Liberty Protocols and Schemas Specification

Version 1.0

11 July 2002

Document Description: liberty-architecture-protocols-schemas-v1.0
Notice

Copyright © 2002 ActivCard; American Express Travel Related Services; America Online, Inc.; Bank of America; Bell Canada; Catavault; Cingular Wireless; Cisco Systems, Inc.; Citigroup; Cyberun Corporation; Deloitte & Touche LLP; EarthLink, Inc.; Electronic Data Systems, Inc.; Entrust, Inc.; Ericsson; Fidelity Investments; France Telecom; Gemplus; General Motors; Hewlett-Packard Company; i2 Technologies, Inc.; Intuit Inc.; MasterCard International; Nextel Communications; Nippon Telegraph and Telephone Company; Nokia Corporation; Novell, Inc.; NTT DoCoMo, Inc.; OneName Corporation; Openwave Systems Inc.; PricewaterhouseCoopers LLP; Register.com; RSA Security Inc; Sabre Holdings Corporation; SAP AG; SchlumbergerSema; Sony Corporation; Sun Microsystems, Inc.; United Airlines; VeriSign, Inc.; Visa International; Vodafone Group Plc; Wave Systems. All rights reserved.

This Specification has been prepared by Sponsors of the Liberty Alliance. Permission is hereby granted to use the Specification solely for the purpose of implementing the Specification. No rights are granted to prepare derivative works of this Specification. Entities seeking permission to reproduce portions of this document for other uses must contact the Liberty Alliance to determine whether an appropriate license for such use is available.

Implementation of this Specification may involve the use of one or more of the following United States Patents claimed by AOL Time Warner, Inc.: No.5,774,670, No.6,134,592, No.5,826,242, No.5,825,890, and No.5,671,279. The Sponsors of the Specification take no position concerning the evidence, validity or scope of the claimed subject matter of the aforementioned patents. Implementation of certain elements of this Specification may also require licenses under third party intellectual property rights other than those identified above, including without limitation, patent rights. The Sponsors of the Specification are not and shall not be held responsible in any manner for identifying or failing to identify any or all such intellectual property rights that may be involved in the implementation of the Specification.

This Specification is provided "AS IS", and no participant in the Liberty Alliance makes any warranty of any kind, express or implied, including any implied warranties of merchantability, non-infringement or third party intellectual property rights, and fitness for a particular purpose.

Liberty Alliance Project
Liberty Protocols and Schemas Specification
Editor
John D. Beatty, Sun Microsystems, Inc.

Contributors

The following Liberty Alliance Project Sponsor companies contributed to the development of this specification:

Table of Contents

1 Introduction ................................................................................................................. 5
  1.1 Notation................................................................................................................... 5
  1.2 Overview................................................................................................................... 6
2 Schema Declarations .......................................................................................................... 6
  2.1 Schema Header and Namespace Declarations .......................................................... 6
  2.2 Type and Element Declarations .................................................................................. 6
3 Protocols .................................................................................................................... 7
  3.1 General Requirements............................................................................................... 7
    3.1.1 XML Signature........................................................................................................ 7
    3.1.2 Protocol and Assertion Versioning ......................................................................... 7
    3.1.3 Provider ID Uniqueness .......................................................................................... 7
    3.1.4 Name Identifier Construction ............................................................................... 7
    3.1.5 Security................................................................................................................. 8
  3.2 Single Sign-On and Federation Protocol ....................................................................... 8
    3.2.1 Request.................................................................................................................. 8
    3.2.2 Response................................................................................................................. 11
    3.2.3 Processing Rules ..................................................................................................... 13
    3.2.4 Request Envelope.................................................................................................... 14
    3.2.5 Response Envelope ............................................................................................... 16
  3.3 Name Registration Protocol ....................................................................................... 17
    3.3.1 Request.................................................................................................................. 17
    3.3.2 Response................................................................................................................. 18
    3.3.3 Processing Rules ..................................................................................................... 19
  3.4 Federation Termination Notification Protocol................................................................ 19
    3.4.1 Message.................................................................................................................. 19
    3.4.2 Processing Rules ..................................................................................................... 20
  3.5 Single Logout Protocol ............................................................................................... 20
    3.5.1 Message.................................................................................................................. 20
    3.5.2 Processing Rules ..................................................................................................... 21
4 Provider Metadata Schema................................................................................................ 22
  4.1 Generic Provider Descriptor ........................................................................................ 22
  4.2 Service Provider Descriptor ........................................................................................ 22
    4.2.1 Example.................................................................................................................. 23
  4.3 Identity Provider Descriptor ........................................................................................ 23
    4.3.1 Example.................................................................................................................. 24
5 Schema Definition .................................................................................................... 24
6 References .................................................................................................................. 27
1 Introduction

This specification defines the abstract Liberty protocols for identity federation, single sign-on, name registration, federation termination, and single logout. Several concrete bindings and profiles of these protocols are defined in [LibertyBindProf].

1.1 Notation

This specification uses schema documents conforming to W3C XML Schema (see [Schema1]) and normative text to describe the syntax and semantics of XML-encoded SAML assertions and protocol messages. Note: Phrases and numbers in brackets [ ] refer to other documents; details of these references can be found in Section 6 (at the end of this document).

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this specification are to be interpreted as described in [RFC2119]: “they MUST only be used where it is actually required for interoperation or to limit behavior which has potential for causing harm (e.g., limiting retransmissions).”

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

Listings of schemas appear like this.

Listings of instance fragments appear like this.

The following namespaces are referred to in this document:

- The prefix lib: stands for the Liberty namespace (http://schemas.projectliberty.org/schemas/core/2002/05/). This namespace is the default for instance fragments, type names, and element names in this document.

- The prefix saml: stands for the SAML assertion namespace (urn:oasis:names:tc:SAML:1.0:assertion).

- The prefix samlp: stands for the SAML protocol namespace (urn:oasis:names:tc:SAML:1.0:protocol).

- The prefix ds: stands for the W3C XML signature namespace (http://www.w3.org/2000/09/xmldsig#).

- The prefix xsd: stands for the W3C XML schema namespace (http://www.w3.org/2001/XMLSchema). In schema listings, this is the default namespace and no prefix is shown.

- The prefix xsi: stands for the W3C XML schema instance namespace (http://www.w3.org/2001/XMLSchema-instance).

This specification uses the following typographical conventions in text: <Element>, <ns:ForeignElement>, Attribute, Datatype, OtherCode.
For readability, when an XML Schema type is specified to be `xsd:boolean`, this document
discusses the values as “true” and “false” rather than the “1” and “0” which will exist in the
document instance.

Definitions for Liberty-specific terms can be found in [LibertyGloss].

