AuthXML: A Specification for Authentication Information In XML

Second Draft - 11/22/00

Authors: Evan Prodromou, Outlook
Darren Platt, Securant
Robert L. Grzywinski, Outlook
Eric Olden, Securant

1 Overview

This document describes AuthXML, a specification for authentication and authorization information in XML. AuthXML is a transport-independent XML definition that allows security authorities in separate organizations to communicate about authentication, authorization, user profiles and authenticated user sessions in an open way.

1.1 Rationale

The expanded use of secured networked applications, within enterprises and between them, has led to increased complexity for users and administrators. Users are often required to make multiple logons to different applications in different security domains.

Some solutions for reducing the security complexity for users have been proposed and implemented. Generally they are monolithic, requiring a single, authoritative user database which all other databases and applications must obey. Some solutions use a distributed model, with trust between domains, but these are usually proprietary or ad hoc.

The purpose of the AuthXML standard is to provide an open framework for resource realms, such as applications and Web sites, to trust security domains. It two key technologies to ensure secure, open implementations:

- **XML**, the EXtensible Markup Language, an open standard for language definitions. The wide usage of XML and variety of XML processors allows for a variety of implementations of the AuthXML standard.
- **Digital Signatures** (XML Signature), a standard for securely verifying the origins of messages. The XML Signature specification allows for XML documents to be signed in a standard way, with a variety of different digital signature algorithms. Digital signatures can be used for validation of messages and for non-repudiation.

AuthXML is a flexible framework, requiring a minimum of functionality from implementations to meet the standard, while allowing maximum extensibility.

1.2 Terms Used in This Document

The following terms are used in this document to describe concepts in the domain of authentication and security. They are defined here for clarity.

- **Principal:** A person or process identified with a security account. Principals are granted access and privileges to resources. When the principal is a human being, this document refers to the principal as "user."
- **Session:** A period for which a principal has been authenticated. It generally extends from when the principal provides authentication credentials, such as a password or digital certificate, to when the principal chooses to end the session (logout). The session may also end when an authority decides (timeout).
- **Domain:** A collection of principal accounts managed together, not necessarily related to a DNS domain name. Examples would be an organization, a government agency, a public Web site or an application within an enterprise.
- **Role:** A collection of principal accounts managed within a domain. Accounts can be grouped by criteria external to the application space (e.g., by organizational role as "engineers," by location as "US residents," etc.), or internal to the space (e.g., by role played in an application, such as "administrator"). There is some difference between a "role" and a "group," but they can be used interchangeably in this document.
- **Resource:** Something that is protected using an automated security mechanism, such as a file, a program, a data object, an object method, or an action.
- **Realm:** A collection of resources that are protected by a single security mechanism. Examples: a Web site, a portion of a Web site, a company's network.
- **Authority**: An entity (such as a program) that determines access to one or more realms, or that provides information on one or more domains.
- **Trust:** A realm trusts a domain when it allows the authority for that domain to authenticate principals for access to the realm.

1.3 Goals

- AuthXML should define a message format for passing authentication, authorization, and user profile information between security domains and between domains and resource realms.
- AuthXML is written in XML.
- AuthXML should not be dependent on any particular security or user database format.
- AuthXML should be easily extensible.

1.4 Non-Goals

- No provision is made for negotiation between authorities about trust between domains and realms or the inclusion of optional data. Trust negotiations must be made out-of-band from the AuthXML conversation.
- No specification is made for the transport mechanism for AuthXML messages.
- No specification is made for protecting AuthXML messages from interception by third parties. This is left up to the transport mechanism of choice between authorities.
- No specification is made for providing permission definitions (policies) or authorization information (policy) through AuthXML. It is assumed that realm authorities have authorization policies and tools to specify permissions.

2 AuthXML Message Exchange Model

This section provides a high-level description of the model used for AuthXML messages.

2.1.1 Message Types

AuthXML is based on a message-passing model. Authorities in a security agreement send *request* messages to query partners for information on a given resource, principal, or session. The partner returns *response* messages containing the requested information. In addition, the partners can send *notification* messages that do not require a response.

