHlQF90a7apwYTxpE6JZN8BMhOu8zw6wFEhRg4xQBUerV0fRPkeN5PpyioN6
RvbHftp/ITUlqN9N53lVTWdc9CHYat6PuOtfTWA=
                                    </Y>
                              </DSAKeyValue>
                        </KeyValue>
                        <X509Data>
                              <X509SubjectName>
                                    CN=SomeUser, OU=Some Group,
                                    O=Example, L=SomeCity, ST=SomeState,
                                    C=SomeCountry
                              </X509SubjectName>
                              <X509Certificate>
MIIDMTCCAu8CBDqIR9gwCwYHKoZIzjgEAwUAMH4xCzAJBgNVBAYTAlVTMRYwFAYDVQQIEw1NYXNz
YWNodXNldHRzMRAwDgYDVQQHEwdNZXRodWVuMRIwEAYDVQQKEwlOZXRlZ3JpdHkxGTAXBgNVBAsT
EEIyQiBBZ2VudHMgR3JvdXAxFjAUBgNVBAMTDVJvYmVydCBUYXlsb3IwHhcNMDEwMjEyMjAzMDE2
WhcNMDEwNTEzMjAzMDE2WjB+MQswCQYDVQQGEwJVUzEWMBQGA1UECBMNTWFzc2FjaHVzZXR0czEQ
MA4GA1UEBxMHTWV0aHVlbjESMBAGA1UEChMJTmV0ZWdyaXR5MRkwFwYDVQQLExBCMkIgQWdlbnRz
IEdyb3VwMRYwFAYDVQQDEw1Sb2JlcnQgVGF5bG9yMIIBuDCCASwGByqGSM44BAEwggEfAoGBAP1/
U4EddRIpUt9KnC7s5Of2EbdSPO9EAMMeP4C2USZpRV1AIlH7WT2NWPq/xfW6MPbLm1Vs14E7gB00
b/JmYLdrmVClpJ+f6AR7ECLCT7up1/63xhv4O1fnxqimFQ8E+4P208UewwI1VBNaFpEy9nXzrith
1yrv8iIDGZ3RSAHHAhUAl2BQjxUjC8yykrmCouuEC/BYHPUCgYEA9+GghdabPd7LvKtcNrhXuXmU
r7v6OuqC+VdMCz0HgmdRWVeOutRZT+ZxBxCBgLRJFnEj6EwoFhO3zwkyjMim4TwWeotUfI0o4KOu
HiuzpnWRbqN/C/ohNWLx+2J6ASQ7zKTxvqhRkImog9/hWuWfBpKLZl6Ae1UlZAFMO/7PSSoDgYUA
AoGBAIufw+SYV5v2WwPor3Rk3aq6wLvxx5IjA4reidfyhSSypE4nTMzOcSQA88cOk4/pio4vnyXn
uRB5UBfdGu2qcGE8aROiWTfATITrvM8OsBRIUYOMUAVHq1dH0T5HjeT6coqDekb2x37afyE1Jajf
Ted5VU1nXPQh2Grej7jrX01gMAsGByqGSM44BAMFAAMvADAsAhRy+2AJp8ZZ8OVSe02TsjZ21p0W
BQIUOvsjuK7l5yd7l5WvjEmP+MVzSJg=
                              </X509Certificate>
                        </X509Data>
                  </ds:KeyInfo>
            </Authenticator>
      </Subject>
      <AuthenticationType>X509v3</AuthenticationType>
      <AuthenticationInstant>2001-05-31T13:20:00-05:00
      </AuthenticationInstant>
</Assertion>
```

4.2.2 Attribute Assertion Example

This example illustrates the use of an attribute assertion to assign some attributes to a user. This example has a fictitious consortium assigning a credit summary to a given subject. Note that the value of the attribute is a block of arbitrary XML, presumably following the schema specified by the attribute namespace.

```
<Assertion xsi:type="saml:AttributeAssertionType"
  version="0100"
  AssertionID="{EE52CAF4-3452-4ebe-84D3-4D372C892A5D}"
  Issuer="www.example.com"
  IssueInstant="2001-05-31T13:20:00-05:00">
      <Conditions
        NotBefore="2001-05-31T13:20:00-05:00"
        NotOnOrAfter="2001-05-31T13:25:00-05:00">
      </Conditions>
      <Subject>
            <NameIdentifier>
                  <SecurityDomain>www.example.com</SecurityDomain>
                  <Name> cn=SomeUser,ou=finance,co=example </Name>
            </NameIdentifier>
      </Subject>
      <Attribute>
            <AttributeName>NetWorthSummary</AttributeName>
            <AttributeNamespace>
                  http://ns.finance-vocab.org/finance
            </AttributeNamespace>
            <AttributeValue>
                  <CreditSummary>
                        <HistoryScore>Excellent</HistoryScore>
                        <CurrentAssets>Loaded</CurrentAsserts>
                  </CreditSummary>
            </AttributeValue>
      </Attribute>
</Assertion>
```

4.2.3 Authorization Decision Example

This example shows the result of a credit check, for a given subject. Note that the above attribute assertion is given as evidence.

```
<Assertion xsi:type="saml:AuthorizationDecisionAssertionType"
  version="0100"
  AssertionID="{5CFCA396-C2AC-497c-975F-233CDC69CFE4}"
  Issuer="www.example.com"
  IssueInstant="2001-05-31T13:20:00-05:00">
      <Conditions
        NotBefore="2001-05-31T13:20:00-05:00"
        NotOnOrAfter="2001-05-31T13:25:00-05:00">
            <Condition xsi:type="saml:AudienceRestrictionConditionType">
                  <Audience>
                        http://www.example.com/agreements/credit.html
                  </Audience>
            </Condition>
      </Conditions>
      <Subject>
```

```
            <NameIdentifier>
                  <SecurityDomain>us-staff</SecurityDomain>
                  <Name>cn=SomeUser,ou=finance,co=example</Name>
            </NameIdentifier>
      </Subject>
      <Object>
            <Resource>
                  credit:CheckCredit
            </Resource>
            <Action>
                  Amount=5000&Currency=USD
            </Action>
            <Namespace>
                  credit=http://ns.finance-vocab.org/finance
            </Namespace>
      </Object>
      <Answer>Permit</Answer>
      <Evidence>
            <AssertionID>{EE52CAF4-3452-4ebe-84D3-4D372C892A5D}</AssertionID>
      </Evidence>
</Assertion>
```