1.2 Overview

This specification defines a set of protocols that collectively provide a solution for identity federation
management, cross-domain authentication, and session management. This specification also defines
provider metadata schemas that may be used for making a priori arrangements between providers.

The Liberty architecture contains three actors: Principal, identity provider, and service provider. A
Principal is an entity (for example, an end user) that has an identity provided by an identity provider.
A service provider provides services to the Principal.

Once the Principal is authenticated to the identity provider, the identity provider can provide an
authentication assertion to the Principal, who can present the assertion to the service provider. The
Principal is then also authenticated to the service provider if the service provider trusts the assertion.

An identity federation is said to exist between an identity provider and a service provider when the
service provider accepts authentication assertions regarding a particular Principal from the identity
provider. This specification defines a protocol where the identity of the Principal can be federated
between the identity provider and the service provider.

This specification relies on the SAML specification in [SAMLCore]. In SAML terminology, an
identity provider acts as an Asserting Party and an Authentication Authority, while a service provider
acts as a Relying Party.

2 Schema Declarations

This document specifies an XML schema for Liberty. The schema header along with namespace,
type, and element declarations are in 2.1 and 2.2.

2.1 Schema Header and Namespace Declarations

The following schema fragment defines the XML namespaces and other header information for the
Liberty schema:

```
<schema targetNamespace="http://www.projectliberty.org/schemas/core/2002/05"
xmlns:lib="http://www.projectliberty.org/schemas/core/2002/05"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:ac="http://www.projectliberty.org/schemas/authctx/2002/05"
xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <import namespace="urn:oasis:names:tc:SAML:1.0:assertion" schemaLocation="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-31.xsd"/>
  <import namespace="urn:oasis:names:tc:SAML:1.0:protocol" schemaLocation="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-protocol-31.xsd"/>
  <import namespace="http://www.projectliberty.org/schemas/authctx/2002/05" schemaLocation="draft-
liberty-architecture-authentication-context-06.xsd"/>
</schema>
```

2.2 Type and Element Declarations

Declarations for types and elements that are subsequently referred to in this document are as follows:

```
<element name="ProviderID" type="anyURI"/>
```
3 Protocols

The Liberty protocol suite consists of the following protocols:

- Single Sign-On and Federation: The protocol by which identities are federated and by which single sign-on occurs.
- Name Registration: The protocol by which a service provider can register an alternative opaque handle (or name identifier) for a Principal.
- Federation Termination Notification: The protocol by which a provider can notify another provider than a particular identity federation has been terminated (also known as defederation).
- Single Logout: The protocol by which providers notify each other of logout events for single logout functionality.

3.1 General Requirements

A set of general requirements applicable to all protocols is found in 3.1.1 through 3.1.5.

3.1.1 XML Signature

The XML signature specification calls out a general XML syntax for signing data with many flexibilities and choices. All signed XML entities MUST adhere to the “XML Signature Profile” constraints defined in [SAMLCore].

3.1.2 Protocol and Assertion Versioning

Version information appears in protocol messages and assertions defined in this specification. This specification defines version 1.0 for the protocol messages and assertions. Version numbering of assertions is independent of the version numbering of the protocol messages.

This specification follows the version numbering requirements, processing rules, and error conditions specified in “SAML Versioning” in [SAMLCore].

3.1.3 Provider ID Uniqueness

All providers have a URI-based identifier. The provider’s URI-based identifier MUST be unique within the scope of all providers with which it communicates. It is RECOMMENDED that a provider use a URL with its own domain name for this identifier. The URI-based identifier MUST NOT exceed 1024 characters.

Some profiles of the protocols contained in this specification may require a succinct 20-byte identifier. In this case, a provider’s succinct identifier also MUST be unique within the scope of all providers with which it communicates. It is RECOMMENDED that a provider derive its succinct identifier by generating the SHA-1 hash of its URI-based identifier.

3.1.4 Name Identifier Construction

Principals are assigned name identifiers by identity providers and potentially by service providers. When generated by the identity provider, a name identifier MUST be constructed using pseudo-random values that have no discernable correspondence with the Principal’s identifier (e.g., username) at the identity provider. The intent is to create a nonpublic pseudonym so as to contravene
the linkability of the Principal’s identity or activities. Service providers SHOULD follow the same
construction rules. Name identifier values MUST NOT exceed 256 characters.

3.1.5 Signature Verification

Processing rules for the protocols defined in this document commonly specify digital signature
verification. In these cases, it is not sufficient to only verify the signature of the signed object. That is,
verifying the <ds:Signature> element MUST be performed in accordance with the best practices
for the certification path technology in use. For example, when using X.509 v3 public key
certificates it is strongly RECOMMENDED that certification path validation be performed in
accordance to the PKIX Profile as specified in [RFC3280].

3.1.6 Security

Because this specification defines only abstract protocols and does not define specific protocol
profiles or the environment in which protocols will be deployed, most security requirements are
deferred to individual profiles. See [LibertyBindProf] for security considerations for the Liberty-
defined bindings and profiles. When a general security requirement can be stated for one of the
abstract protocols described in this specification, the requirement is stated in line with the specific
protocol.

3.2 Single Sign-On and Federation Protocol

The Single Sign-On and Federation Protocol defines a request and response protocol by which single
sign-on and identity federation occurs. The protocol works as follows:

1. A service provider issues an <AuthnRequest> request to an identity provider, instructing
the identity provider to provide an authentication assertion to the service provider.

   Optionally, the service provider MAY request that the identity be federated.

2. The identity provider responds with either an <AuthnResponse> containing authentication
   assertions to the service provider or an artifact that can be dereferenced into an authentication
   assertion. Additionally, the identity provider potentially federates the Principal’s identity at
   the identity provider with the Principal’s identity at the service provider.

The resulting authentication statement in the assertion by the identity provider MAY contain an
ReauthenticateOnOrAfter attribute. If this attribute is included, the service provider MUST
send a new <lib:AuthnRequest> for the Principal to the identity provider at the next point of
interaction with the Principal on or after the time specified by the ReauthenticateOnOrAfter
attribute. It is then up to the identity provider to authenticate the user. Note: The Principal may
already have an authenticated session with the identity provider, in which case a new authentication
assertion would be generated by the identity provider without any intervention by the Principal.

3.2.1 Request

The service provider issues an <AuthnRequest> request to the identity provider. A set of
parameters is included in the request that allows the service provider to specify desired behavior at
the identity provider in processing the request. The service provider can control the following
identity provider behaviors:

- Prompt the Principal for credentials if the Principal is not presently authenticated.
- Prompt the Principal for credentials, even if the Principal is presently authenticated.
Federate the Principal’s identity at the identity provider with the Principal’s identity at the service provider.

