Liberty Metadata Description and Discovery Specification

Version: v1.1

Editors:
Peter Davis, NeuStar, Inc.

Contributors:
Paul Madsen, Entrust, Inc.
Jeff Hodges, Sun Microsystems, Inc.
Bronislav Kavsan, RSA Security, Inc.
Scott Cantor, Internet2

Abstract:

This document details the metadata schema and methods of resolution for discovering the location of metadata instances for the Liberty Identity Federation Framework

Filename: liberty-metadata-v1.1.pdf

Notice

This document has been prepared by Sponsors of the Liberty Alliance. Permission is hereby granted to use the document 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 certain elements of this document may require licenses under third party intellectual property rights, including without limitation, patent rights. The Sponsors of and any other contributors to the Specification are not, and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights. **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 of third party intellectual property rights, and fitness for a particular purpose.** Implementors of this Specification are advised to review the Liberty Alliance Project's website (http://www.projectliberty.org/) for information concerning any Necessary Claims Disclosure Notices that have been received by the Liberty Alliance Management Board.

Liberty Alliance Project
Licensing Administrator
c/o IEEE-ISTO
445 Hoes Lane
Piscataway, NJ 08855-1331, USA
info@projectliberty.org

Contents

1. Introduction

Within **ID-FF** version 1.1 specification [LibertyProtSchema1.1] of the Liberty Alliance protocols [LibertyProtSchema], basic metadata were exchanged out-of-band between entities. This specification more formally describes metadata, as well as protocols to facilitate real-time requests for this data allowing for more spontaneous conversations between Liberty enabled entities.

There are three primary functions for this metadata:

- declarations of entity metadata for providers, principals and devices, and affiliations
- entity trust metadata, which enables entities to cast business decisions based on the characteristic trust information provided in this class, conveyed through document signature(s), server authenticated protected channel delivery of the instance using TLS [RFC2246] as amended by [RFC3546], DNS zone signatures, and, optionally, additional material that publishers may convey within the `Extension` and `AdditionalMetaLocation` elements
- origin and document verification through signature use in (server authenticated) HTTPS retrieval of the instance documents, DNS signatures, and document level signatures

This document presents extensions to the model for metadata described in Liberty ID-FF versions 1.1 to better support ad-hoc interactions between entities. The location of cryptographic keys in a distributed-computing architecture that contains "arms-length" peer domains presents an opportunity for some fresh thinking. Conventional solutions to this problem fail to fully exploit the potential of the evolving Web Services architecture to minimize administrative costs. Liberty ID-FF version 1.2 [LibertyProtSchema], ID-WSF and ID-SIS set of specification [LibertyIDFFOverview] operations between previously unintroduced parties will benefit from any mechanisms that simplify how keying material and service interface points can be discovered, leading to mechanisms for trust establishment and services invocations in both direct and indirect means.

1.1. Notation and Conventions

This specification uses schema documents conforming to W3C XML Schema [Schema1] and normative text to describe the syntax and semantics of XML-encoded protocol messages.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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.

Within this document, a *publisher* is the subject of, or authorized representing party for, the subject of the instance document, as referenced by `providerID` and a *consumer* is the entity resolving, retrieving, or otherwise processing the instance as a relying party to its information.

1.2. Overview

The metadata protocols and schemas specified in this document will enable two Liberty-enabled entities to exchange or request cryptographic keys, service endpoints information, and protocol and profile support in real time, allowing dynamic interactions between these parties, eliminating the need for out-of-band negotiations to have occurred a-priori. The addition of interactions between separate authentication authorities and identity chaining in the Liberty **ID-WSF** will depend upon this exchange, as portions of a principle's identity may be previously established outside the range of providers established agreements.

2. Metadata Schema

The metadata schema allows several methods of representation:

- A document expressing metadata describing an entity, referenced by a `providerID`, acting in the role of a Service Provider or an Identity Provider, or both, within the `EntityDescriptor Node`
- As a single instance document expressing metadata describing multiple entities, each referenced by a `providerID`, each entity acting as declared above. The metadata for each entity is contained within separate `EntityDescriptor` nodes, each being an immediate descendant of the plural `EntitiesDescriptor` node
- As a single instance document describing an affiliation (a set of entities) collectively identified as `providerID` (located with the `EntityDescriptor`parent), which in turn enumerates each entity member by it's own `providerID` and maintained by an entity referenced by its `affiliationOwnerID`. Each member's metadata is then located by the methods provided in this specification.

The first two forms may also be expressed as multiple documents, involving additional metadata, which MAY be of a namespace `urn:liberty:metadata:2003-08` (the default), or another namespace, as specified by the element `Location`'s corresponding `namespace` attribute. Additionally, the document location(s) may be identified by multiple NAPTR resource records.

2.1. Schema Declarations

The metadata schema is constructed to allow an entity, referenced by one or more `providerID`'s, to publish single or multiple schema instances to describe their identity services architecture.

The primary container for a published document is either `EntityDescriptor` or the plural form `EntitiesDesciptor` (used when an affiliated set of entities chooses to publish a consolidated set of metadata documents as one).

The expected immediate child nodes of `EntityDescriptor` are one or more of:

- `SPDescriptor`
- `IDPDescriptor`

or one of:

- `AffiliationDescriptor`

which are described below. Additionally, an extension point `Extension` is provided in order to convey additional metadata.

2.1.1. Namespaces in Metadata

The following namespace declarations are used to complete the metadata schema:

- `ds:` is described by the W3C XML Signature [XMLDsig] schema
- `saml:` is described by the OASIS Security Services SAML 1.1 Assertion [SAMLCore11] schema

In addition, the Liberty Utility Schema is included allowing the common `Extension` element that is used throughout the Liberty Specifications suite

Schema Fragment:

```
<xs:schema targetNamespace="urn:liberty:metadata:2003-08"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   xmlns="urn:liberty:metadata:2003-08"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified"
   version="1.0">
  <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
  <xs:import namespace="urn:oasis:names:tc:SAML:1.0:assertion"
     schemaLocation="oasis-sstc-saml-schema-assertion-1.1.xsd"/>
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
     schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xs:include schemaLocation="liberty-utility-v1.1.xsd"/>
```

2.1.2. Datatype entityIDType

The datatype `entityIDType` restricts the XML data to a length of 1024 bytes.

