Liberty Protocols and Schemas Specification

Draft Version 1.1-06

15th November 2002

Document Description: draft-liberty-architecture-protocols-schemas-v1.1-07

Notice

Liberty Alliance Project
Licensing Administrator
c/o IEEE-ISTO
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA

66 ## Editors

67 John D. Beatty, Sun Microsystems, Inc.
68 John Kemp, IEEE-ISTO

69 ## Contributors

70
71 The following Liberty Alliance Project Sponsor companies contributed to the development of
72 this specification:
73

ActivCard
American Express Travel Related Services
America Online, Inc.
Bank of America
Bell Canada
Cingular Wireless
Cisco Systems, Inc.
Citigroup
Communicator, Inc.
Consignia
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
NEC Corporation
Netegrity
NeuStar
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
SK Telecom
Sony Corporation
Sun Microsystems, Inc.
United Airlines
VeriSign, Inc.
Visa International
Vodafone Group Plc
Wave Systems

74

75 ## Document History

Revision	Date	Log
00	20-Oct-02	CRs implemented: 1096; 1113; 1116; 1117; 1128; 1129; 1130; 1134; 1135; 1136; 1137; 1141; 1142; 1143; 1149; 1152; 1154; 1146
01	28-Oct-02	CRs implemented: 1161/1160, 1132 1161/1160: • Section 3.3.1.1 – added RegisterNameIdentifierReturnURL element in text and schema. • Section 3.3.1.2 – updated example to match the above changes • Section 3.3.2.1 – added RegisterNameIdentifierReturnURL element in text and schema. • Section 3.3.2.2 – updated example to match the above changes • Section 4.2 – added new metadata element to the service provider metadata to specify the preferred RegisterNameIdentifier protocol 1132: • Section 3.2.1.1 – added AuthnContextComparisonType in text and changed

		schema. • Section 3.2.2.2 – updated example to match the above changes • Section 3.2.3 – updated text in processing rules to note this change.
02	29-Oct-02	1161: • Sections 3.3 –- 3.3.3 –- updated textual notes to reflect symmetry of RegisterNameIdentifier protocol between providers. • Section 4.2 –- added new metadata elements for RegisterNameIdentifierReturnURL and RegisterNameIdentifierProtocolProfile to the generic provider descriptor information. Removed them from Request/Response in 3.3.1/2 1132: • Made the AuthnContextComparisonType an enumerated attrib. 1158: • Section 32.4.2 –- added paragraphs discussing the logout of the Principal after federation termination. • Section 3.2.3 –- added a processing rule stating that an AuthnResponse MUST have a samlp:StatusCode of lib:FederationDoesNotExist if the SP is not currently federated with the IDP. General: • Updated namespace calls to /2002/12 for 1.1 namespace(s).
03	30-Oct-02	1153: • Updated all examples to have UTC compliant date/time data with a Z at the end of each date/time. 1133: • Section 3.3.3 - added a line to processing rules noting that messages SHOULD be signed, and noted that the signature MUST bbe verified if present. • Section 3.2.3 added a line to processing rules, noting that an error should be returned if authentication fails. • Section 3.4 - added a line to processing rules noting that messages SHOULD be signed, and noted that the signature MUST be verified if present. • Section 3.5.2.1 –- modified the processing rules to note that the signature SHOULD be verified, if present. 1146: • Slightly revised language in lines 965-968 in line with John B's comment about tightening language. 1152: • Section 4.2.1 - corrected example URIs for all profiles General: • Section 3.1.6 - added a reference for W3 XMLSchema dateTime • Fixed a spelling mistake 'occurred' line 654 • Added references to XML Signature and Canonicalization schema documents
04	31-Oct-02	1158:

		• Section 3.4.2 – modified language to state that a provider MAY invalidate a user's session as the result of federation termination. 1161: • Section 3.3.1.1 – added a new element <OldProvidedNameIdentifier> to the RegisterNameIdentifierRequest message. • Updated examples, and schema to match • Section 3.3 – added text to note that the <OldProvidedNameIdentifier> should contain the previous version of the <XXXProvidedNameIdentifier>
04	01-Nov-02	1161: • Section 3.2.2.4 – added paragraphs to specify that saml:AuthenticationMethod should be populated with athe Liberty context URI, when any AuthnContext is specified. Also noted that the SP must look to the Liberty AuthnContext (and ignore the saml attribute) if the saml:AuthenticationMethod is equal to the Liberty context URI. • Section 3.3 – added note to the effect that implementations may want to take account of possible propagation delays in processing of name identifier changes.1164: General: • Section 4.1 – removed a reference to profiles in the text • Section 3.2.3 – slightly tightened language about processing of AuthnContextComparisonType rules • All sections – removed CR comments Examples: • Updated AuthnRequest example
05	05-Nov-02	General edits for spelling, grammar, formatting.
06	15-Nov-02	1216: section 3.2.2.2 changed Assertion to AssertionType 1215: changed SAML reference to point to SAML 1.0 document 1214: corrected erroneous URIs lines 1186-1190 1213: corrected description of SingleSignOnProtocolProfile 1209: removed line 1010 regarding federation termination 1208: removed duplicate signing requirement 1207: saml:Success is now samlp:Success (line 908) 1205: the name identifier that the service provider should use when communicating with the identity provider. 1204: removed idp:Foo element from example 1203: added 'supported' regarding lib:NoAvailableIDP (lines 689-690) 1202: removed indentation 1201: reworded language regarding AuthnContextComparisonType (622-633) 1199: corrected URLs in examples 1198: created reference from ReauthenticateOnOrAfter to prior section discussing it 1197: removed incorrect indentation 1196: corrected examples