Use a specific protocol profile in responding to the request.

Use a specific authentication context (for example, smartcard-based authentication vs. username/password-based authentication).

Additionally, the service provider MAY include any desired state information in the request that the identity provider should relay back to the service provider in the response.

### 3.2.1.1 Element <AuthnRequest>

The `<AuthnRequest>` request is defined as an extension of `samlp:RequestAbstractType`. The `RequestID` attribute in `samlp:RequestAbstractType` has uniqueness requirements placed on it by [SAMLCore], which require it to have the properties of a nonce.

The elements of the request are as follows:

- **ProviderID [Required]**
  
  The service provider’s URI-based identifier.

- **IsPassive [Optional]**
  
  If “true,” specifies that the identity provider MUST NOT interact with the Principal and MUST NOT take control of the user interface from the service provider. If “false,” the identity provider MAY interact with the user and MAY temporarily take control of the user interface for that purpose. If not specified, “true” is presumed.

- **ForceAuthn [Optional]**
  
  Controls whether the identity provider authenticates the Principal regardless of whether the Principal is already authenticated. This element is specified only when `<IsPassive>` is “false.” If `<ForceAuthn>` is “true,” specifies that the identity provider MUST always authenticate the Principal, regardless of whether the Principal is presently authenticated. If “false,” specifies that the identity provider MUST reauthenticate the user only if the Principal is not presently authenticated. If not specified, “false” is presumed.

- **Federate [Optional]**
  
  Specifies that the service provider wishes to federate the Principal’s identity at the service provider with the Principal’s identity at the identity provider. If the element is not specified, it is presumed that the service provider does not wish to federate the identity.

- **ProtocolProfile [Optional]**
  
  The protocol profile that the service provider wishes to use for the response. If the element is not specified, the default protocol profile is `http://projectliberty.org/profiles/brws-art` defined in [LibertyBindProf].

- **AuthnContext [Optional]**
  
  Information regarding which authentication context the service provider desires the identity provider to use in authenticating the Principal.

- **RelayState [Optional]**
State information that will be relayed back in the response. This data SHOULD be integrity-protected by the request author and MAY have other protections place on it by the request author. An example of such protection is confidentiality.

The `<AuthnContext>` element has the following mutually exclusive elements:

- **AuthnContextClassRef** [Optional]

  The ordered set of authentication context class references the service provider desires the identity provider to use in authenticating the Principal.

- **AuthnContextMinimumClassRef** [Optional]

  The minimum authentication context class the service provider will accept.

- **AuthnContextStatementRef** [Optional]

  The ordered set of exact authentication statements the service provider desires the identity provider to use in authenticating the Principal.

The schema fragment defining the element and its type is as follows:

```xml
<element name="AuthnRequest" type="lib:AuthnRequestType"/>
<complexType name="AuthnRequestType">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <element ref="lib:ProviderID"/>
        <element name="ForceAuthn" type="boolean" minOccurs="0"/>
        <element name="IsPassive" type="boolean" minOccurs="0"/>
        <element ref="lib:ProtocolProfile" minOccurs="0"/>
        <element ref="lib:AuthnContext" minOccurs="0"/>
        <element ref="lib:RelayState" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

### 3.2.1.2 Example

```xml
<AuthnRequest RequestID="4e7c3772-4fa4-4a0f-99e8-7d719ff6067c" MajorVersion="1" MinorVersion="0"
IssueInstant="2001-12-17T09:30:47-05:00">
  <ds:Signature> ... </ds:Signature>
  <ProviderID>http://ServiceProvider.com</ProviderID>
  <ForceAuthn>1</ForceAuthn>
  <IsPassive>0</IsPassive>
  <Federate>1</Federate>
  <ProtocolProfile>http://projectliberty.org/profiles/brws-post</ProtocolProfile>
  <AuthnContext>
    <AuthnContextClassRef>http://projectliberty.org/schemas/authctx/2002/05/Password</AuthnContextClassRef>
    <AuthnContext>http://projectliberty.org/schemas/authctx/2002/05/Password</AuthnContext>
    <RelayState>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</RelayState>
  </AuthnContext>
</AuthnRequest>
```
3.2.2 Response

The response is a <AuthnResponse> element containing either a set of authentication assertions or a set of artifacts the service provider can dereference into a set of authentication assertions.

All authentication assertions generated by an identity provider for a service provider MUST be of type AssertionType. The <saml:Subject> element in any subject statement MUST be of type SubjectType. If the service provider registered a name identifier for the Principal (see 3.3), the <saml:NameIdentifier> element in the <saml:Subject> element MUST be the service provider-provided name identifier for the Principal. Otherwise, <saml:NameIdentifier> MUST be the name identifier provided by the identity provider at the time of federation. The <IDPProvidedNameIdentifier> MUST contain the name identifier provided by the identity provider at the time of federation.

All authentication statements MUST be of type AuthenticationStatementType.

Identity providers MUST include a <saml:AudienceRestrictionCondition> element that specifies the intended consumers of the assertion.

Identity providers MAY include a SessionIndex attribute in resulting authentication statements, which is used to aid the identity provider in managing multiple sessions with the Principal. If the identity provider includes this SessionIndex attribute, subsequent messages from the service provider to the identity provider that are session-dependent MUST include this SessionIndex attribute.

All assertions in the <AuthnResponse> MUST be signed by the identity provider.

3.2.2.1 Element <AuthnResponse>

The type AuthnResponseType, extended from samlp:ResponseType, provides for the <RelayState> element. The schema fragment is as follows:

```
<element name="AuthnResponse" type="lib:AuthnResponseType"/>
<complexType name="AuthnResponseType">
  <complexContent>
    <extension base="samlp:ResponseType">
      <sequence>
        <element ref="lib:RelayState" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

3.2.2.2 Element <Assertion>

Assertions provided in an <AuthnResponse> element MUST be of type AssertionType, which is an extension of saml:AssertionType, so that the RequestID attribute from the original <AuthnRequest> is included in the InResponseTo attribute in the <Assertion> element. This step is done because the <AuthnResponse> element is not required to be signed, whereas the contained <Assertion> elements are required to be signed. The schema fragment is as follows:

```
<element name="Assertion" type="lib:AssertionType"/>
<complexType name="AssertionType">
  <complexContent>
    <extension base="saml:AssertionType">
      <attribute name="InResponseTo" type="saml:IDReferenceType"/>
    </extension>
  </complexContent>
</complexType>
```
3.2.2.3 Type SubjectType