The message types defined by AuthXML are:

- **Access-Request**: sent from a security domain to a resource realm requesting access for a principal to a resource.
- **Access-Response:** sent in response to an Access-Request from a resource realm, granting or denying access to a resource.
- **Session-Request:** sent from a realm to a domain requesting information about a session.
- **Session-Response:** sent from a domain to a realm in response to a Session-Request, providing information about a session.
- **Principal-Request:** sent from a realm to a domain requesting information about a principal, including the principal's profile.
- **Principal-Response:** sent from a domain to a realm in response to a Session-Request, providing information about a session.
- **Session-Notification:** a notification message, sent by either domain or realm, to update a partner about changes in state in the principal's session.

More detailed definition of these message types are provided in section 3.2 of this document.

2.1.2 Entity Types

Each message contains information due to the type of the message. In addition, messages contain *entities* that define the objects that the message refers to. For example, a Session-Request message contains a session entity for which the realm requires further information.

Entities used in AuthXML messages include the following:

- **Principal**
- **Session**
- **Profile**
- **Extension**

Each of these entities represents the corresponding security concept defined in section 1.2 of this document. Extensibility of the AuthXML protocol is provided for by the Extension entity, which can contain arbitrary data.

Each entity type can contain a minimum amount of data -- simply enough to reference the entity -- or a full description of the entity. In general, request messages contain reference-level entities, while response messages contain full descriptions of the entity.

More detailed specifications of these entities is made in section 3.1.

2.1.3 Digital Signature

Every message in the AuthXML protocol must be signed with a digital signature per the XML Signature specification. The signature algorithm and signature certificates are defined by security partners out of band from the AuthXML conversation. No exchange of digital keys is defined in AuthXML.

3 Data Types

This section describes the XML data types that define the AuthXML standard. An XML Schema definition for AuthXML is provided in Appendix A.

3.1 Entity Types

Entities are objects that are passed in XML format between authorities. They are the "nouns" of the AuthXML space.

3.1.1 Principal

This entity defines a principal account. Each principal is identified uniquely by the combination of principal ID and domain.

3.1.1.1 Fields

Name	*Type*	*Req?*	*Description*
Id	String	Yes	A unique identifier for the principal.
Domain	String	Yes	The security domain that contains the principal's account.
Profile	Profile	No	Additional information about the principal.
Extension	Extension	No	Extension data about the principal.

3.1.1.2 Notes

- The ID field can be a free-form string, encompassing user ID numbers ("110004"), alphanumeric user Ids ("aaronl"), as well as more complicated identifications such as an LDAP distinguished name.
- The domain field provides a namespace for users within a security domain.
- Realm authorities MUST be able to determine the security authority for a principal based on the domain.
- The ID field should be sufficient to define a single principal account. The security authority should be able to guarantee that a principal's ID is unique within the domain.
- The profile field is optional for optimization purposes.
- In general, persistent extension information about a principal should be encoded in an extension to the profile rather than as an extension to the principal. However, the field is provided for occasions when the "optimized," shorter version of a principal, without a profile, is used.

3.1.1.3 Example

A user "aaronl" in security domain "acme.com" might be represented with the following principal entity:

```
<principal id="aaronl" domain="acme.com">
      <profile>
            <role>users</role>
            <role>engineers</role>
            <field name="last name">Lawrence</field>
            <field name="first name">Aaron</field>
      </profile>
</principal>
```

An American citizen with a given social security number might have the following definition:

```
<principal id="333-33-3333" domain="Social Security" />
```

A company with multiple local security domains may choose to present a single security interface to the external world. If user IDs are unique within local domains, but not within the entire company, a good format for managing principals might be to include the local domain name with the ID. For example, user

"aaronl" at the Toronto office (with its own local security domain) would be identified as:

```
<principal id="aaronl/toronto" domain="acme.com" />
```

A user with an identical user name in the Sydney office would be identified as:

```
<principal id="aaronl/sydney" domain="acme.com" />
```

Note that the domain field is the same, so that realm authorities can determine the (single) domain authority to use for the principal.

3.1.2 Session

A *session* entity represents an authentication session.

3.1.2.1 Fields

Name	*Type*	*Req?*	*Description*
Id	String	Yes	A unique identifier for the session.
Principal	Principal	Yes	The principal that was authenticated.
Status	String	No	The status of the session.
Authentication	Authentication	No	Authentication type and time for the session.
TouchTime	DateTime	No	The last time a resource was known to be accessed by a principal.
Extension	Extension	No	One or more extension objects that pertain to the session.