		1194: corrected typo – placed instead of place (366) 1192: reworded lines 283-285 to better deal with the specification of sub-second values. 1191: added further requirement on UTC time + Z specified lines 281-282 1190: reworded to be grammatically correct (256) 1189: added the word 'Individual' line 252 1188: Some profiles of the protocols contained in this specification may require a succinct 20-byte identifier. A provider MUST derive this by generating the SHA-1 hash of its URI-based identifier. 1187: removed section number reference (226) 1186: reworded lines 170-172. 1185: corrected typos line 424, 428 1184: corrected inconsistency in location of signing requirements 1212: corrected differences between RegisterNameIdentifier and other protocol profiles 1210: added RegisterNameIdentifierServiceURL, and corrected RegisterNameIdentifierServiceReturnURL

76

Table of Contents

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://projectliberty.org/schemas/core/2002/12`). This namespace is the default for instance fragments, type names, and element names in this document.
- The prefix `ac:` stands for the Liberty authentication context namespace (`http://projectliberty.org/schemas/authctx/2002/05`)
- 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 are also legal `xsd:boolean` values.

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://projectliberty.org/schemas/core/2002/12"
xmlns:lib="http://projectliberty.org/schemas/core/2002/12"
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://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.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
    <import namespace="http://projectliberty.org/schemas/authctx/2002/" schemaLocation="draft-
liberty-architecture-authentication-context-06.xsd"/>
```

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 de-federation).
- Single Logout: The protocol by which providers notify each other of logout events for single logout functionality.

3.1 General Requirements

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. A provider MUST derive this 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 non-public 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, verification of 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].

XML signatures SHOULD NOT be performed with any transforms other than:

- Enveloped Signature [XMLDsig]
- Exclusive XML Canonicalization [XMLCanon]

Receivers MUST NOT accept XML signatures created using other transforms without verification that the transforms used do not omit any part of the data to be signed from the signed byte stream. Receivers MAY reject any messages with transforms other than the set specified above. Senders MUST NOT send messages using other transforms without prior agreement as to their contents.

XML signatures in messages MUST use a proper URI fragment in the URI attribute of the Reference element to identify the signed element. The signer MUST NOT assume that the signed element will be at the root of the document during verification. It MUST be possible to validate the signature after adding or removing surrounding context for the profile in use (for example, the SOAP envelope, or the `<samlp:Response>` element). Implementers are encouraged to verify compliance with this requirement via empirical testing.

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.

3.1.7 Time Values

All Liberty time values have the type dateTime, which is built in to the W3C XML Schema Datatypes specification [Schema2]. Liberty time values MUST be expressed in UTC form, indicated by a “Z” immediately following the time portion of the value..

Liberty Requestors and Responders SHOULD NOT rely on other applications supporting time resolution finer than seconds, as implementations MAY ignore sub-second components specified in timestamp values. Implementations MUST NOT generate time instants that specify leap seconds.

3.1.8 Time Synchronization

Providers SHOULD NOT assume that other providers have clocks that are synchronized closer than one minute.

The Identity Provider SHOULD NOT include a NotBefore attribute on the Conditions element of the assertion it generates which contains the time the assertion was generated.

The Identity Provider SHOULD NOT include a NotOnOrAfter attribute on the Conditions element of the assertion it generates which is less than one minute later than the time when the assertion was generated.

The Service Provider SHOULD NOT terminate the principal's session based solely on the NotOnOrAfter attribute of the Conditions element of the assertion used to authenticate the principal. If the assertion was valid when the principal was authenticated, the principal SHOULD remain authenticated until one of the following occurs:

- LogoutNotification is received
- The user's session times out via normal means
- The ReauthenticateOnOrAfter time on the AuthenticationStatement used to authenticate the principal, if any, is reached

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 de-referenced 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.

The `<AuthnRequest>` message SHOULD be signed.

3.2.1.1 Element <AuthnRequest>

The `<AuthnRequest>` 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 re-authenticate 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]

This contains 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 placed on it by the request author. An example of such protection is confidentiality.

`id` [Optional]

Identifier used to identify this element in the signature. See section 3.1.5, Signature Verification for more information.

`AuthnContextComparisonType` [Optional]

If set to “exact”, then the identity provider is asked to match at least one of the specified `<AuthnContext>` elements exactly. This can also be set to “minimum”, which asks that the identity provider use a context that he feels is at least as good as any specified in the `<AuthnContext>` or “better”, which means that the they can use any context better than any that were supplied. If not specified, this is assumed to be “exact”.

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.

`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:

```
    <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"/>
                    <element ref=”lib:AuthnContextComparisonType” minOccurs=”0”/>
                </sequence>
                <attribute name="id" type="ID" use="optional"/>
            </extension>
        </complexContent>
    </complexType>
    <simpleType name=””AuthnContextComparisonType””>
        <restriction base=”string”>
            <enumeration value="exact"/>
            <enumeration value="minimum"/>
            <enumeration value="better"/>
        </restriction>
    </simpleType>
    <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"/>

            </choice>
        </complexType>
    </element>
```

3.2.1.2 Example

```
<lib:AuthnRequest id=”12345” RequestID="RPCUk2ll+GVz+t1lLURp51oFvJXk" MajorVersion="1"
MinorVersion="0" IssueInstant="2001-12-17T21:42:4Z”
xmlns:lib=”http://projectliberty.org/schemas/core/2002/12”>
    <ds:Signature> ... </ds:Signature>
    <lib:ProviderID>http://ServiceProvider.com</lib:ProviderID>
    <lib:ForceAuthn>false</lib:ForceAuthn>
```

```
    <lib:IsPassive>false</lib:IsPassive>
    <lib:Federate>true</Federate>
    <lib:ProtocolProfile>http://projectliberty.org/profiles/brws-post</lib:ProtocolProfile>
    <lib:AuthnContext>
        <lib:AuthnContextClassRef>http://projectliberty.org/schemas/authctx/classes/Password-
ProtectedTransport</lib:AuthnContextClassRef>
    </lib:AuthnContext>
    <lib:RelayState>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</lib:RelayState>
    <lib:AuthnContextComparisonType>exact</lib:AuthnContextComparisonType>
</lib: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. The <saml:Audience> element MUST be set to the intended recipient's ProviderID. The recipient MUST validate that it is the intended viewer before using 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 individually signed by the identity provider (that is, each assertion must contain a Signature element which signs only the assertion). It is RECOMMENDED that the signature be omitted from the <AuthnResponse> itself, but this is not forbidden.

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 <AssertionType>

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 id attribute is also included to facilitate signatures (see section 3.1.5, Signature Verification).

The schema fragment is as follows:

```
<complexType name="AssertionType">
    <complexContent>
        <extension base="saml:AssertionType">
            <attribute name="InResponseTo" type="saml:IDReferenceType"/>
            <attribute name="id" type="ID" use="optional"/>
        </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:

```
<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, or after which the service provider reauthenticates the Principal with the identity provider (as required in Section 3.2 above).

`SessionIndex` [Optional]

Indexes the particular session between the Principal and the identity provider under which this authentication statement is being issued. This value SHOULD be a small, positive integer but may be any string of text. However, this value MUST NOT be a globally unique value for the Principal's session at the Identity Provider.

When an `<AuthnContext>` element is specified, the **saml:AuthenticationMethod** attribute on the `<saml:AuthenticationStatement>` MUST be "http://projectliberty.org/schemas/authctx/2002/05".

When the Service Provider is processing a `<saml:AuthenticationStatement>` of type **lib:AuthenticationStatementType** and the `saml:AuthenticationMethod` attribute is "http://projectliberty.org/schemas/authctx/2002/05", the Service Provider MUST refer to the `<AuthnContext>` element and ignore the `saml:AuthenticationMethod` attribute.

The schema fragment is as follows:

```
    <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>
                        </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

```
<lib:AuthnResponse ResponseID="9hhuuja1bc744hGJn5Q9A5yvEIgS"
InResponseTo="Zon3WjJ2KL7j+bJu7MuIr4Pt2go5" MajorVersion="1" MinorVersion="0" IssueInstant="2002-10-
31T21:55:41Z">
    <samlp:Status>
        <samlp:StatusCode Value="samlp:Success"/>
    </samlp:Status>
    <saml:Assertion id=”12345” MajorVersion="1" MinorVersion="0" AssertionID="e06e5a28-bc80-4ba6-
9ecb-712949db686e" Issuer="http://IdentityProvider.com" IssueInstant="2001-12-17T09:30:47Z"
InResponseTo="4e7c3772-4fa4-4a0f-99e8-7d719ff6067c" xsi:type="AssertionType">
        <saml:Conditions NotBefore="2001-12-17T09:30:47Z" NotOnOrAfter="2001-12-17T09:35:47Z">
            <saml:AudienceRestrictionCondition>
                <saml:Audience>http://ServiceProvider.com</saml:Audience>
            </saml:AudienceRestrictionCondition>
        </saml:Conditions>
        <saml:AuthenticationStatement AuthenticationInstant="2001-12-17T09:30:47Z" SessionIndex="3"
ReauthenticateOnOrAfter="2001-12-17T11:30:47Z" xsi:type="AuthenticationStatementType"
AuthenticationMethod=”urn:oasis:names:tc:SAML:1.0:am:password”>
            <saml:Subject xsi:type="SubjectType">
                <saml:NameIdentifier>342ad3d8-93ee-4c68-be35-cc9e7db39e2b</saml:NameIdentifier>
                <IDPProvidedNameIdentifier>342ad3d8-93ee-4c68-be35-
cc9e7db39e2b</IDPProvidedNameIdentifier>
            </saml:Subject>
        </saml:AuthenticationStatement>
        <ds:Signature>...</ds:Signature>
    </saml:Assertion>
    <RelayState>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</RelayState>
</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.