Additionally, the `entityIDType` structure is defined by the following BNF, derived from *URI* Specification [RFC2396] as modified by [RFC2732]

```
BNF for Liberty entityIdentifiers
    # constraint on absoluteURI
    entityID     = absoluteURI [ "#" fragment ]
    absoluteURI   = scheme ":" ( hier_part | opaque_part )

    # constraint on hier_part (net_path only)
    hier_part    = net_path [ "?" query ]
    opaque_part   = uric_no_slash *uric

    uric_no_slash = unreserved | escaped | ";" | "?" | ":" | "@" |
                "&" | "=" | "+" | "$" | ","

    net_path     = "//" authority [ abs_path ]
    abs_path     = "/"  path_segments

    ; pragmatically, scheme SHOULD be an officially IANA registered URI scheme
    ; http://www.iana.org/assignments/uri-schemes
    scheme      = alpha *( alpha | digit | "+" | "-" | "." )

    authority    = server | reg_name

    reg_name     = 1*( unreserved | escaped | "$" | "," |
                   ";" | ":" | "@" | "&" | "=" | "+" )

    server      = [ [ userinfo "@" ] hostport ]
    userinfo     = *( unreserved | escaped |
                  ";" | ":" | "&" | "=" | "+" | "$" | "," )

    hostport     = host [ ":" port ]
    ; constraint on host (no ipAddress)
    host        = hostname
    hostname     = *( domainlabel "." ) toplabel [ "." ]
    domainlabel   = alphanum | alphanum *( alphanum | "-" ) alphanum
    toplabel     = alpha | alpha *( alphanum | "-" ) alphanum
    port        = *digit
```

```
    path        = [ abs_path | opaque_part ]
    path_segments = segment *( "/" segment )
    segment     = *pchar *( ";" param )
    param       = *pchar
    pchar       = unreserved | escaped |
                  ":" | "@" | "&" | "=" | "+" | "$" | ","

    query       = *uric

    fragment    = *uric

    uric        = reserved | unreserved | escaped
    reserved    = ";" | "/" | "?" | ":" | "@" | "&" | "=" | "+" |
                  "$" | ","
    unreserved  = alphanum | mark
    mark        = "-" | "_" | "." | "!" | "~" | "*" | "'" |
                  "(" | ")"

    escaped     = "%" hex hex
    hex         = digit | "A" | "B" | "C" | "D" | "E" | "F" |
                          "a" | "b" | "c" | "d" | "e" | "f"

    alphanum    = alpha | digit
    alpha       = lowalpha | upalpha

    lowalpha = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |
               "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |
               "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
    upalpha  = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
               "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
               "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"
    digit    = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
               "8" | "9"
```

The schema fragment for `entityIDType`:

```
<xsd:simpleType name="entityIDType">
   <xsd:restriction base="xsd:anyURI">
      <xsd:maxLength id="maxlengid" value="1024"/>
   </xsd:restriction>
</xsd:simpleType>
```

2.1.3. Common Attributes

Several common attributes are defined and generally used throughout the schema:

- `libertyPrincipalIdentifier` of type `entityIDType` used to provide a pointer to contact an entity's metadata which MAY be dereferencable

- `providerID` of type `entityIDType` indicates the providerID of the entity described by the descendants of the node

- `validUntil` of type `dateTime` indicates the expiration date and time of the node (and its descendants). If `dateTime` expressions evaluate to nonequivalent values, parsers MUST adhere to the most restrictive value (the earliest dateTime).

- `cacheDuration` of type `duration` indicates the maximum elapsed time a consumer may cache the metadata document (or fragment). Consistent with the `validUntil` attribute, the most restrictive value MUST be used when conflicting cache directives occur

Publishers MUST provide either a `validUntil` or `cacheDuration` attribute when publishing metadata. Since this directive is available at both the top-level EntityDescriptor and it immediate descendants, care should taken in selecting expiration settings. It is RECOMMENDED that publishers express document expiration at the `EntityDescriptor` or `AffiliationDescriptor` element only, and not on the child nodes.

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 entities SHOULD NOT rely on other applications supporting time resolution finer than seconds, as implementations MAY ignore fractional second components specified in timestamp values. Implementations MUST NOT generate time instants that specify leap seconds.

The consumer MAY reset the retrieval dateTime, effectively resetting the duration clock (see Section 5.2) if consumers send an *HTTP (1.1)* [RFC2616] request to the publisher URL with a header *If-Modified-Since: [last retrieval dateTime]*, the publisher server returns a *304 Not-Modified* response, and the publisher expresses the expiration as a `cacheDuration`,

The schema fragment for the common attributes:

```
<xs:attribute name="libertyPrincipalIdentifier" type="entityIDType"/>
<xs:attribute name="providerID" type="entiyIDType"/>
<xs:attribute name="validUntil" type="xs:dateTime"/>
<xs:attribute name="cacheDuration" type="xs:duration"/>
```

2.1.4. Common DataTypes

There are several common datatypes defined globally, and used throughout the schema:

2.1.4.1. organizationType Data Type

The `organizationType` datatype provides some basic information consumers may require when interacting with a principal:

- `OrganizationName` of type `string` [Required, 1-many]: a localizable ([XML] Section 2.12 Language Identification) Organizational Name of the entity, generally the complete Organization Legal name

- `OrganizationDisplayName` of type `string` [Required, 1-many]: a localizable organization name suitable for display to a principal

- `OrganizationURL` of type `anyURI` [Required, 1-many]: a localizable URL of the organization suitable for dereferencing by a user-agent, which may be used for directing a principal for additional information on the entity