3.1.2.2 Notes

- An authentication authority MUST provide a unique identifier for a session. The identifier MUST be unique for a principal in a domain, and the identifier SHOULD be unique for a domain.
- Session ID, principal ID, and domain name should be sufficient to define a session.
- The status field shows the current status of the session. It can hold one of three values:
  - "active" - the session is active.
  - "logout" - the principal has terminated or "logged out" of the session.
  - "timeout" - the authenticating authority has cancelled the session.
- Extensions can be added to the Session entity to encode transient, per-session data. For example, data about an e-commerce transaction could be encoded in an extension. Extension data about the principal proper should be added to the profile (see below).

3.1.2.3 Example

The following XML fragment would represent an authentication session for user "aaronl" from domain "acme.com". Note that an extension determining a hypothetical transaction state is added, and that the extension uses the "acme" namespace.

```
<session id="2001-11-14-0976540">
      <principal id="aaronl" domain="acme.com" />
      <authentication>
            <type>password</type>
            <time>2001-11-14T10:47:03.000-800</time>
      </authentication>
      <touch-time>2001-11-14T12:33:00.000-800</touch-time>
      <extension type="acme.com transaction state"
            xmlns:acme="http://www.acme.com/tran">
            <acme:transaction-state acme:state="paid">
                  <acme:product>Widget</acme:product>
                  <acme:quantity>2</acme:quantity>
            </acme:transaction-state>
      </extension>
</session>
```

3.1.3 Authentication

This object represents an authentication event for a session. Sessions may have one or more authentication events.

3.1.3.1 Fields

Name	*Type*	*Req?*	*Description*
Type	String	Yes	The type of credentials presented when the authentication was performed.
Time	DateTime	Yes	The time of the authentication event.

3.1.3.2 Notes

- AuthenticationType is an open field that can take any value. However, a more formal set of authentication types may be added to this specification at a future date. Suggested values include the following:
  - "password" - the principal provided a password or passphrase to authenticate.
  - "certificate" - the principal provided an encryption certificate to authenticate.

3.1.3.3 Example

The following authentication entity represents a password authentication event for a user.

```
      <authentication>
```

```
        <type>password</type>
        <time>2001-11-14T10:47:03.000-800</time>
    </authentication>
```

3.1.4 Profile

A profile provides detailed information about a principal account. This can include personal or organizational data about the user, as well as groups or roles that the user is a member of within the security domain.

3.1.4.1 Fields

Name	*Type*	*Req?*	*Description*
Role	String	No	One or more groups within a domain that the principal belongs to. Alternately, one or more roles that the user plays in a domain.
Field	String	No	An open data field about the principal.
Extension	Extension	No	Extension data about the principal.

3.1.4.2 Notes

- It's possible to provide simple data about a principal using the "field" field. More complex data about a principal, or data that must conform to a given schema, should be provided in an extension instead. For example, DSML data about a user could be encoded as an extension.
- It's possible to provide both fields and extensions in a single profile entity.
- A realm authority may use groups or roles to determine the principal's access to resources within that realm.
- Extension elements should be used for persistent data about a principal that exists between sessions. Transient data, such as a transaction information, should be added as extensions to the session element.

3.1.4.3 Example

A user profile in X.509 format could be defined with this XML fragment:

```
    <profile>
        <role>engineers</role>
        <field name="full name">Aaron Lawrence</field>
        <extension type="X.509 fields"
              xmlns="http://www.acme.com/x509">
          <name>Aaron Lawrence</name>
          <organization>Acme Corporation</organization>
        </extension>
    </profile>
```

3.1.5 Extension

The extension type provides a standard way for partners in a security agreement to share additional data about users, sessions, profiles and messages beyond the data provided in this document.

3.1.5.1 Fields

Name	*Type*	*Required?*	*Description*
Type	String	Yes	A code indicating the type of the extension.

3.1.5.2 Notes

- The partners in a security agreement define the meaning of the type of an extension. This attribute simply provides an easy way to determine whether an extension is meaningful to the authority that receives the message.
- It is recommended that extensions use a separate XML Namespace to define elements within the extension.

3.1.5.3 Example

The following example shows an extension for describing a user session. It provides information on a user's current transaction state on an e-commerce Web site. Note that the extension type is not meaningful except for identifying the extension. Also note that an XML Namespace is provided.

```
<extension type="acme.com transaction state"
      xmlns:acme="http://www.acme.com/tran">
      <acme:transaction-state acme:state="paid">
            <acme:product>Widget</acme:product>
            <acme:quantity>2</acme:quantity>
      </acme:transaction-state>
</extension>
```

3.2 Message Types

Messages are sent between authorities to request information about principals and session, to request access to resources, to modify principal information, and to provide notification of changes to sessions.