- `<ds:Signature>`, if present, MUST be the signature of the service provider as specified by the `<ProviderID>`.
- If `<IsPassive>` is "true," 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."
- Failure to authenticate the Principal is indicated by a status code other than "Success." For failures, assertions MUST NOT appear in the `<AuthnResponse>`.
- 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." If <Federate> is "true" but the identity provider already has a previous federation on record for the Principal's identity at the service provider, the identity provider SHOULD treat the request as it would have if <Federate> was "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.
- If the service provider is not federated with the identity provider, and the `<Federate>` flag is not set, then the identity provider MUST return a `<samlp:StatusCode>` of `lib:FederationDoesNotExist` indicating that the service provider is not federated with this identity providerhen the identity provider.
- If `<AuthnContextComparisonType>` is specified and set to "exact", then the resulting authentication statement in the assertion (if any) MUST be the exact match of at least one of the authorization contexts specified.
- If `<AuthnContextComparisonType>` is specified and set to "minimum", then the resulting authentication statement in the assertion (if any) MUST be at least as strong (as deemed by the identity provider) as one of the authorization contexts specified.
- If `<AuthnContextComparisonType>` is specified and set to "better", then the resulting authentication statement in the assertion (if any) MUST be stronger (as deemed by the identity provider) than any specified in the supplied authorization contexts.

Additionally, if the `<AuthnContext>` element is specified, the identity provider MUST authenticate the Principal according to the following rules:

- If one or more `<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 `<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 one or more <AuthnContextStatementRef> elements are included, then the resulting authentication statement in the assertion (if any) MUST follow the rules specified in the <AuthnContextComparisonType>. 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 <AuthnResponse>.

If the service provider attempts to federate a Principal's identity with an identity provider, but another Principal's identity at the same service provider is already federated with the same identity provider, it will receive the other Principal's established name identifier in the <AuthnResponse>, rather than a new random one. The service provider MUST detect this error and handle it appropriately without leaving either Principal's identity at the service provider in an unusable state.

In some profiles, an intermediary is active between the service provider’s authentication request and the identity provider’s authentication response. Examples of an active intermediary are a user agent or client proxy that implement the "Liberty-Enabled Client and Proxy Profile" described in [LibertyBindProf].

NOTE: an active intermediary has the capability to return status codes to the service provider it interacts with. For example, the intermediary may be unable to contact an identity provider identified by the service provider, and the intermediary may return a status code to the service provider indicating that an error occurred. Status codes MUST be conveyed within <AuthnResponse> messages using the <samlp:Status> element. Specific values for specific cases are defined below.

For all profiles specifying an active intermediary, the profile specification must:

- Specify whether the <AuthnRequest> element sent from the service provider to the identity provider via the intermediary is wrapped in an <AuthnRequestEnvelope>. See section 3.2.4.
- Specify whether the <AuthnResponse> element sent from the identity provider to the service provider via the intermediary is wrapped in an <AuthnResponseEnvelope>. See section 3.2.5.

3.2.3.1.1 Processing Rules for Active Intermediaries

For all profiles specifying an active intermediary, the intermediary MUST follow these processing rules:

- If the profile specifies that the message sent from the service provider to the identity provider, via the intermediary, is wrapped in an <AuthnRequestEnvelope>:
- The intermediary MUST remove the enveloping <AuthnRequestEnvelope> before forwarding the <AuthnRequest> element to the identity provider.

- The intermediary MAY locally generate <AuthnResponse> elements and send them to the service provider using the <AssertionConsumerServiceURL> contained within the <AuthnRequestEnvelope>. Such <AuthnResponse> elements MUST NOT contain any <lib:Assertion> elements. These messages MAY be generated as a result of local errors on the intermediary, and should indicate the underlying reasons in the <samlp:Status> element in the <AuthnResponse>.

The following are error conditions for which <samlp:Status> values are defined in section 3.2.3.1.2:

- The identity provider cannot be reached
- There is no identity provider in common between the intermediary and the service provider
- If the profile specifies that the message from the identity provider to the service provider, via the intermediary, is wrapped in an <AuthnResponseEnvelope>:
- The intermediary MUST remove the enveloping <AuthnResponseEnvelope> before forwarding the <AuthnResponse> element to the service provider.
- The intermediary MUST send <AuthnResponse> messages received from the identity provider to the service provider using the <AssertionConsumerServiceURL> contained within the <AuthnResponseEnvelope> sent by the identity provider.

3.2.3.1.2 Status Code Values for Error Conditions

If an error occurs in the processing at the intermediary, the following status code values are defined for the <samlp:Status> element:

- lib:NoAvailableIDP: Used to indicate that none of the supported identity provider URLs from the <IDPList> can be resolved or that none of the identity providers is available.
- 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 <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. Processing rules are given in section 3.2.3.1.1. Note that the envelope is for consumption by the intermediary and is removed before the enveloped <AuthnRequest> element is forwarded to the identity provider.