Localized strings SHOULD be used when present in the metadata instance, and the preferred language of the target entity is known by the consumer

```
<xs:complexType name="organizationType">
   <xs:sequence>
      <xs:element maxOccurs="unbounded" name="OrganizationName" type="organizationNameType"/>
      <xs:element maxOccurs="unbounded" name="OrganizationDisplayName"
         type="organizationDisplayNameType"/>
      <xs:element maxOccurs="unbounded" name="OrganizationURL" type="localizedURIType"/>
      <xs:element minOccurs="0" ref="Extension"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="organizationNameType">
   <xs:simpleContent>
      <xs:extension base="xs:string">
         <xs:attribute ref="xml:lang"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
<xs:complexType name="organizationDisplayNameTyp e">
   <xs:simpleContent>
      <xs:extension base="xs:string">
         <xs:attribute ref="xml:lang" use="required"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
<xs:complexType name="localizedURIType">
   <xs:simpleContent>
      <xs:extension base="xs:anyURI">
         <xs:attribute ref="xml:lang" use="required"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

2.1.4.2. contactType Data Type

The contactType data type conveys general contact information for human-to-human contact regarding an entity. It is defined with the following attributes:

- `libertyPrincipalIdentifier` [Optional]: a Principal's dereferencable nameIdentifier of type entityIDType which may point to an online instance of the person's PIP profile
- `contactType` [Required]: the type of contact, which may be one of: `technical`, `administrative`, `billing`, or `other`. The default value is `technical`

The elements defined by this type:

- `Company` [Optional, 0-1]: The company name of type `xs:string`, by which the cited individual is employed for the purposes relating to the instance document
- `GivenName` [Optional, 0-1]: The given name of the contact of type `xs:string`
- `SurName` [Optional, 0-1]: The surname of the contact of type `xs:string`
- `EmailAddress` [Optional, 0-many]: The email address of the contact of type `xs:anyURI`
- `TelephoneNumber` [Optional, 0-many]: The contact's telephone number of type `xs:string`

The schema fragment for `contactType`:

```
<xs:complexType name="contactType">
   <xs:sequence>
      <xs:element maxOccurs="1" minOccurs="0" name="Company" type="xs:string"/>
      <xs:element maxOccurs="1" minOccurs="0" name="GivenName" type="xs:string"/>
      <xs:element maxOccurs="1" minOccurs="0" name="SurName" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="EmailAddress" type="xs:anyURI"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="TelephoneNumber" type="xs:string"/>
      <xs:element minOccurs="0" ref="Extension"/>
   </xs:sequence>
   <xs:attribute name="libertyPrincipalIdentifier" use="optional"/>
   <xs:attribute name="contactType" type="attr.contactType" use="required"/>
</xs:complexType>
<xs:simpleType name="attr.contactType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="technical"/>
      <xs:enumeration value="administrative"/>
      <xs:enumeration value="billing"/>
      <xs:enumeration value="other"/>
   </xs:restriction>
</xs:simpleType>
```

2.1.4.3. keyDescriptorType Complex Type

The elements of type `keyDescriptorType` convey to a consumer two cryptographic metadata statements:

- encryption preferences described by `EncryptionMethod` [Optional, 0-1], whose valid URI values are defined in [xmlenc-core], and
  `KeySize` [Optional, 0-1] which may optionally constrain the length of keys used by the consumer when interacting with another entity

- Key Material information located in `ds:KeyInfo` [Optional, 0-1] as described by [XMLDsig]

The schema fragment for `keyDescriptorType`:

```
<xs:complexType name="keyDescriptorType">
   <xs:sequence>
      <xs:element minOccurs="0" name="EncryptionMethod" type="xs:anyURI"/>
      <xs:element minOccurs="0" name="KeySize" type="xs:integer"/>
      <xs:element minOccurs="0" ref="ds:KeyInfo"/>
      <xs:element minOccurs="0" ref="Extension"/>
   </xs:sequence>
   <xs:attribute name="use" type="keyTypes use="required"/>
</xs:complexType>
```

The `KeyDescriptor` includes the required attribute `use` of type `keyTypes`. `use` may have values of `encryption` or `signing`:

```
<xs:simpleType name="keyTypes">
   <xs:restriction base="xs:string">
      <xs:enumeration value="encryption"/>
      <xs:enumeration value="signing"/>
   </xs:restriction>
</xs:simpleType>
```

2.1.4.4. providerDescriptorType Complex Type

The providerDescriptorType is a utility type, which describes generic metadata for any Liberty-enabled entity with attributes that include:

- `id` [Optional]: The fragment identifier of the instance node (required if the node is signed as described in Section 5.1).
- `validUntil` [Optional]: The dateTime the fragment expires. Processing rules are described in Section 2.1.3 [7].
- `cacheDuration` [Optional]: The maximum duration a consumer may cache the fragment. Processing rules are described in Section 2.1.3 [7].
- `protocolSupportEnumeration` [Required] describes the protocol release supported by the entity described by `providerID`. NMTOKENS type allows the enumeration of a set of Liberty ID-FF protocol releases which the interfaces described within MUST support. The datatype of the tokens MUST be URNs (presently `http://projectliberty.org/schemas/core/2002/12` for release ID-FF 1.1 and `urn:liberty:iff:2003-08` for release ID-FF 1.2). Subsequent releases of ID-FF shall express protocol support using the defined `nameSpace` attribute of the corresponding ID-FF schema.
  When an entity supports both ID-FF 1.1 and ID-FF 1.2 protocols, it SHOULD publish a ID-FF 1.1 valid instance and make reference to it within `AddtionalMetaLocation`, using the appropriate corresponding namespace identifier for that schema. Metadata consumers MUST retrieve (or otherwise obtain) this instance if they intend to use the protocols of ID-FF 1.1. If publisher entities support both protocols on the same `SoapEndPoint`, they MAY publish one document which describes both protocols uniformly, citing both protocols in the attribute `protocolSupportEnumeration` or they MAY make reference to it using `AdditionalMetaLocation`. Consumers in possession of an ID-FF 1.1 provider's metadata obtained in an "Out-of-Band" manner as described in that version of the specification, MAY continue to use this instance, but SHOULD check for a newer version whenever possible.

The elements describing the entity include:

- `KeyDescriptor` [Optional, 0-many] expresses a set of keying material and key metadata which the cooresponding entity `providerID` will use within Liberty protocols and interactions.
- `SoapEndpoint` [Required, 1] The provider's SOAP endpoint URI.
- `SingleLogoutServiceURL` [Optional, 1] The URL used for user-agent-based Single Logout Protocol profiles.
- `SingleLogoutServiceReturnURL` [Optional, 0-1] The URL to which the provider redirects at the end of user-agent-based Single Logout Protocol profiles.
- `FederationTerminationServiceURL` [Optional, 0-1] The URL used for user-agent-based Federation Termination Notification Protocol profiles.
- `FederationTerminationServiceReturnURL` [Optional, 0-1] The URL to which the provider redirects at the end of user-agent-based Federation Termination Notification Protocol profiles.
- `FederationTerminationNotificationProtocolProfile` [Optional, 0-many] 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 as defined in [LibertyBindProf]. The absence of this element SHALL mean that provider does not support any profile of the Federation Termination Notification Protocol.

- `SingleLogoutProtocolProfile` [Optional, 0-many] 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, 0-many] 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 as defined in [LibertyBindProf]. The absence of this element SHALL mean that the provider does not support any profile of the Register Name Identifier Protocol.

- `RegisterNameIdentifierServiceURL` [Optional, 0-1] The URL used for user-agent-based Register Name Identifier Protocol profiles.

- `RegisterNameIdentifierServiceReturnURL` [Optional, 0-1] The provider's redirecting URL for use after HTTP name registration has taken place.

- `NameIdentifierMappingProtocolProfile` [Optional, 0-many] of type `anyURI`, which indicates the profile of the NameIdentifierMapping protocol supported by the Provider. This subject entity of the metadata instance should be a provider who administers identifiers for a subject across multiple namespaces.

- `NameIdentifierMappingEncryptionProfile` [optional, 0-many] of type `anyURI`, which indicates the encryption profiles supported by the provider as a recipient of an encrypted `NameIdentifier`.

- `Organization` [Optional, 0-1] The Organization (see Section 2.1.4.1) information about the provider.

- `ContactPerson` [Optional, 0-many] A Container expressing one or more contacts responsible for technical, administrative, billing, or other information concerning an identity service implementation expressed in the metadata (see Section 2.1.4.2)

- `AdditionalMetaLocation` [Optional, 0-many] The location of other relevant metadata about the provider which MAY contain the attribute `namespace`, indicating the namespace of the target document.

- `Extension` [Optional, 0-1] Provides for metadata extensions describing an *SP* or *IdP*

- `ds:Signature` [Optional, 0-1] An optional signature of the provider metadata (see Section 5.1)

Each of these elements is optional. The schema fragment for `providerDescriptorType`:

```
<xs:complexType name="providerDescriptorType">
   <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0"
         name="KeyDescriptor" type="keyDescriptorType"/>
      <xs:element minOccurs="0" name="SoapEndpoint" type="xs:anyURI"/>
      <xs:element minOccurs="0" name="SingleLogoutServiceURL" type="xs:anyURI"/>
      <xs:element minOccurs="0" name="SingleLogoutServiceReturnURL" type="xs:anyURI"/>
      <xs:element minOccurs="0" name="FederationTerminationServiceURL" type="xs:anyURI"/>
      <xs:element minOccurs="0" name="FederationTerminationServiceReturnURL" type="xs:anyURI"/>
      <xs:element maxOccurs="unbounded" minOccurs="0"
         name="FederationTerminationNotificationProtocolProfile" type="xs:anyURI"/>
      <xs:element maxOccurs="unbounded" minOccurs="0"
         name="SingleLogoutProtocolProfile" type="xs:anyURI"/>
      <xs:element maxOccurs="unbounded" minOccurs="0"
         name="RegisterNameIdentifierProtocolProfile" type="xs:anyURI"/>
      <xs:element minOccurs="0"
         name="RegisterNameIdentifierServiceURL" type="xs:anyURI"/>
      <xs:element minOccurs="0"
         name="RegisterNameIdentifierServiceReturnURL" type="xs:anyURI"/>
      <xs:element maxOccurs="unbounded" minOccurs="0"
         name="NameIdentifierMappingProtocolProfile" type="xs:anyURI"/>
      <xs:element minOccurs="0" maxOccurs="unbounded"
```

```
            name="NameIdentifierMappingEncryptionProfile" type="xs:anyURI"/>
         <xs:element minOccurs="0" name="Organization" type="organizationType"/>
         <xs:element maxOccurs="unbounded" minOccurs="0" name="ContactPerson" type="contactType"/>
         <xs:element maxOccurs="unbounded" minOccurs="0"
           name="AdditionalMetaLocation" type="additionalMetadataLocationType"/>
         <xs:element minOccurs="0" ref="Extension"/>
         <xs:element minOccurs="0" ref="ds:Signature"/>
      </xs:sequence>
      <xs:attribute name="protocolSupportEnumeration" type="xs:NMTOKENS" use="required"/>
      <xs:attribute name="id" type="xs:ID" use="optional"/>
      <xs:attribute name="validUntil" use="optional"/>
      <xs:attribute name="cacheDuration" use="optional"/>
   </xs:complexType>
```

2.1.5. Descriptors for Entities

2.1.5.1. SPDescriptor Element

`SPDescriptor` extends `providerDescriptorType` with the following elements:

- `AssertionConsumerServiceURL` [Required, 1-many] One or more URI(s) of the SP for receiving Authentication Assertions from an authenticating party. When an SP sends an *AuthNRequest* to the IdP, it may indicate the preferred `AssertionConsumerServiceURL` using the provided `id` (QNAME) attribute to direct the principal to for consumption of the *AuthNResponse*.
  IdP's should inspect the Service Providers metadata for the appropriate URL, or the default (indicated by the `isDefault` attribute) location, if no `id` is provided. Publishers MUST express only one default `AssertionConsumerServiceURL`. `AssertionConsumerServiceURL` requires the following attributes:

  - `id` [Required]. The fragment identifier of the `AssertionConsumerServiceURL` used as a reference in an AuthNRequest.

  - `isDefault` [Required]. A boolean indicator for the default `AssertionConsumerServiceURL` value to use when no identifier is provided in the request.

- `AuthnRequestsSigned` [Required, 1] boolean element indicating whether the Service Provider will always signed it's *AuthNRequest*s

the schema fragment for SPDescriptor:

```
<xs:complexType name="SPDescriptorType">
   <xs:complexContent>
      <xs:extension base="providerDescriptorType">
         <xs:sequence>
            <xs:element maxOccurs="unbounded" name="AssertionConsumerServiceURL">
               <xs:complexType>
                  <xs:simpleContent>
                     <xs:extension base="xs:anyURI">
                        <xs:attribute name="id" type="xs:ID" use="required"/>
                        <xs:attribute default="false" name="isDefault" type="xs:boolean"/>
                     </xs:extension>
                  </xs:simpleContent>
               </xs:complexType>
            </xs:element>
            <xs:element name="AuthnRequestsSigned" type="xs:boolean"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
```

```
   </xs:complexType>
```

2.1.5.2. IDPDescriptor Element

`IDPDescriptor` extends `providerDescriptorType` with the following elements:

- `SingleSignOnServiceURL` [Required, 1]. The identity provider's URL for accepting authentication requests for the Single Sign-On and Federation Protocol.
- `SingleSignOnProtocolProfile` [Required, 1-many]. The Single Sign-On Protocol profiles supported by the provider. Each element MUST contain a valid Single Sign-On Protocol profile identification URI.
- `AuthnServiceURL` [Optional, 0-1] of type `anyURI` describes the SOAP Endpoint supporting the ID-FF authentication by the identity provider as defined in [LibertyAuthn] and supports the relevant profile(s) cited in `SingleSignOnProtocolProfile`. IF the IDP supports SOAP-based IDFF authentication, indicated by the associated `SingleSignOnProtocolProfile`, and there is no `AuthnServiceURL` provided, then the IDP supports this profile at the URL identified by `SingleSignOnServiceURL`.

The schema fragment for `IDPDescriptor`:

```
   <xs:complexType name="IDPDescriptorType">
      <xs:complexContent>
         <xs:extension base="providerDescriptorType">
            <xs:sequence>
               <xs:element name="SingleSignOnServiceURL" type="xs:anyURI"/>
               <xs:element maxOccurs="unbounded" name="SingleSignOnProtocolProfile"
type="xs:anyURI"/>
               <xs:element name="AuthnServiceURL" type="xs:anyURI" minOccurs="0"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
```

2.1.5.3. EntityDescriptor Element

The element `EntityDescriptor` is used to contain one or more descriptor types for a given organization. Publishers MUST NOT convey metadata for other unaffiliated organizations within this node. Representations of multiple, unaffiliated providers within a single instance document MUST be done using the plural node form EntitiesDescriptor (Section 2.1.5.4) instead. Publishers MUST publish all relevant roles in this single document, or indirectly through `AdditionalMetaLocation`.

Entities describing a single `providerID`, but wish to publish different metadata for two implementations protocol support (e.g. IDFF 1.1 services vs. IDFF 1.2 services) may use the plurality of `IDPDescriptor` and `SPDescriptor` to convey this.

Note that it is possible for a single organization to be represented by more than one `providerID`, by indicating different providerID attributes for each entity descriptor, and publishing the document as `EntitiesDescriptor`.

`EntityDescriptor` may contain **either**: zero or more `IDPDescriptors` and zero or more `SPDescriptors`, **or** exactly one `AffiliationDescriptor` followed by any of: `ContactPerson`, `Organization`, `ds:Signature`, and `Extension`.

Attributes for `EntityDescriptor`:

- `providerID` [Required]: the providerID of the entity whose metadata is represented by all descendants of `EntityDescriptor`
- `id` [Optional] fragment identifier which is required if `ds:Signature` is present.
- `validUntil` [Optional] The expiration dateTime of the metadata.
- `cacheDuration` [Optional] The cache duration period for the metadata.

Elements contained in `EntityDescriptor`:

- `IDPDescriptor` Metadata describing an entity acting as an Identity Provider.
- `SPDescriptor` Metadata describing an entity acting as a Service Provider.
- `AffiliationDescriptor` Metadata describing a set of entities identified by their respective `providerID`s collectively referred to as an affiliation Section 2.1.5.5
- `ContactPerson` [Optional, 0-1] Contact information for the overall entity (see Section 2.1.4.2.
- `Organization` [Optional, 0-1]Organizational information about the entity (see Section 2.1.4.1).
- `Extension` [Optional, 0-1] provides extension point for additional entity metadata
- `ds:Signature` [Optional, 0-1] An XML Signature on the entire entity metadata instance.

The schema fragment for `entityDescriptorType`:

```
<xs:complexType name="entityDescriptorType">
   <xs:sequence>
      <xs:choice>
         <xs:group ref="providerGroup"/>
         <xs:element name="AffiliationDescriptor" type="affiliationDescriptorType"/>
      </xs:choice>
      <xs:element minOccurs="0" name="ContactPerson" type="contactType"/>
      <xs:element minOccurs="0" name="Organization" type="organizationType"/>
      <xs:element minOccurs="0" ref="Extension"/>
      <xs:element minOccurs="0" ref="ds:Signature"/>
   </xs:sequence>
   <xs:attribute name="providerID" use="required"/>
   <xs:attribute name="id" type="xs:ID" use="optional"/>
   <xs:attribute name="validUntil" use="optional"/>
   <xs:attribute name="cacheDuration" use="optional"/>
</xs:complexType>