The type SubjectType, extended from saml:SubjectType, is used to include the <IDPProvidedNameIdentifier> element in subject statements. The schema fragment is as follows:

```xml
<complexType name="SubjectType">
  <complexContent>
    <extension base="saml:SubjectType">
      <sequence>
        <element ref="lib:IDPProvidedNameIdentifier"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

3.2.2.4 Type AuthenticationStatementType

The type AuthenticationStatementType is an extension of saml:AuthenticationStatementType, which allows for the following elements and attributes:

- **AuthnContext [Optional]**
  - The authentication context that the identity provider used in the authentication event that yielded this authentication statement. Contains either an authentication context statement or a reference to an authentication context statement. Optionally contains a reference to an authentication context class.

- **ReauthenticateOnOrAfter [Optional]**
  - The time at which the service provider reauthenticates the Principal with the identity provider.

- **SessionIndex [Optional]**
  - The index into the particular session between the Principal and the identity provider under which this authentication statement is being issued.

The schema fragment is as follows:

```xml
<complexType name="AuthenticationStatementType">
  <complexContent>
    <extension base="saml:AuthenticationStatementType">
      <sequence>
        <element name="AuthnContext" minOccurs="0">
          <complexType>
            <sequence>
              <element name="AuthnContextClassRef" type="anyURI" minOccurs="0"/>
            </sequence>
          </complexType>
        </element>
      </sequence>
      <attribute name="ReauthenticateOnOrAfter" type="dateTime" use="optional"/>
      <attribute name="SessionIndex" type="string" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

3.2.2.5 Example

```xml
<AuthnResponse ResponseID="a10072f3-4dae-4757-9357-03b95327fd77" InResponseTo="4e7c3772-4fa4-48099e8-7d719ff6067c" MajorVersion="1" MinorVersion="0" IssueInstant="2001-12-17T09:30:47-05:00">
  <samlp:Status>
    <samlp:StatusCode Value="saml:Success"/>
  </samlp:Status>
</AuthnResponse>
```
3.2.3 Processing Rules

When an identity provider receives an authentication request, it MUST process the request according to the following rules:

- The <ProviderID> MUST be the Provider ID of a known service provider with which the identity provider has established a relationship. The <ProviderID> MUST be resolvable to an assertion consumer service URL at the service provider that the identity provider may use when returning the corresponding assertion reference.

- If <IsPassive> is “false,” the identity provider MUST NOT interact with the Principal and MUST NOT take control of the user interface (if applicable).

- The identity provider MUST attempt to authenticate the Principal if <ForceAuthn> is “true,” regardless of whether the Principal is presently authenticated, unless <IsPassive> is “true.”

- The identity provider MUST attempt to authenticate the Principal if <ForceAuthn> is “false” and the Principal is not presently authenticated, unless <IsPassive> is “true.”

- The identity provider MAY federate the Principal’s identity at the service provider with the user’s identity at the identity provider if <Federate> is “true” and the Principal has consented for such an action to occur. The identity provider MUST NOT federate if <Federate> is “false.”

- The identity provider MUST respond using the specified <ProtocolProfile>.

- If <RelayState> contains a value, the identity provider MUST include this value in unmodified form in the <RelayState> element of the returned authentication assertion.

- The InResponseTo attribute in all generated <Assertion> elements in the <AuthnResponse> element MUST be set to the value of the RequestID attribute in the corresponding <AuthnRequest> element.

Additionally, if the <AuthnContext> element is specified, the identity provider MUST authenticate the Principal according to the following rules:
• If one or more \(<\text{AuthnContextClassRef}>\) elements are included, then the resulting authentication statement in the assertion (if any) MUST contain an authentication statement that conforms to one of the specified classes. Additionally, the set of \(<\text{AuthnContextClassRef}>\) elements MUST be evaluated as an ordered set, where the first element is the most preferred authentication context class. If none of the specified authentication context classes can be satisfied, the identity provider MUST not include an authentication statement in the resulting assertion.

• If \(<\text{AuthnContextMinimumClassRef}>\) is specified, then the resulting authentication statement in the assertion (if any) MUST have an authentication statement that is deemed to be at least as strong as the stated minimum class, as deemed by the identity provider. If the minimum class cannot be satisfied, the identity provider MUST not include an authentication statement in the resulting assertion.

• If one or more \(<\text{AuthnContextStatementRef}>\) elements are included, then the resulting authentication statement in the assertion (if any) MUST have an authentication statement that exactly matches one of the specified authentication context statement(s). If this requirement cannot be satisfied, the identity provider MUST not include an authentication statement in the resulting assertion.

If an identity is being federated, the identity provider MUST adhere to the following rules in generating the name identifier:

• The name identifier MUST be unique across all Principals in the scope of the service provider-identity provider pairwise relationship.

• The name identifier for the specific Principal MUST be unique across all service providers with which an identity federation exists with the identity provider.

Failure to either authenticate the Principal and/or federate the identity is indicated by a status code other than “Success.” For failures, assertions MUST NOT appear in the \(<\text{AuthnResponse}>\).

In some profiles, an intermediary is active between the service provider’s authentication request and the identity provider’s authentication response. If an error occurs in the processing at the intermediary, the following status code values are defined for the \(<\text{saml:Status}>\) element:

• \text{lib:NoAvailableIDP}: Used to indicate that none of the identity provider URLs from the \(<\text{IDPList}>\) can be resolved or that none of the identity providers is available.

• \text{lib:NoSupportedIDP}: Used to indicate that none of the identity providers is supported by the intermediary.

### 3.2.4 Request Envelope

Some profiles MAY wrap the \(<\text{AuthnRequest}>\) element in an envelope. This envelope allows for extra processing by an intermediary between the service provider and the identity provider. An example of an intermediary is a user agent or proxy.

#### 3.2.4.1 Element \(<\text{AuthnRequestEnvelope}>\)

The authentication request envelope contains the following elements:

\begin{itemize}
  \item \text{AuthnRequest} [Required]
\end{itemize}

The enveloped authentication request.
ProviderID [Required]  
The ProviderID of the requestor.

ProviderName [Optional]  
The human-readable name of the requestor.

AssertionConsumerServiceURL [Required]  
The service provider’s URL where the authentication response should be sent.

IDPList [Optional]  
A list of identity providers from which a provider may be chosen to service the authentication request.

IsPassive [Optional]  
If “true,” specifies that any intermediary between the service provider and identity provider MUST NOT interact with the Principal. If not specified, “true” is presumed.