3.2.4.1 Element <AuthnRequestEnvelope>

The authentication request envelope contains the following elements:

AuthnRequest [Required]

The authentication request contained in the envelope.

ProviderID [Required]

The ProviderID of the requestor.

ProviderName [Optional]

The human-readable name of the requestor.

`AssertionConsumerServiceURL` [Required]

The service provider's URL specifying where `<AuthnResponse>` elements, locally generated by the intermediary, are to be sent. See the processing rules for active intermediaries specified in section 3.2.3.1.1.

`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:

```
<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>
<complexType name="RequestEnvelopeType">
    <sequence>
        <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</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 identity provider's URI, to which authentication requests may be sent.

`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</AssertionConsume
rServiceURL>
    <IDPList>
      <IDPEntries>
        <IDPEntry>
            <ProviderID>http://IdentityProvider.com</ProviderID>
            <ProviderName>Identity Provider X</ProviderName>
            <Loc>http://www.IdentityProvider.com/liberty/sso</Loc>
        </IDPEntry>
      </IDPEntries>
      <GetComplete>https://ServiceProvider.com/idplist?id=604be136-fe91-441e-afb8-f88748ae3b8b
</GetComplete>
    </IDPList>
    <IsPassive>0</IsPassive>
</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. Processing rules are given in section 3.2.3.1.1. Note that the envelope is for consumption by the intermediary and is removed before the enveloped `<AuthnResponse>` element is forwarded to the service provider.

3.2.5.1 Element <AuthnResponseEnvelope>

The authentication response envelope contains the following elements:

`AuthnResponse` [Required]

The enveloped authentication response.

`AssertionConsumerServiceURL` [Required]

The service provider's URL where the authentication response should be sent. This element's value SHOULD be obtained from the element of the same name in the service provider's Provider Metadata.

The schema fragment is as follows:

```
<element name="AuthnResponseEnvelope" type="lib:AuthnResponseEnvelopeType"/>
<complexType name="AuthnResponseEnvelopeType">
    <complexContent>
        <extension base="lib:ResponseEnvelopeType">
            <sequence>
                <element ref="lib:AuthnResponse"/>
                <element name="AssertionConsumerServiceURL" type="anyURI"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="ResponseEnvelopeType">
    <sequence>
        <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

3.2.5.2 Example

```
<AuthnResponseEnvelope>
    <AuthnResponse> ... </AuthnResponse>
<AssertionConsumerServiceURL>
     http://ServiceProvider.com/lecp_assertion_consumer
</AssertionConsumerServiceURL>
</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, the service provider MAY register a different opaque handle with the identity provider. This opaque handle is termed the `SPProvidedNameIdentifier`.

After 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.

Either the service provider, or the identity provider MAY register a new name identifier for a Principal with each other at any time following federation. The name identifiers specified by either 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.

The same `<RegisterNameIdentifierRequest>` message may be sent by an identity provider, seeking to change the `IDPProvidedNameIdentifier` stored by the service provider.

The `<RegisterNameIdentifierRequest>` message SHOULD be signed.

3.3.1.1 Element <RegisterNameIdentifierRequest>

The elements of the message are as follows:

`ProviderID` [Required]

The provider's identifier.

`IDPProvidedNameIdentifier` [Required]

The name identifier the service provider should use when communicating with the identity provider.

`SPProvidedNameIdentifier` [Required]

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

`OldProvidedNameIdentifier` [Required]

In the case of either provider choosing to request a change of provided name identifiers, this element holds the previous version.

`id` [Optional]

Identifier used to identify this element in the signature. See section 3.1.5, Signature Verification.

The schema fragment is as follows:

```
<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"/>
                <element name="OldProvidedNameIdentifier" type="saml:NameIdentifierType"/>
            </sequence>
            <attribute name="id" type="ID" use="optional"/>
        </extension>
    </complexContent>
</complexType>
<element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
<element name="SPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
<element name="OldProvidedNameIdentifier" type="saml:NameIdentifierType"/>
```

3.3.1.2 Example

```
<RegisterNameIdentifierRequest id=”12345” RequestID="eb20e77f-d982-44f9-936e-dd135bf437d4"
MajorVersion="1" MinorVersion="0" IssueInstant="2001-12-17T09:30:47Z">
    <ds:Signature>...</ds:Signature>
    <ProviderID>http://ServiceProvider.com</ProviderID>
```

```
    <IDPProvidedNameIdentifier>342ad3d8-93ee-4c68-be35-cc9e7db39e2b</IDPProvidedNameIdentifier>
    <SPProvidedNameIdentifier>e958019a</SPProvidedNameIdentifier>
    <OldprovidedNameIdentifier>e895014a</OldProvidedNameIdentifier>
</RegisterNameIdentifierRequest>
```

3.3.2 Response

The responding 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.

This message SHOULD be signed.

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:

```
    <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

```
<RegisterNameIdentifierResponse ResponseID="74ffec0f-1165-4fa3-b088-3dd2c2388b91"
InResponseTo="eb20e77f-d982-44f9-936e-dd135bf437d4" MajorVersion="1" MinorVersion="0"
IssueInstant="2001-12-17T09:30:47Z" Recipient="http://ServiceProvider.com">
    <ds:Signature>...</ds:Signature>
    <samlp:Status>
        <samlp:StatusCode Value="samlp:Success"/>
    </samlp:Status>
</RegisterNameIdentifierResponse>
```

3.3.3 Processing Rules

The recipient MUST validate any signature present on the message. To be considered valid, the signature provided 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 `<samlp:Status>` with a `<samlp:StatusCode>` of `lib:FederationDoesNotExist`.