<xs:element name="EntityDescriptor" type="entityDescriptorType"/>

<xs:group name="providerGroup">
   <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0"
         name="IDPDescriptor" type="IDPDescriptorType"/>
      <xs:element maxOccurs="unbounded" minOccurs="0"
         name="SPDescriptor" type="SPDescriptorType"/>
   </xs:sequence>
</xs:group>
```

2.1.5.4. EntitiesDescriptor

The element `EntitiesDescriptor` describes more than one organization in a single instance document. It consists of 2 or more `EntityDescriptors`.

The schema fragment for `EntitiesDescriptor` element:

```
<xs:element name="EntitiesDescriptor" type="entitiesDescriptorType"/>
<xs:complexType name="entitiesDescriptorType">
   <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="2" ref="EntityDescriptor"/>
   </xs:sequence>
</xs:complexType>
```

2.1.5.5. AffiliationDescriptor

The `AffiliationDescriptor` element describes a group of entities, identified collectively by `providerID` (located within `EntityDescriptor`), as an enumeration of `providerID`'s. The uniqueness constraints for `providerID` also apply for `providerID` in this context, such that it MUST be unique across all Liberty entities with which the affiliation expects to interact, including other affiliations and providers therefore, it MUST NOT be the `providerID` of any of the members of the affiliation, and SHOULD be unique across the set of `providerID`'s with which the affiliation expects to interact. It is the responsibility of the entity represented by `affiliationOwnerID` to administer this identifier, and thus, its members and uniqueness.

`AffiliationDescriptor` element contains the following attributes:

- `affiliationOwnerID` [Required] the `providerID` of the owner or parent operator of the affiliation, from which, additional metadata may be derived. This DOES NOT indicate affiliation membership of entity described as `affiliationOwnerID`. Thus if a member is both the owner of and a member of the affiliation, they must indicate both in the instance (e.g. the entities `providerID` appears in both `affiliationOwnerID` AND `AffiliateMember`.
- `validUntil` [Optional] The expiration dateTime of the metadata.
- `cacheDuration` [Optional] The cache duration period for the metadata.
- `id` [Optional].

and the following elements:

- `AffiliateMember` [Required, 1-many] One or more providers who are members of the affiliation. The value MUST be a `providerID` who's metadata MUST be obtained via methods described in Section 3
- `Extension` [Optional, 0-1] provides an extension point to convey additional metadata concerning the affiliation
- `KeyDescriptor` [Optional, 0-many] Zero or more public key material reference that is the property of the affiliation. This keying material SHOULD be separate from the keying material of the `providerID` who may be referenced as the `affiliateOwnerID` and MAY be used for encryption or signing, as indicated by its corresponding `use` attribute.
- `ds:Signature` [Optional, 0-1] An XML Signature of the metadata node `AffiliationDescriptor`.

The schema fragment for the `AffiliationDescriptor` element:

```
   <xs:complexType name="affiliationDescriptorType">
      <xs:sequence>
         <xs:element maxOccurs="unbounded" name="AffiliateMember" type="entityIDType"/>
         <xs:element minOccurs="0" ref="Extension"/>
         <xs:element maxOccurs="unbounded" minOccurs="0"
            name="KeyDescriptor" type="keyDescriptorType"/>
         <xs:element minOccurs="0" ref="ds:Signature"/>
      </xs:sequence>
      <xs:attribute name="affiliationOwnerID" type="entityIDType" use="required"/>
      <xs:attribute name="validUntil" use="optional"/>
      <xs:attribute name="cacheDuration" use="optional"/>
      <xs:attribute name="id" type="xs:ID" use="optional"/>
   </xs:complexType>
```

2.1.6. WSDL Usage

A WSDL [WSDLv1.1] document MAY be used to describe the web services available at the location `SoapEndpoint` in addition to the metadata itself. Following is the abstract WSDL describing the ID-FF services:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="IDFF" targetNamespace="urn:liberty:md:IDFF:wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:iff="urn:liberty:iff:2003-08"
   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="urn:liberty:md:IDFF:wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <import location="oasis-sstc-saml-schema-assertion-1.1.xsd"
     namespace="urn:oasis:names:tc:SAML:1.0:assertion"/>
  <import location="liberty-idff-protocols-schema-1.2-errata-v3.0.xsd"
     namespace="urn:liberty:iff:2003-08"/>
  <import location="oasis-sstc-saml-schema-protocol-1.1.xsd"
     namespace="urn:oasis:names:tc:SAML:1.0:protocol"/>
  <types/>
  <!--annotation>
    <documentation>WSDL from Metadata description and discovery protocols</documentation>
    <documentation>The source code in this WSDL file was excerpted verbatim from:

Liberty Metadata Description and Discovery Specification
Version 1.1
14th December 2004

Copyright (c) 2004-2005 Liberty Alliance participants, see
https://www.projectliberty.org/specs/idff_copyrights.html

</documentation>
  </annotation-->
  <message name="authenticationResponse">
    <part name="body" element="iff:AuthnResponseEnvelope"/>
  </message>
  <message name="NameIdentifierMappingResponse">
    <part name="body" element="saml:Assertion"/>
  </message>
  <message name="artifactResponse">
    <part name="body" element="samlp:Response"/>
  </message>
  <message name="LogoutRequest">
    <part name="body" element="iff:LogoutRequest"/>
  </message>
  <message name="registerNameIdentifierRequest">
    <part name="body" element="iff:RegisterNameIdentifierRequest"/>
```

```
  </message>
  <message name="LogoutResponse">
    <part name="body" element="iff:LogoutResponse"/>
  </message>
  <message name="NameIdentifierMappingRequest">
    <part name="body" element="samlp:Request"/>
  </message>
  <message name="FederationTermination">
    <part name="body" element="iff:FederationTerminationNotification"/>
  </message>
  <message name="authenticationRequest">
    <part name="body" element="iff:AuthnRequest"/>
  </message>
  <message name="artifactRequest">
    <part name="body" element="samlp:Request"/>
  </message>
  <message name="registerNameIdentifierResponse">
    <part name="body" element="iff:RegisterNameIdentifierResponse" />
  </message>
  <portType name="IDPPort">
    <operation name="registerNameIdentifier">
      <input message="tns:registerNameIdentifierRequest"/>
      <output message="tns:registerNameIdentifierResponse"/>
    </operation>
    <operation name="FederationTermination">
      <input message="tns:FederationTermination"/>
    </operation>
    <operation name="Logout">
      <input message="tns:LogoutRequest"/>
      <output message="tns:LogoutResponse"/>
    </operation>
    <operation name="NameIdentifierMapping">
      <input message="tns:NameIdentifierMappingRequest"/>
      <output message="tns:NameIdentifierMappingResponse"/>
    </operation>
    <operation name="authentication">
      <input message="tns:authenticationRequest"/>
      <output message="tns:authenticationResponse"/>
    </operation>
    <operation name="artifact">
      <input message="tns:artifactRequest"/>
      <output message="tns:artifactResponse"/>
    </operation>
  </portType>
  <service name="IDFF">
    <port binding="tns:IDPBinding" name="IDFFPort">
      <soap:address location="http://localhost:8000/ccx/IDFF"/>
    </port>
  </service>
</definitions>
```