The schema fragment is as follows:

```xml
<element name="AuthnRequestEnvelope" type="lib:AuthnRequestEnvelopeType"/>
<complexType name="AuthnRequestEnvelopeType">
    <complexContent>
        <extension base="lib:RequestEnvelopeType">
            <sequence>
                <element ref="lib:AuthnRequest"/>
                <element ref="lib:ProviderID"/>
                <element name="ProviderName" type="string" minOccurs="0"/>
                <element name="AssertionConsumerServiceURL" type="anyURI"/>
                <element ref="lib:IDPList" minOccurs="0"/>
                <element name="IsPassive" type="boolean" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

3.2.4.2 Element <IDPList>

In the request envelope, some profiles may wish to allow the service provider to transport a list of identity providers to the user agent. This specification provides a schema that profiles SHOULD use for this purpose. The elements are as follows:

IDPList  
The container element for an IDP List.

IDPEntries  
Contains a list of identity provider entries.

IDPEntry  
Describes an identity provider that the service provider supports.

ProviderID  
The identity provider’s ProviderID.

ProviderName  
The identity provider’s human-readable name.
Loc

The URI at which the identity provider can be contacted for sending the authentication request.

GetComplete

If the identity provider list is not complete, this element is included with a URI that points to where the complete list can be retrieved.

The schema fragment is as follows:

```
<element name="IDPList" type="lib:IDPListType"/>
<complexType name="IDPListType">
  <sequence>
    <element ref="lib:IDPEntries"/>  
    <element ref="lib:GetComplete" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
<element name="IDPEntry">
  <complexType>
    <sequence>
      <element ref="lib:ProviderID"/>
      <element name="ProviderName" type="string" minOccurs="0"/>
      <element name="Loc" type="anyURI"/>
    </sequence>
  </complexType>
</element>
<element name="IDPEntries">
  <complexType>
    <sequence>
      <element ref="lib:IDPEntry" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
<element name="GetComplete" type="anyURI"/>
```

### 3.2.4.3 Example

```
<AuthnRequestEnvelope>
  <AuthnRequest> ... </AuthnRequest>
  <ProviderID>http://ServiceProvider.com</ProviderID>
  <ProviderName>Service Provider X</ProviderName>
  <AssertionConsumerServiceURL>http://ServiceProvider.com/lecp_assertion_consumer</AssertionConsumerServiceURL>
  <IDPList>
    <IDPEntries>
      <IDPEntry>
        <ProviderID>http://IdentityProvider.com</ProviderID>
        <ProviderName>Identity Provider X</ProviderName>
        <Loc>http://www.IdentityProvider.com/liberty/sso</Loc>
      </IDPEntry>
    </IDPEntries>
  </IDPList>
  <GetComplete>https://ServiceProvider.com/idplist?id=604be136-fe91-441e-afb8-f88748ae3b8b</GetComplete>
</AuthnRequestEnvelope>
```

### 3.2.5 Response Envelope

As with the `<AuthnRequest>` element, some profiles MAY wrap the `<AuthnResponse>` element in an envelope. This envelope allows for extra processing by an intermediary between the identity provider and the service provider. An example of an intermediary is a user agent or proxy.

#### 3.2.5.1 Element `<AuthnResponseEnvelope>`

The authentication response envelope contains a single `<AuthnResponse>` element, along with any number of arbitrary elements. The schema fragment is as follows:
3.2.5.2 Example

```xml
<AuthnResponseEnvelope>
  <AuthnResponse> ... </AuthnResponse>
  <idp:Foo></idp:Foo>
</AuthnResponseEnvelope>
```

### 3.3 Name Registration Protocol

At the time of federation, the identity provider generates an opaque handle that serves as the name identifier the service provider and the identity provider use in referring to the Principal when communicating with each other. This name identifier is termed the IDPProvidedNameIdentifier. Subsequent to federation, however, the service provider MAY register a different opaque handle with the identity provider. This opaque handle is termed the SPProvidedNameIdentifier. Subsequent to name registration, the identity provider MUST use the SPProvidedNameIdentifier for `<saml:NameIdentifier>` elements when communicating to the service provider about the Principal. The service provider MUST continue using the IDPProvidedNameIdentifier for `<saml:NameIdentifier>` elements when communicating to the identity provider about the Principal.

The service provider MAY register a new name identifier for a Principal with the identity provider at any time subsequent to federation.

The name identifier specified by the service provider SHOULD adhere to the following guidelines:

- The name identifier SHOULD be unique across the identity providers with which the Principal’s identity is federated.
- The name identifier SHOULD be unique across all name identifiers that have been registered with the identity provider by this service provider.

#### 3.3.1 Request

To register a SPProvidedNameIdentifier with an identity provider, the service provider sends a `<RegisterNameIdentifierRequest>` message.

#### 3.3.1.1 Element `<RegisterNameIdentifierRequest>`

The elements of the message are as follows:

- **ProviderID** [Required]
  
  The service provider’s identifier.
IDPProvidedNameIdentifier [Required]

The name identifier the identity provider provided at the time of federation.

SPProvidedNameIdentifier [Required]

The name identifier the identity provider will use when communicating to the service provider.

The schema fragment is as follows:

```xml
<element name="RegisterNameIdentifierRequest" type="lib:RegisterNameIdentifierRequestType"/>
<complexType name="RegisterNameIdentifierRequestType">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <element ref="lib:ProviderID"/>
        <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
        <element name="SPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

### 3.3.1.2 Example

```xml
<RegisterNameIdentifierRequest RequestID="eb20e77f-d982-44f9-936e-dd135bf437d4" MajorVersion="1" MinorVersion="0" IssueInstant="2001-12-17T09:30:47-05:00">
  <ProviderID>http://ServiceProvider.com</ProviderID>
  <IDPProvidedNameIdentifier>342ad3d8-93ee-4c68-be35-cc9e7db39e2b</IDPProvidedNameIdentifier>
  <SPProvidedNameIdentifier>e958019a</SPProvidedNameIdentifier>
</RegisterNameIdentifierRequest>
```

### 3.3.2 Response

The identity provider MUST respond with `<RegisterNameIdentifierResponse>`, which is of type `StatusResponseType`. `StatusResponseType` is an extension of `samlp:ResponseAbstractType` so that only a `<samlp:Status>` element exists in the body.

#### 3.3.2.1 Element `<RegisterNameIdentifierResponse>`

The elements of the message are as follows:

**Status** [Required]

The status of the request processing.