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

Either provider MAY choose to change their provided name identifier. In this case, the `<OldProvidedNameIdentifier>` should contain the previous version of the identifier.

Changes to these identifiers may take a potentially significant amount of time to propagate through the systems at both the sender and the receiver. Implementations MAY wish to allow each party to accept either identifier for some period of time following the successful completion of a name identifier change. Not doing so could result in the inability of the Principal to access resources.

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. Reasonable best-effort delivery MUST be employed by all providers sending this message.

3.4.1 Message

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

The `<FederationTerminationNotification>` message SHOULD be signed.

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 for which federation is terminating. This name identifier MUST be equal to the `<saml:NameIdentifier>` element (and its included attributes) agreed upon earlier between the two communicating providers.

`id` [Optional]

Identifier used to identify this element in the signature. See section 3.1.5, Signature Verification.

The schema fragment is as follows:

```
    <element name="FederationTerminationNotification"
type="lib:FederationTerminationNotificationType"/>
    <complexType name="FederationTerminationNotificationType">
        <complexContent>
            <extension base="samlp:RequestAbstractType">
                <sequence>
                    <element ref="lib:ProviderID"/>
                    <element ref="saml:NameIdentifier"/>
                </sequence>
                <attribute name="id" type="ID" use="optional"/>
            </extension>
        </complexContent>
    </complexType>
```

3.4.1.2 Example

```
<FederationTerminationNotification id="12345" RequestID="9ec2-eb65-4bce-ab8f-4becdf229815"
MajorVersion="1" MinorVersion="0" IssueInstant="2001-12-17T09:30:47Z">
    <ds:Signature>...</ds:Signature>
```

```
    <ProviderID>http://IdentityProvider.com</ProviderID>
    <saml:NameIdentifier>e958019a</saml:NameIdentifier>
</FederationTerminationNotification>
```

3.4.2 Processing Rules

The receiving provider MUST validate any signature present on the message. The signature on the message MUST be the signature of the `<ProviderID>` contained in the message. If the signature is not valid, the provider MUST ignore the message.

If a provider receives a federation termination notification message that refers to a federation that does not exist from the perspective of the provider, the provider MUST ignore the message. Otherwise, the provider MAY perform any maintenance with the knowledge that the federation has been terminated.

A provider MAY choose to invalidate the session of a user for whom federation has been terminated.

3.5 Single Logout Protocol

The Single Logout Protocol provides a one-way asynchronous message exchange protocol by which all sessions authenticated by a particular identity provider are near-simultaneously terminated. The Single Logout Protocol is used either when a Principal logs out at a service provider or when the Principal logs out at an identity provider.

When the Principal invokes the single logout process at a service provider, the service provider MUST send a `<LogoutNotification>` message to the identity provider that provided the authentication service for the session.

When either the Principal invokes a logout at the identity provider or a service provider sends a logout notification to the identity provider specifying that Principal, the identity provider MUST send a `<LogoutNotification>` message to each service provider to which it provided authentication assertions in the current session with the Principal, with the exception of the service provider that sent the `<LogoutNotification>` message to the Identity Provider.

This notification is a one-way asynchronous message, and reasonable best-effort delivery SHOULD be employed by both service providers and identity providers.

3.5.1 Message

The `<LogoutNotification>` message indicates to the message receiver that a Principal's session was terminated. The message includes an optional `<SessionIndex>` element that MUST be specified if and only if the authentication statement in the assertion that the service provider used in establishing the session with the Principal contained a `SessionIndex` attribute. This message SHOULD be signed.

3.5.1.1 Element <LogoutNotification>

`NameIdentifier` [Required]

The name identifier of the Principal that logged out. This name identifier MUST be equal to the `<saml:NameIdentifier>` element (including the equality of contained attributes) agreed upon between the two communicating providers.

`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. If a `<SessionIndex>` element was present in the authentication statement, an identical `<SessionIndex>` MUST be present in the `<LogoutNotification>`. If no `<SessionIndex>` element was present in the authentication statement, the `<SessionIndex>` MUST be omitted from the `<LogoutNotification>`.

`id` [Optional]

Identifier used to identify this element in the signature. See section 3.1.5, Signature Verification.