3.2.1 Common Fields

Due to the sensitive nature of authentication data, AuthXML messages MUST provide digital signatures to verify authenticity of the message, according to the XML Signature specification.

Regardless of transport, signatures MUST be included in the same document as the message.

3.2.2 Access-Request

This message represents a request by a peer to allow an authenticated user to access a resource. It is used primarily in a "push" model for authentication sharing (see below). A resource-request is typically sent from a domain authority to a realm authority.

3.2.2.1 Fields

Name	*Type*	*Required?*	*Description*
Session	Session	Yes	The authenticated session for the principal who is requesting access.
Realm	String	Yes	The security realm for which access is requested.
Resource	String	No	The resource for which access is requested.
Mapped-Principal	Principal	No	If the requesting peer maintains a database that maps local principals to a more definitive security domain for the realm, it may provide information for how the principals are mapped.
Extension	Extension	No	Extended information about the session or principal.

3.2.2.2 Notes

- Mapping data between principal accounts in the authenticating domain and in the authoritative domain for a security realm can be stored either bye the authenticating domain authority or with the realm authority. If it's stored by the authenticating domain authority, that domain should pass a mapped-principal to the realm authority.
- The resource that is requested can be in any format, but must be meaningful for the realm authority. Examples of requested resources can be URLs, class + method pairs ("Account.withdraw"), named applications ("Calendar") or named actions ("Download Latest Files").

3.2.2.3 Example

This example shows a request where the

```
<access-request>
      <session id="2001-11-14-00858451">
            <principal id="aaronl" domain="acme.com" />
      </session>
      <realm>widget.com</realm>
      <resource>http://www.widget.com/private/</resource>
      <mapped-principal>
            <principal id="Aaron_Lawrence" domain="widget.com" />
      </mapped-principal>
</access-request>
```

3.2.3 Access-Response

This message is sent by a realm authority to a domain authority in response to a access-*request* message. It provides information on the success or failure of the request.

3.2.3.1 Fields

Name	*Type*	*Required?*	*Description*
SuccessCode	String	Yes	The results of the request.
AccessToken	String	No	If the result was successful, this field should hold a token that the principal or a program can use for accessing the resource.
Extension	Extension	No	Extended information about the access.

3.2.3.2 Notes

- The success code field has no defined format. The meaning and content of the code is the responsibility of the domain and realm authorities. Sample success codes could be "0", "1", "true", "yes", "continue", "error", etc. However, the code should *not* contain any per-request data.
- The AccessToken contains data that can be used to continue the principal's access to the resource. For example, it could contain an URL for login, a serialized security ticket, or other data. The nature of the access token, and how the domain authority uses it, is implementation- and agreement-dependent.
- The extension data can contain further information about the access. For example, for a restricted resource it may contain billing information.

3.2.3.3 Example

An example message in response to a resource-request might be the following. It shows a sample success code, and a sample access token (in this case, the access token is an URL that the principal can be redirected to):

```
<access-response>
      <success>true</success>
      <access-token>
      http://www.widget.com/login?sid=35431
  </access-token>
</access-response>
```

3.2.4 Session-Request

A session request is a request for further information about an authentication session. It is typically sent from a realm authority to a domain authority in a "pull" authentication model.

3.2.4.1 Fields

Name	*Type*	*Required?*	*Description*
Session	Session	Yes	The authenticated session for which information is requested.
Extension	Extension	No	Extended information about the request.

3.2.4.2 Notes

- The passed-in session can be of any format, including full principal profile, etc. However, the most common use of this message will pass in only the fields necessary to uniquely identify a session (session ID, principal, and domain), and retrieve a more fully defined session (with authentication times, profile, extensions, etc.)

3.2.4.3 Example

The following example shows a typical session-request.

```
<session-request>
      <session id="2000-14-11-0965431">
            <principal id="aaronl" domain="acme.com" />
      </session>
</session-request>
```

3.2.5 Session-Response

This message is sent in response to a session-request message. It is typically sent from a domain authority to a realm authority in a "pull" model for authentication.

3.2.5.1 Fields

Name	*Type*	*Required?*	*Description*
Success-Code	String	Yes	Indicator of the success or failure of the request.
Session	Session	No	The authenticated session for which information is requested.
Extension	Extension	No	Extension for the response.