3. Publishing the Metadata

Two mechanisms are provided for entities to publish metadata document locations: via the DNS and via a "well-known-location" by directly dereferencing the entities' `providerID`s.

When retrieval requires network transport of the document, in both cases above the transport SHOULD be protected with *TLS/SSL* [RFC2246] as amended by [RFC3546]. This will ensure the integrity of the metadata document, as among other information within the document, trust establishment may be based in part on information provided within the metadata. Relying parties of this metadata are RECOMMENDED to authenticate the server via *TLS/SSL* validation procedures.

Trust establishment of the Metadata will be based on one or more of the following: DNS signatures (RECOMMENDED); TLS server authentication (RECOMMENDED); and Metadata `ds:Signature` (STRONGLY RECOMMENDED) evaluations. Publishers MAY implement additional trust mechanisms, in conjunction with the required suggested server authentication. Additional trust metadata content, if supplied, MUST be placed in the extension points provided.

3.1. Instance Publication Forms

If separate documents are used, references to each MUST be made, either through one or more additional PID2MD NAPTR record(s), or by using the `AdditionalMetaLocation` element within a document which has an associated NAPTR RR, or which is situated at the "well-known location" (see Section 3.3).

3.2. Using the DNS to Publish Metadata Location(s)

In order to ensure that all providers have accessible metadata locations, entities are STRONGLY RECOMMENDED to publish their metadata document locations in a zone of their corresponding DNS [RFC1034]. As *providerID*s are flexible identifiers, publication and resolution is determined by an entity's URI scheme and fully qualified name part of the identifier. URI locations for metadata will subsequently be derived through queries of the NAPTR Resource Record (RR) as defined in [RFC2915] and [RFC3403].

It is RECOMMENDED that entities publish their resource records in signed zone files using [RFC2535] such that relying parties may establish the validity of the published location and authority of the zone, and integrity of the DNS response. If DNS zone signatures are present, relying parties MUST properly validate the signature.

3.2.1. Publication of Metadata Locations

This specification makes use of the resource record described in [RFC2915] and [RFC3403]. Familiarity with these documents is encouraged.

Dynamic Delegation Discovery System (DDDS) [RFC3401] is a general purpose system for the retrieval of information based on an application-specific input string and the application of well known rules to transform that string until a terminal condition is reached requiring a look-up into an application-specific defined database or execution of a URL based on the rules defined by the application. DDDS defines a specific type of DNS Resource Record, NAPTR records, for the storage of information in the DNS necessary to apply DDDS rules.

Entities MAY publish separate URL's when the metadata documents need to be distributed, or when different metadata documents are required due to multiple Authentication Domain memberships which require separate keying material, or when service interfaces require separate metadata declarations. This may be accomplished through the use of the optional `AdditionalMetaLocation` attribute in the core or other subordinate metadata document, or through the `regexp` facility and multiple service definition fields in the NAPTR resource record itself.

If `providerID` is a URN, resolution of the `MetadataLocation` proceeds as specified in [RFC3404]. Otherwise, the resolution of the metadata location proceeds as specified in this specification.

Following are the application-specific descriptions for the DDDS application for the Liberty Metadata resolution protocols.

3.2.1.1. Application Unique String

Liberty metadata resolution shall begin with the application unique string of `providerID`.

3.2.1.2. First Well Known Rule

The "first well-known-rule" for processing Liberty Alliance Metadata resolution is to parse the `providerID` URI and extract the fully qualified domain name (subexpression 3) as described in section Section 4.1.1.

3.2.1.3. The Order Field

The order field indicates the order for processing each NAPTR resource record returned. Publishers MAY provide multiple NAPTR resource record's which MUST be processed by the resolver application in the order indicated by this field.

3.2.1.4. The Preference Field

For terminal NAPTR resource records, the publisher expresses the preferred order of use to the resolving application. The resolving application MAY ignore this order, in cases where the service field value does not meet the resolver's requirements (e.g.: the resource record returns a protocol the application does not support).

3.2.1.5. The Flag Field

Liberty Metadata resolution twice makes use of the "U" flag, which is terminal, and the null value (implying additional resource record's are to be processed). The "U" flag indicates that the output of the rule is a URI.