The schema fragment is as follows:

```
    <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>
                <attribute name="id" type="ID" use="optional"/>
            </extension>
        </complexContent>
    </complexType>
```

3.5.1.2 Example

```
<LogoutNotification id=”12345” RequestID="47693d03-7c33-4d65-931f-ddeb19fa6a73" MajorVersion="1"
MinorVersion="0" IssueInstant="2001-12-17T09:30:47Z">
    <ds:Signature>...</ds:Signature>
    <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 any signature present 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 providers 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. This value MUST be the SHA-1 hash of the ProviderID. Since the SHA-1 hash output may include arbitrary binary data, the ProviderSuccinctID element value is encoded into hexadecimal (according to the hexBinary XML Schema type) when included in an XML metadata document. Since the SHA-1 hash output always consists of 20 bytes of binary data, its hexBinary encoding will always consist of 40 hexadecimal digits.

`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.

`FederationTerminationNotificationProtocolProfile` [Optional]

The Federation Termination Notification Protocol profiles supported by the provider. Each value of the element MUST contain a valid Federation Termination Notification Protocol

profile identification URI. The absence of this element SHALL mean that provider does not support any profile of the Federation Termination Notification Protocol.

`SingleLogoutProtocolProfile` [Optional]

The Single Logout Protocol profiles supported by the provider. Each element MUST contain a valid Single Logout Protocol profile identification URI. The absence of this element SHALL mean that the provider does not support any profile of the Single Logout Protocol.

`RegisterNameIdentifierProtocolProfile` [Optional]

The provider's preferred Register Name Identifier Protocol profile, which should be used by other providers when registering a new identifier. Each element MUST contain a valid Register Name Identifier Protocol profile identification URI. The absence of this element SHALL mean that the provider does not support any profile of the Register Name Identifier Protocol.

`RegisterNameIdentifierServiceURL` [Optional]

The URL used for user-agent-based Register Name Identifier Protocol profiles.

`RegisterNameIdentifierServiceReturnURL` [Optional]

The provider's redirecting URL for use after HTTP name registration has taken place.

The schema fragment is as follows:

```
    <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"/>
            <element name="FederationTerminationNotificationProtocolProfile" type="anyURI"
minOccurs="0"/>
            <element name="SingleLogoutProtocolProfile" type="anyURI" minOccurs="0"/>
            <element name="RegisterNameIdentifierProtocolProfile" type="anyURI" minOccurs="0"/>
            <element name="RegisterNameIdentifierServiceURL" type="anyURI" minOccurs="0"/>
            <element name="RegisterNameIdentifierServiceReturnURL" 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.

`AuthnRequestsSigned` [Required]

Specifies whether the service provider will always sign authentication requests it sends to the identity provider.

The schema fragment is as follows:

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

```
                </sequence>
            </extension>
        </complexContent>
    </complexType>
```

4.2.1 Example

```
<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>
    <SingleLogoutServiceReturnURL>http://ServiceProvider.com/liberty/slo_return</SingleLogoutServiceR
eturnURL>
    <FederationTerminationServiceURL>http://ServiceProvider.com/liberty/term</FederationTerminationSe
rviceURL>
    <FederationTerminationServiceReturnURL>http://ServiceProvider.com/liberty/term return</Federation
TerminationServiceReturnURL>
    <FederationTerminationNotificationProtocolProfile>http://projectliberty.org/profiles/fedterm-idp-
soap</FederationTerminationNotificationProtocolProfile>
    <FederationTerminationNotificationProtocolProfile>http://projectliberty.org/profiles/fedterm-idp-
http</FederationTerminationNotificationProtocolProfile>
    <SingleLogoutProtocolProfile>http://projectliberty.org/profiles/slo-idp-
http</SingleLogoutProtocolProfile>
    <RegisterNameIdentifierProtocolProfile>http://projectliberty.org/profiles/rni-idp-soap</
RegisterNameIdentifierProtocolProfile>
    <RegisterNameIdentifierServiceURL>http://Provider.com/liberty/register name</RegisterNameIdentifi
erServiceURL>
    <RegisterNameIdentifierServiceReturnURL>http://Provider.com/liberty/ReturnURL</
RegisterNameIdentifierReturnServiceURL>
    <AssertionConsumerServiceURL>http://ServiceProvider.com/liberty/assertion_consumer</AssertionCons
umerServiceURL>
    <AuthnRequestsSigned>1</AuthnRequestsSigned>
</SPDescriptor>
```

4.3 Identity Provider Descriptor

The additional identity provider-specific metadata are as follows:

`SingleSignOnServiceURL` [Required]

The identity provider's URL for accepting authentication requests for the Single Sign-On and Federation Protocol.

`SingleSignOnProtocolProfile` [Required]

The Single Sign-On Protocol profiles supported by the provider. Each element MUST contain a valid Single Sign-On Protocol profile identification URI.