3.2.5.2 Notes

- As with other requests, the success-code field is implementation- and agreement-dependent.
- Typically, the session data will be a more detailed version of the session that was passed in.

3.2.5.3 Example

An example session-response is shown below.

```
<session-response>
      <success-code>true</success-code>
      <session id="2001-11-14-0976540">
            <principal id="aaronl" domain="acme.com" />
            <authentication>
                  <type>password</type>
                  <time>2001-11-14T10:47:03.000-800</time>
        </authentication>
            <touch-time>
                  2001-11-14T12:33:00.000-800
            </touch-time>
      </session>
</session-response>
```

3.2.6 Principal-Request

A principal-request message is sent to request additional information about a principal.

3.2.6.1 Fields

Name	*Type*	*Required?*	*Description*
Principal	Principal	Yes	The principal for which information is requested.
Extension	Extension	No	Extended information about the request.

3.2.6.2 Notes

- The principal field can be as detailed as desired. However, typically the principal will only contain necessary fields to identify the principal (id and domain), with the expectation that more detailed data will be returned in the response.

3.2.6.3 Example

An example principal-request follows.

```
      <principal-request>
            <principal id="aaronl" domain="acme.com" />
      </principal-request>
```

3.2.7 Principal-Response

A principal-response is sent by a domain authority in response to a principal-request. It gives further information about a principal.

3.2.7.1 Fields

Name	*Type*	*Required?*	*Description*
Success-Code	String	Yes	A code indicating the success of the operation.
Principal	Principal	No	The principal for which information was requested.
Extension	Extension	No	Extended information about the response.

3.2.7.2 Notes

- As with other responses, there is no set value for the success-code field.

3.2.7.3 Example

```
<principal-response>
      <success-code>true</success-code>
      <principal id="aaronl" domain="acme.com">
            <profile>
                  <role>engineers</role>
                  <field name="full name">Aaron Lawrence</field>
                  <extension type="X.509 fields"
                        xmlns="http://www.acme.com/x509">
                  <name>Aaron Lawrence</name>
                  <organization>Acme Corporation</organization>
            </extension>
            </profile>
      </principal>
</principal-response>
```

3.2.8 Session-Notification

There are some circumstances where domain authorities may need to keep realm authorities updated on the status of a session. They may send out session-notification messages to notify authorities of changes to the session. In addition, realm authorities may send session-notifications to a domain authority, to give information about resources touched.

3.2.8.1 Fields

Name	*Type*	*Required?*	*Description*
Session	Session	Yes	A code indicating the success of the operation.
Extension	Extension	No	Extended information about the response.

3.2.8.2 Notes

- The session, its enclosed principal, and the profile may have as much or as little data as needed, per agreement by the authorities. However, in general the session will normally just have the minimum amount of data for identification, plus status fields (status, authentication-time, touch-time).

3.2.8.3 Examples

The following session-notification may be sent from a domain authority to indicate that the session has been terminated:

```
<session-notification>
      <session id="2000-14-11-564091">
            <principal id="aaronl" domain="acme.com" />
            <status>logout</status>
      </session>
</session-notification>
```

A session-notification sent from a realm authority to notify the domain authority of a recently visited resource might look like:

```
<session-notification>
      <session id="2000-14-11-564091">
            <principal id="aaronl" domain="acme.com" />
            <touch-time>
                  Mon, 14 Nov 2001 12:33:03 PST
            </touch-time>
            <touch-resource>
                  Widget.com calendar program
            </touch-resource>
  </session>
</session>
```

4 XML Namespace

The XML Namespace for AuthXML messages is:

http://www.authxml.org/authxml/1.0/

5 Appendix A: XML Schema for AuthXML

The following XML Schema defines the format for AuthXML.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2000/08/XMLSchema">

  <xsd:complexType name="extension" mixed="true">
      <xsd:attribute name="type" type="xsd:string" use="required" />
</xsd:complexType>

  <xsd:complexType name="principal">
    <xsd:attribute name="id" type="xsd:string" use="required" />
    <xsd:attribute name="domain" type="xsd:string" use="required" />
    <xsd:sequence>
      <xsd:element name="profile" type="profile"
```