The schema fragment is as follows:

```xml
<element name="RegisterNameIdentifierResponse" type="lib:StatusResponseType"/>
<complexType name="StatusResponseType">
  <complexContent>
    <extension base="samlp:ResponseAbstractType">
      <sequence>
        <element ref="samlp:Status"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

#### 3.3.2.2 Example

```xml
<RegisterNameIdentifierResponse ResponseID="7ffec0e0f-1165-4f4a-3b088-3dd2c2388b91" InResponseTo="eb20e77f-d982-44f9-936e-dd135bf437d4" MajorVersion="1" MinorVersion="0" IssueInstant="2001-12-17T09:30:47-05:00" Recipient="http://ServiceProvider.com">
  <ds:Signature>...</ds:Signature>
  <samlp:Status/>
</RegisterNameIdentifierResponse>
```
3.3.3 Processing Rules

The identity provider MUST validate the signature on the message. To be considered valid, the signature on the message MUST be the signature of the <ProviderID> contained in the message.

If the request includes an <IDPProvidedNameIdentifier> for which no federation exists between the service provider and the identity provider, the Provider MUST respond with a <saml:Status> with a <saml:StatusCode> of lib:FederationDoesNotExist.

Otherwise, the identity provider MUST use <SPProvidedNameIdentifier> when subsequently communicating to the service provider regarding this Principal.

3.4 Federation Termination Notification Protocol

When the Principal terminates an identity federation between a service provider and an identity provider from the service provider, the service provider MUST send a <FederationTerminationNotification> message to the identity provider. Semantically, the service provider is stating that it will no longer accept authentication assertions from the identity provider for the specified Principal.

Likewise, when the Principal terminates an identity federation from the identity provider, the identity provider MUST send a <FederationTerminationNotification> message to the service provider. Semantically, the identity provider is stating that it will no longer provide authentication assertions to the service provider for the specified Principal.

This notification message is a one-way asynchronous message, and reasonable best-effort delivery MUST be employed by both service providers and identity providers sending the message. The message MUST be signed.

3.4.1 Message

The provider sends a <FederationTerminationNotification> to the provider with which it is terminating a federation.

3.4.1.1 Element <FederationTerminationNotification>

The elements are as follows:

ProviderID [Required]

The identifier of the provider that is sending the notification of federation termination.

NameIdentifier [Required]

The name identifier of the Principal to which the notification pertains.

The schema fragment is as follows:
822</complexContent>
823</complexType>

3.4.1.2 Example

824<FederationTerminationNotification RequestID="9ec2-eb65-4bce-ab8f-4becdf229815" MajorVersion="1"
825MinorVersion="0" IssueInstant="2001-12-17T09:30:47-05:00">
826<ds:Signature>...</ds:Signature>
827<ProviderID>http://IdentityProvider.com</ProviderID>
828<saml:NameIdentifier>e958019a</saml:NameIdentifier>
829</FederationTerminationNotification>

3.4.2 Processing Rules

830The receiving provider MUST validate the signature on the message. The signature on the message
831MUST be the signature of the <ProviderID> contained in the message. If the signature is not valid,
832the provider MUST ignore the message.
833If a provider receives a federation termination notification message that refers to a federation that
834does not exist from the perspective of the provider, the provider MUST ignore the message.
835Otherwise, the provider MAY perform any maintenance with the knowledge that the federation has
836been terminated.

3.5 Single Logout Protocol

840The Single Logout Protocol provides a one-way asynchronous message exchange protocol by which
841all sessions authenticated by a particular identity provider are near-simultaneously terminated. The
842Single Logout Protocol is used either when a Principal logs out at a service provider or when the
843Principal logs out at an identity provider.
844When the Principal invokes the single logout process at a service provider, the service provider
845MUST send a <LogoutNotification> message to the identity provider that provided the
846authentication service for the session.
847When either the Principal invokes a logout at the identity provider or a service provider sends a
848logout notification to the identity provider specifying that Principal, the identity provider MUST
849send a <LogoutNotification> message to each service provider to which it provided
850authentication assertions in the current session with the Principal, with the exception of the service
851provider that sent the <LogoutNotification> message to the Identity Provider.
852This notification is a one-way asynchronous message, and reasonable best-effort delivery SHOULD
853be employed by both service providers and identity providers.
854This message SHOULD be signed.

3.5.1 Message

856The <LogoutNotification> message indicates to the message receiver that a Principal’s session
858was terminated. The message includes an optional <SessionIndex> element that MUST be
859specified if and only if the authentication statement in the assertion that the service provider used in
860establishing the session with the Principal contained a SessionIndex attribute.

3.5.1.1 Element <LogoutNotification>

861NameIdentifier [Required]
862The name identifier of the Principal that logged out.
ProviderID [Required]
The identifier of the provider that is making the request.

SessionIndex [Optional]
The session index specified in the authentication statement in the assertion that was used in establishing the session being terminated.

The schema fragment is as follows:

```xml
<element name="LogoutNotification" type="lib:LogoutNotificationType"/>
<complexType name="LogoutNotificationType">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <element ref="lib:ProviderID"/>
        <element ref="saml:NameIdentifier"/>
        <element name="SessionIndex" type="string" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

### 3.5.1.2 Example

```xml
<LogoutNotification RequestID="47693d03-7c13-4d65-931f-ddeb19fa6a73" MajorVersion="1"
MinorVersion="0" IssueInstant="2001-12-17T09:30:47-05:00" ProviderID=http://ServiceProvider.com/ProviderID>
<saml:NameIdentifier>342ad3d8-93ee-4c68-be35-cc9e7db39e2b</saml:NameIdentifier>
<SessionIndex>3</SessionIndex>
</LogoutNotification>
```

### 3.5.2 Processing Rules

Unique processing rules apply based on whether the message receiver is an identity provider or a service provider.

#### 3.5.2.1 Identity Provider Processing Rules

When an identity provider receives the `<LogoutNotification>` message, the identity provider MUST validate that the signature on the message is the signature of a service provider to which the identity provider provided an authentication assertion for the current session. If that holds, the identity provider SHOULD do the following:

- Send a `<LogoutNotification>` message to each service provider for which the identity provider provided authentication assertions in the current session.
- Terminate the Principal’s current session as specified by the `<saml:NameIdentifier>` element.

#### 3.5.2.2 Service Provider Processing Rules

When the service provider receives the `<LogoutNotification>` message, the service provider MUST validate the identity provider’s signature contained in the `<ds:Signature>` element. If the signature is that of the identity provider that provided the authentication for the Principal’s current session, the service provider MUST invalidate the Principal’s session referred to in the `<saml:NameIdentifier>` element.
4 Provider Metadata Schema

For providers to communicate with each other, they must a priori have obtained metadata regarding each other. These provider metadata include items such as X.509 certificates and service endpoints. This specification defines metadata schemas for identity providers and service providers that may be used for provider metadata exchange. However, provider metadata exchange protocols are outside the scope of this specification.