The schema fragment is as follows:

```
    <element name="IDPDescriptor" type="lib:IDPDescriptorType"/>
    <complexType name="IDPDescriptorType">
        <complexContent>
            <extension base="lib:ProviderDescriptorType">
                <sequence>
                    <element name="SingleSignOnServiceURL" type="anyURI"/>
                    <element name="SingleSignOnProtocolProfile" 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>
    <SingleLogoutServiceURL>http://IdentityProvider.com/liberty/slo</SingleLogoutServiceURL>
```

```
    <SingleLogoutServiceReturnURL>http://IdentityProvider.com/liberty/slo return</SingleLogoutService
ReturnURL>
    <FederationTerminationServiceURL>http://IdentityProvider.com/liberty/term</FederationTerminationS
erviceURL>
    <FederationTerminationServiceReturnURL>http://IdentityProvider.com/liberty/term_return</Federatio
nTerminationServiceReturnURL>
    <SingleSignOnServiceURL>http://IdentityProvider.com/liberty/sso</SingleSignOnServiceURL>
    <SingleSignOnProtocolProfile>http://projectliberty.org/profiles/brws-

</IDPDescriptor>
```

5 Schema Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://projectliberty.org/schemas/core/2002/12"
xmlns:lib="http://projectliberty.org/schemas/core/2002/12"
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://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.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
    <import namespace="http://projectliberty.org/schemas/authctx/2002/05" schemaLocation="draft-
liberty-architecture-authentication-context-07.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"/>
                    <element ref="lib:AuthnContextComparisonType" minOccurs="0"/>
                </sequence>
                <attribute name="id" type="ID" use="optional"/>
            </extension>
        </complexContent>
    </complexType>
    <simpleType name="AuthnContextComparisonType">
        <restriction base="string">
            <enumeration value="exact"/>
            <enumeration value="minimum"/>
            <enumeration value="better"/>
        </restriction>
    </simpleType>
    <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"/>
            </choice>
        </complexType>
    </element>
    <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>
<complexType name="RequestEnvelopeType">
    <sequence>
        <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
<element name="AuthnResponseEnvelope" type="lib:AuthnResponseEnvelopeType"/>
<complexType name="AuthnResponseEnvelopeType">
    <complexContent>
        <extension base="lib:ResponseEnvelopeType">
            <sequence>
                <element ref="lib:AuthnResponse"/>
                <element name="AssertionConsumerServiceURL" type="anyURI"/>
            </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>
                    </complexType>
                </element>
            </sequence>
            <attribute name="ReauthenticateOnOrAfter" type="dateTime" use="optional"/>
            <attribute name="SessionIndex" type="string" use="optional"/>
        </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"/>
                <element name="OldProvidedNameIdentifier" type="saml:NameIdentifierType"/>
            </sequence>
```

```
            <attribute name="id" type="ID" use="optional"/>
        </extension>
    </complexContent>
</complexType>
<element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
<element name="SPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
<element name="RegisterNameIdentifierResponse" type="lib:StatusResponseType"/>
<element name="OldProvidedNameIdentifier" type="saml:NameIdentifierType"/>
<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>
            <attribute name="id" type="ID" use="optional"/>
        </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>
            <attribute name="id" type="ID" use="optional"/>
        </extension>
    </complexContent>
</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"/>
            <attribute name="id" type="ID" use="optional"/>
        </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:IDPEntry" maxOccurs="unbounded"/>
```

```
            </sequence>
        </complexType>
    </element>
    <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"/>
            <element name="FederationTerminationNotificationProtocolProfile" type="anyURI"
minOccurs="0"/>
            <element name="SingleLogoutProtocolProfile" type="anyURI" minOccurs="0"/>
            <element name=”RegisterNameIdentifierProtocolProfile” type=”anyURI” minOccurs=”0”/>
            <element name="RegisterNameIdentifierServiceURL" type="anyURI" minOccurs=”0”/>
            <element name="RegisterNameIdentifierServiceReturnURL" 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="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"/>
                    <element name="SingleSignOnProtocolProfile" type="anyURI"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- End provider metadata schema -->
</schema>
```

6 References

[LibertyBindProf] J. Rouault, “Liberty Bindings and Profiles Specification.”, http://projectliberty.org/specs/liberty-architecture-bindings-profiles-v1.0.pdf, October 2002

[LibertyGloss] H. Mauldin, “Liberty Glossary,” http://projectliberty.org/specs/liberty-tech-glossary-v1.0.pdf, October 2002.

[RFC2119] S. Bradner, “Key words for use in RFCs to Indicate Requirement Levels”, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[RFC3280] R. Housley, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, http://www.ietf.org/rfc/rfc3280.txt, IETF RFC 3280, April 2002.

[SAMLCore] P. Hallam-Baker et al., “Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)”, http://www.oasis-open.org/committees/security/docs/cs-sstc-core-01.pdf, OASIS, May 2002.

[Schema1] H. S. Thompson et al., "XML Schema Part 1: Structures," http://www.w3.org/TR/xmlschema-1/, World Wide Web Consortium Recommendation, May 2001.

[Schema2] H. S. Thompson et al., "XML Schema Part 1: Structures," http://www.w3.org/TR/xmlschema-2/, World Wide Web Consortium Recommendation, May 2001.

[XMLDsig] D. Eastlake, J. Reagle, D. Solo, M. Bartel, J. Boyer, B. Fox, and E. Simon, "XML-Signature Syntax and Processing", IETF Draft/W3C Candidate Recommendation, http://www.w3.org/TR/xmldsig-core, 31 October 2000.

[XMLCanon] J. Boyer, "Canonical XML, IETF Draft/W3C Candidate Recommendation", http://www.w3.org/xml-c14n, 15 March 2001.