```
            minOccurs="0" maxOccurs="1" />
      <xsd:element name="extension" type="extension"
            minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name=”authentication”>
    <xsd:sequence>
        <xsd:element name=”type” type=”xsd:string” minOccurs=”1”
             maxOccurs=”1” />
        <xsd:element name=”time” type=”xsd:timeInstant” minOccurs=”1”
             maxOccurs=”1” />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="session">
    <xsd:attribute name="id" type="xsd:string" use="required" />
    <xsd:sequence>
        <xsd:element name="principal" type="principal" minOccurs="1"
maxOccurs="1" />
        <xsd:element name="status" type="xsd:string"
            minOccurs="0" maxOccurs="1" />
      <xsd:element name="authentication" type="authentication"
            minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="touch-time" type="xsd:timeInstant"
            minOccurs="0" maxOccurs="1" />
      <xsd:element name="touch-resource" type="xsd:string"
            minOccurs="0" maxOccurs="1" />
      <xsd:element name="extension" type="extension"
            minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="field">
     <xsd:simpleContent>
        <xsd:extension base="xsd:string">
          <xsd:attribute name="name" type="xsd:string" use="required" />
      </xsd:extension>
     </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="profile">
    <xsd:sequence>
      <xsd:element name="role" type="xsd:string"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="field" type="field"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="extension" type="extension"
        minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
```

```
<xsd:element name="access-request">
  <xsd:sequence>
    <xsd:element name="session" type="session"
      minOccurs="1" maxOccurs="1" />
    <xsd:element name="realm" type="xsd:string"
      minOccurs="1" maxOccurs="1" />
    <xsd:element name="resource" type="xsd:string"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="mapped-principal" type="principal"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="extension" type="extension"
      minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:element>

<xsd:element name="access-response">
  <xsd:sequence>
    <xsd:element name="success-code" type="xsd:string"
      minOccurs="1" maxOccurs="1" />
    <xsd:element name="access-token" type="xsd:string"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="extension" type="extension"
      minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:element>

<xsd:element name="session-request">
  <xsd:sequence>
    <xsd:element name="session" type="session"
      minOccurs="1" maxOccurs="1" />
    <xsd:element name="extension" type="extension"
      minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:element>

<xsd:element name="session-response">
  <xsd:sequence>
    <xsd:element name="success-code" type="xsd:string"
      minOccurs="1" maxOccurs="1" />
    <xsd:element name="session" type="session"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="extension" type="extension"
      minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:element>

<xsd:element name="principal-request">
  <xsd:sequence>
    <xsd:element name="principal" type="principal"
      minOccurs="1" maxOccurs="1" />
```

```
      <xsd:element name="extension" type="extension"
        minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:element>

  <xsd:element name="principal-response">
    <xsd:sequence>
      <xsd:element name="success-code" type="xsd:string"
        minOccurs="1" maxOccurs="1" />
      <xsd:element name="principal" type="principal"
        minOccurs="0" maxOccurs="1" />
      <xsd:element name="extension" type="extension"
        minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:element>

  <xsd:element name="session-notification">
    <xsd:sequence>
      <xsd:element name="session" type="session"
        minOccurs="1" maxOccurs="1" />
      <xsd:element name="extension" type="extension"
        minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:element>

</xsd:schema>
```

6 Appendix B: Authentication Conversation Examples

The following examples provide some illustrative examples for a series of message exchanges between authorities.

6.1 Push Model

In this model, the authority for a security domain sends an access request to the authority for a security realm directly. The domain authority will generally give sufficient information for the realm authority to make decisions about granting access, such as role membership.

```
      <access-request>
            <realm>widget.com</realm>
            <resource>http://www.widget.com/private/</resource>
            <session id="2000-11-14-5409194">
                  <principal id="aaronl" domain="acme.com">
                        <profile>
                              <role>users</role>
                              <role>engineers</role>
                        </profile>
                  </principal>
            </session>
```

```
</access-request>
```

The realm authority will determine if access is granted, and return an access-response. If so, it will provide an access token that the domain authority can pass to a user agent, such as a browser, to use for accessing the resource. For example, it may create a local session (with a local session ID), that would be a proxy for the remote session.