4.1 Generic Provider Descriptor

Certain provider metadata are generic to both service providers and identity providers. The complex type ProviderDescriptorType contains the following elements:

- ProviderID [Required]
  The provider’s URI-based identifier.
- ProviderSuccinctID [Required]
  The provider’s succinct identifier.
- KeyInfo [Optional]
  The provider’s public key.
- SoapEndpoint [Optional]
  The provider’s SOAP endpoint URI.
- SingleLogoutServiceURL [Optional]
  The URL used for user-agent-based Single Logout Protocol profiles.
- SingleLogoutServiceReturnURL [Optional]
  The URL to which the provider redirects at the end of user-agent-based Single Logout Protocol profiles.
- FederationTerminationServiceURL [Optional]
  The URL used for user-agent-based Federation Termination Notification Protocol profiles.
- FederationTerminationServiceReturnURL [Optional]
  The URL to which the provider redirects at the end of user-agent-based Federation Termination Notification Protocol profiles.

The schema fragment is as follows:

```xml
<complexType name="ProviderDescriptorType">
  <sequence>
    <element name="ProviderID" type="anyURI"/>
    <element name="ProviderSuccinctID" type="hexBinary"/>
    <element ref="ds:KeyInfo" minOccurs="0"/>
    <element name="SoapEndpoint" type="anyURI" minOccurs="0"/>
    <element name="SingleLogoutServiceURL" type="anyURI" minOccurs="0"/>
    <element name="SingleLogoutServiceReturnURL" type="anyURI" minOccurs="0"/>
    <element name="FederationTerminationServiceURL" type="anyURI" minOccurs="0"/>
    <element name="FederationTerminationServiceReturnURL" type="anyURI" minOccurs="0"/>
  </sequence>
</complexType>
```

4.2 Service Provider Descriptor

The additional service provider-specific metadata are as follows:
AssertionConsumerServiceURL [Required]
The service provider’s URL for consuming assertions from identity providers.

FederationTerminationNotificationProtocolProfile [Required]
The Federation Termination Notification Protocol profile that the identity provider should use when communicating with the service provider.

SingleLogoutProtocolProfile [Required]
The Single Logout Protocol profile that the identity provider should use when communicating with the service provider.

AuthnRequestsSigned [Required]
Specifies whether the service provider will always sign authentication requests it sends to the identity provider.

The schema fragment is as follows:

```xml
<element name="SPDescriptor" type="lib:SPDescriptorType"/>
<complexType name="SPDescriptorType">
  <complexContent>
    <extension base="lib:ProviderDescriptorType">
      <sequence>
        <element name="AssertionConsumerServiceURL" type="anyURI"/>
        <element name="FederationTerminationNotificationProtocolProfile" type="anyURI"/>
        <element name="SingleLogoutProtocolProfile" type="anyURI"/>
        <element name="AuthnRequestsSigned" type="boolean"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

4.2.1 Example

```xml
<SPDescriptor>
  <ProviderID>http://ServiceProvider.com/</ProviderID>
  <ProviderSuccinctID>A9FD64E12C</ProviderSuccinctID>
  <ds:KeyInfo>...</ds:KeyInfo>
  <SoapEndpoint>http://ServiceProvider.com/soap</SoapEndpoint>
  <SingleLogoutServiceURL>http://ServiceProvider.com/liberty/slo</SingleLogoutServiceURL>
  <FederationTerminationServiceURL>http://ServiceProvider.com/liberty/term</FederationTerminationServiceURL>
  <FederationTerminationServiceReturnURL>http://ServiceProvider.com/liberty/term_return</FederationTerminationServiceReturnURL>
  <AssertionConsumerServiceURL>http://ServiceProvider.com/liberty/assertion_consumer</AssertionConsumerServiceURL>
  <FederationTerminationNotificationProtocolProfile>http://projectliberty.org/profiles/fedterm_soap</FederationTerminationNotificationProtocolProfile>
  <AuthnRequestsSigned>1</AuthnRequestsSigned>
</SPDescriptor>
```

4.3 Identity Provider Descriptor

The additional identity provider-specific metadata are as follows:

SingleSignOnServiceURL [Required]

The schema fragment is as follows:

```xml
<element name="IDPDescriptor" type="lib:IDPDescriptorType"/>
```
<complexType name="IDPDescriptorType">
  <complexContent>
    <extension base="lib:ProviderDescriptorType">
      <sequence>
        <element name="SingleSignOnServiceURL" type="anyURI"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

4.3.1 Example

<IDPDescriptor>
  <ProviderID>http://IdentityProvider.com</ProviderID>
  <ProviderSuccinctID>A9FD64E12C</ProviderSuccinctID>
  <ds:KeyInfo>...</ds:KeyInfo>
  <SoapEndpoint>http://IdentityProvider.com/soap</SoapEndpoint>
  <FederationTerminationServiceURL>http://IdentityProvider.com/liberty/term</FederationTerminationServiceURL>
  <FederationTerminationServiceReturnURL>http://IdentityProvider.com/liberty/term_return</FederationTerminationServiceReturnURL>
</IDPDescriptor>

5 Schema Definition

<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.projectliberty.org/schemas/core/2002/05"
  xmlns:lib="http://www.projectliberty.org/schemas/core/2002/05"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns:ac="http://www.projectliberty.org/schemas/authctx/2002/05"
  xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <import namespace="http://www.projectliberty.org/schemas/authctx/2002/05" schemaLocation="draft-liberty-architecture-authentication-context-06.xsd"/>
  <!-- Begin protocols schema -->
  <element name="ProviderID" type="anyURI"/>
  <element name="AuthnRequest" type="lib:AuthnRequestType"/>
  <complexType name="AuthnRequestType">
    <complexContent>
      <extension base="samlp:RequestAbstractType">
        <sequence>
          <element ref="lib:ProviderID"/>
          <element name="ForceAuthn" type="boolean" minOccurs="0"/>
          <element name="IsPassive" type="boolean" minOccurs="0"/>
          <element name="Federate" type="boolean" minOccurs="0"/>
          <element ref="lib:ProtocolProfile" minOccurs="0"/>
          <element ref="lib:AuthnContext" minOccurs="0"/>
          <element ref="lib:RelayState" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <element name="RelayState"/>
  <element name="ProtocolProfile" type="anyURI"/>
  <element name="AuthnContext">
    <complexType>
      <choice>
        <element name="AuthnContextClassRef" type="anyURI" maxOccurs="unbounded"/>
        <element name="AuthnContextStatementRef" type="anyURI" maxOccurs="unbounded"/>
        <element name="AuthnContextMinimumClassRef" type="anyURI"/>
      </choice>
    </complexType>
  </element>
  <element name="AuthnRequestEnvelope" type="lib:AuthnRequestEnvelopeType"/>
<complexType name="AuthnRequestEnvelopeType">
  <extension base="lib:RequestEnvelopeType">
    <sequence>
      <element ref="lib:AuthnRequest"/>
      <element ref="lib:ProviderID"/>
      <element name="ProviderName" type="string" minOccurs="0"/>
      <element name="AssertionConsumerServiceURL" type="anyURI"/>
      <element ref="lib:IDPList" minOccurs="0"/>
      <element name="IsPassive" type="boolean" minOccurs="0"/>
    </sequence>
  </extension>
</complexContent>
</complexType>

<complexType name="RequestEnvelopeType">
  <sequence>
    <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<complexType name="AuthnResponseEnvelope" type="lib:AuthnResponseEnvelopeType"/>
<complexType name="AuthnResponseEnvelopeType">
  <extension base="lib:ResponseEnvelopeType">
    <sequence>
      <element ref="lib:AuthnResponse"/>
    </sequence>
  </extension>
</complexContent>
</complexType>

<complexType name="ResponseEnvelopeType">
  <sequence>
    <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<complexType name="SubjectType">
  <complexContent>
    <extension base="saml:SubjectType">
      <sequence>
        <element ref="lib:IDPProvidedNameIdentifier"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="AuthnResponse" type="lib:AuthnResponseType"/>
<complexType name="AuthnResponseType">
  <complexContent>
    <extension base="samlp:ResponseType">
      <sequence>
        <element ref="lib:RelayState" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name="AuthenticationStatementType">
  <complexContent>
    <extension base="saml:AuthenticationStatementType">
      <sequence>
        <element name="AuthnContext" minOccurs="0">
          <complexType>
            <sequence>
              <element name="AuthnContextClassRef" type="anyURI" minOccurs="0"/>
              <choice>
                <element ref="ac:AuthenticationContextStatement"/>
                <element name="AuthnContextStatementRef" type="anyURI"/>
              </choice>
            </sequence>
            <attribute name="ReauthenticateOnOrAfter" type="dateTime" use="optional"/>
            <attribute name="SessionIndex" type="string" use="optional"/>
          </complexType>
        </element>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="RegisterNameIdentifierRequest" type="lib:RegisterNameIdentifierRequestType"/>
<complexType name="RegisterNameIdentifierRequestType"/>
<complexContent>
  <extension base="samlp:RequestAbstractType">
    <sequence>
      <element ref="lib:ProviderID"/>
      <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
      <element name="SPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
    </sequence>
  </extension>
</complexContent>

<complexType>
  <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
  <element name="SPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
  <element name="RegisterNameIdentifierResponse" type="lib:StatusResponseType"/>
  <complexType name="StatusResponseType">
    <complexContent>
      <extension base="samlp:ResponseAbstractType">
        <sequence>
          <element ref="samlp:Status"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <element name="FederationTerminationNotification" type="lib:FederationTerminationNotificationType"/>
  <complexType name="FederationTerminationNotificationType">
    <complexContent>
      <extension base="samlp:RequestAbstractType">
        <sequence>
          <element ref="lib:ProviderID"/>
          <element ref="saml:NameIdentifier"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <element name="LogoutNotification" type="lib:LogoutNotificationType"/>
  <complexType name="LogoutNotificationType">
    <complexContent>
      <extension base="samlp:RequestAbstractType">
        <sequence>
          <element ref="lib:ProviderID"/>
          <element ref="saml:NameIdentifier"/>
          <element name="SessionIndex" type="string" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</complexType>

<!-- End protocols schema -->

<!-- Begin assertion schema -->

<element name="Assertion" type="lib:AssertionType"/>
<complexType name="AssertionType">
  <complexContent>
    <extension base="saml:AssertionType">
      <attribute name="InResponseTo" type="saml:IDReferenceType"/>
    </extension>
  </complexContent>
</complexType>

<!-- End assertion schema -->

<!-- Begin IDP list schema -->

<element name="IDPList" type="lib:IDPListType"/>
<complexType name="IDPListType">
  <sequence>
    <element ref="lib:IDPEntries"/>
    <element ref="lib:GetComplete" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
<element name="IDPEntry">
  <complexType>
    <sequence>
      <element ref="lib:ProviderID"/>
      <element name="ProviderName" type="string" minOccurs="0"/>
      <element name="Loc" type="anyURI"/>
    </sequence>
  </complexType>
</element>
<element name="IDPEntries">
  <complexType>
    <sequence>
      <element ref="lib:IDPEntries"/>
      <element ref="lib:GetComplete" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
</complexType>
<sequence>
  <element ref="lib:IDPEntry" maxOccurs="unbounded"/>
</sequence>
</complexType>

<element name="GetComplete" type="anyURI"/>

<!-- End IDP list schema -->

<!-- Begin provider metadata schema -->
<complexType name="ProviderDescriptorType">
  <sequence>
    <element name="ProviderID" type="anyURI"/>
    <element name="ProviderSuccinctID" type="hexBinary"/>
    <element ref="ds:KeyInfo" minOccurs="0"/>
    <element name="SoapEndpoint" type="anyURI" minOccurs="0"/>
    <element name="SingleLogoutServiceURL" type="anyURI" minOccurs="0"/>
    <element name="SingleLogoutServiceReturnURL" type="anyURI" minOccurs="0"/>
    <element name="FederationTerminationServiceURL" type="anyURI" minOccurs="0"/>
    <element name="FederationTerminationServiceReturnURL" type="anyURI" minOccurs="0"/>
  </sequence>
</complexType>

<element name="SPDescriptor" type="lib:SPDescriptorType"/>
<complexType name="SPDescriptorType">
  <complexContent>
    <extension base="lib:ProviderDescriptorType">
      <sequence>
        <element name="AssertionConsumerServiceURL" type="anyURI"/>
        <element name="FederationTerminationNotificationProtocolProfile" type="anyURI"/>
        <element name="SingleLogoutProtocolProfile" type="anyURI"/>
        <element name="AuthnRequestsSigned" type="boolean"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="IDPDescriptor" type="lib:IDPDescriptorType"/>
<complexType name="IDPDescriptorType">
  <complexContent>
    <extension base="lib:ProviderDescriptorType">
      <sequence>
        <element name="SingleSignOnServiceURL" type="anyURI"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<!-- End provider metadata schema -->

</schema>

6 References


expert group/draft-liberty-tech-glossary-01.doc, April 2002.

[RFC2119] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels,

[RFC3280] R. Housley, Internet X.509 Public Key Infrastructure Certificate and
Certificate Revocation List (CRL) Profile,

Assertion Markup Language (SAML), http://www.oasis-
open.org/committees/security/docs/draft-sstc-core-27.pdf, OASIS,

http://www.w3.org/TR/xmlschema-1/, World Wide Web Consortium