```
<access-response>
          <success-code>true</success-code>
          <access-token>
                http://www.widget.com/private/?sid=102438
          </access-token>
</access-response>
```

The domain authority would keep the realm authority updated to changes in the session status, such as logout:

```
<session-notification>
      <session id="2000-11-14-5409194">
            <principal id="aaronl" domain="acme.com" />
            <status>logout</status>
      </session>
</session-notification>
```

The realm authority can act on the session notification by cancelling the proxy session, or by some other action.

6.2 Pull Model

In this model, the realm authority receives information about a session by some other means besides AuthXML. For example, it may receive HTTP parameters that define the session id, principal id, and domain. Alternately, it may receive method parameters from a remote object protocol. It would use these parameters to construct a session-request message to request information about the session and principal, and send the request to the domain authority:

```
<session-request>
      <session id="2000-14-11-2210418">
            <principal id="aaronl" domain="acme.com" />
      </session>
</session-request>
```

The domain authority would provide information about the session and principal in a session-response message. It should give sufficient information for the realm authority to make authorization decisions.

```
<session-response>
```

```
        <success-code>true</success-code>
        <session id="2000-14-11-0965431">
              <principal id="aaronl" domain="acme.com">
                    <profile>
                          <role>users</role>
                          <role>engineers</role>
                    </profile>
              <status>active</status>
              <authentication>
                    <type>password</type>
                    <time>2001-11-14T10:47:03.000-800</time>
              </authentication>
              <touch-time>
                    2001-11-14T12:33:00.000-800
              </touch-time>
        </session>
  </session-response>
```

The realm authority may make additional requests about the session or principal to aid in making access decisions, or to provide personalization or other customization for the principal.

7 Appendix C: Transport Recommendations

This section suggests some possible implementations of an AuthXML session using various currently-used Internet protocols.

7.1 Hypertext Transfer Protocol (HTTP)

For an HTTP conversation, the sender of a request or notification submits an HTTP Request with method POST to the designated receiver of the message. The Content-type of the request is “text/xml,” and the body of the request is the XML content of the AuthXML message. For example:

```
POST HTTP/1.1 http://www.widget.com/servlets/AuthXMLServlet
Content-Type: text/xml
Date: Mon, Nov 14 2000 12:35:22 PST -800

<!-- digital signature data -->
<access-request>
      <realm>widget.com</realm>
      <resource>http://www.widget.com/private/</resource>
      <session id="2000-11-14-5409194">
            <principal id="aaronl" domain="acme.com">
                  <profile>
                        <role>users</role>
                        <role>engineers</role>
                  </profile>
            </principal>
      </session>
```

```
</access-request>
<!-- digital signature data -->
```

Access-Request, Session-Request, Principal-Request and Session-Notification messages should be sent as HTTP Requests.

Response messages should be send as an HTTP Response to the HTTP Request containing the original -request message. They should have a text/xml content type, and should contain the AuthXML message as the body of the response. An example:

```
200 HTTP/1.1 OK
Content-Type: text/xml
Date: Mon, Nov 14 2000 12:38:44 PST -800

<!-- digital signature data -->
<principal-response>
      <success-code>true</success-code>
      <principal id="aaronl" domain="acme.com">
            <profile>
                  <role>engineers</role>
                  <field name="full name">Aaron Lawrence</field>
                  <extension type="X.509 fields"
                        xmlns="http://www.acme.com/x509">
                        <name>Aaron Lawrence</name>
                        <organization>Acme Corporation</organization>
                  </extension>
            </profile>
      </principal>
</principal-response>
<!-- digital signature data -->
```

HTTP Responses should be formed for Access-Response, Principal-Response, and Session-Response messages. Note that no standard response is necessary for a Session-Notification message, except the HTTP status code specifying that the message was received.

Due to the sensitive nature of AuthXML messages, it is suggested that secure HTTP channels, such as HTTPS, be used. However, the use of HTTPS does not supercede the requirement for all AuthXML messages to be signed using XML Signature.

7.2 Simple Object Access Protocol (SOAP)

SOAP is itself a transport-independent protocol. AuthXML messages fit into the SOAP scheme as the body of simple message type. They should be enclosed in SOAP envelopes as with any SOAP message. Note that the digital signature should also be included in the SOAP message body.

AuthXML follows the SOAP request-response model for message interchange.

8 References

- Extensible Markup Language (XML): http://www.w3.org/XML/
- XML Namespaces: http://www.w3.org/REC-xml-names
- XML Signature: http://www.w3.org/Signature/
- HTTP: ftp://ftp.isi.edu/in-notes/rfc2616.txt
- SOAP: http://www.w3.org/TR/SOAP/
- XML Schema: http://www.w3.org/XML/Schema