3.2.1.6. The Service Field

The Liberty specific service fields shall include:

```
servicefield = 1("PID2U" / "NID2U") "+" proto [*(":" class) *(":" servicetype)]
proto = 1("https" / "uddi ")
class = 1[ "entity" / "entitygroup" )
servicetype = 1(si / "sp" / "idp" / "authn" / alphanum )
si = "si" [":" alphanum] [":endpoint"]
alphanum = 1*32(ALPHA / DIGIT)
```

where:

- PID2U resolves a providerID identifier to metadata URL
- NID2U resolves a nameIdentifier (principal) metadata URL
- `proto` describes the retrieval protocol (https or uddi). In the case of UDDI, the resulting URI will be an http(s) URI referencing a `WSDL` document.
- `class` identifies which indicates whether the referenced metadata document describes a single provider, or multiple. In the latter case, the referenced document MUST contain the entity defined by `providerID` as a member of a group of entities within the document itself.

- `servicetype` allows a publishers to publish service provider, identity provider and service instance metadata locations as separate documents. Resolvers who encounter multiple servicetype declarations will dereference the appropriate URI, depending on which service type required for an operation (e.g.: a provider operating both and IdP and an SP service, may publish SP and IdP metadata at different locations).

- the `si` component (with optional endpoint component) allows the publisher to either directly publish the metadata for a service instance, or by articulating the soap endpoint (using `endpoint`

For example:

- PID2U+https:entity - represents the complete entity metadata document via the https protocol

- PID2U+https:entity:si:pip - returns the PIP metadata URL for the entity described by providerID via the https protocol profile

- PID2U+uddi:entity:si:foo - returns the WSDL document location which describes a service instance "foo"

- PID2U+https:entitygroup:idp - returns the metadata for a group of entities, of which providerID is a member. the referenced document describes (one or more) IdPs in the group

- NID2U+https:idp - returns an IdP providerIDs, who can provider authentication services for a principal

- NID2U+https:authn - returns a URL to attempt to authenticate the principal against

3.2.1.7. The Regex and Replacement Fields

The expected output after processing the application-unique string through the regex MUST be a valid https URL or UDDI node (http references wsdl document) address.

3.2.2. NAPTR Examples

3.2.2.1. Provider Metadata NAPTR Examples

Entities publish metadata URLs in the following manner:

```
$ORIGIN provider.biz

;; order pref f service regexp or replacement

 IN NAPTR 100 10 "U" PID2U+https:entity
    "!^.*$!https://host.provider.biz/some/directory/trust.xml!" ""
 IN NAPTR 110 10 "U" PID2U+https:entity:trust
    "!^.*!https://foo.provider.biz:1443/mdtrust.xml!" ""
 IN NAPTR 125 10 "U" PID2U+https:"
 IN NAPTR 110 10 "U" PID2U+uddi:entity "!^.*$ !https://this.uddi.node.provider.biz/libmd.wsdl" ""
```

3.2.2.2. Name Identifier Examples

Principals employer example.int operates an IdP which may be used by a office supply company to authenticate authorized buyers. The supplier takes users email address buyer@example.int as input to the resolution process, and parses the email address to extract the FQDN (example.int). The employer publishes the following NAPTR in example.int:

```
$ORIGIN

example.int.

  IN NAPTR 100 10 "U" NID2U+https:authn
     "!^([^@]+)@ (.*)$!https://serv.example.int:8000/cgi-bin/getmd?\1!" ""
  IN NAP TR 100 10 "U" NID2U+https:idp
     "!^([ ^@]+)@(.*)$!https://auth.example.int/app/auth?\1" ""
```

3.3. Publication via Well-Known Location

Entities MAY publish their metadata documents at a well known location. The core metadata document location in this profile simply involves directly dereferencing the providerID and obtaining the document directly (or through schema-specific means of indirection)

For well known location documents, the XML document MUST describe the metadata for the **providerID** entity only. If other entities need to be described, the **AdditionalMetaLocation** MUST be used. Thus the **entitiesDescriptor** MUST NOT be used in documents published at a well know location, since entities as a group, are not defined by such an identifier.

4. Metadata Resolution and Retrieval

Metadata publication is provided in two fashions: via a "well-known-location" and via queries on the DNS. Both mechanisms depend upon processing of the `providerID` element (see [Section 3]), which is the primary identifier for Liberty-enabled entities. Consumers are STRONGLY RECOMMENDED to attempt DNS-based resolution prior to performing a direct dereferencing of a providerID.

The providerID is defined as a restricted form of `anyURI` Section 2.1.2; therefore, it shall be parsed as in Section 4.1.1 for these resolution profiles.

4.1. Resolving Locations and Retrieving Metadata

The summarized steps for retrieving metadata from a given `providerID` is as follows:

- If the providerID is a URN, proceed with the resolution steps as defined in [RFC3404]
- parse the `providerID` to obtain the *FQDN*
- query the DNS for NAPTR resource records of the `domain name` iteratively until a terminal resource record is returned
  (optionally, or if DNS-based resolution fails) attempt locating the metadata document(s) via the *well known location* profile by directly dereferencing the providerID (end if a document was located, validated and fulfills the metadata requirements for the present operations).
- identify which resource record to use based on the service fields, then order fields, then preference fields of the result set.
- obtain the document(s) at the provided location(s) as required by the application

4.1.1. Parsing the ProviderID

To initiate the resolution of the location of the target metadata elements, it will be necessary in some cases to decompose the ProviderID (expressed as a URI) into one or more atomic elements.

The following regular expression should be used when initiating the decomposition process:

```
  ^([^:/?#]+:)?/*([^:/?#]*@)?(([^/?:#]*\.)*(([^/?#:\.]+)\.([^/?#:\.]+)))(:\d+)?(
[^?#]*)(\?[^#]*)?(#.*)?$
    1            2           34            56            7            8      9       10      ↩
 11
```

Subexpression 3 MUST result in a Fully Qualified Domain Name (FQDN), which will be the basis for retrieving metadata locations from this zone.

4.1.2. Obtaining Metadata via the DNS

Upon completion of the parsing of the `providerID`, the application then performs a DNS query for the resulting domain (subexpression 5) for NAPTR resource records; it should expect 1 or more responses. Applications MAY exclude from the result set any service definitions which do not concern the present request operations. Should the DNS not produce a valid response, the consumer MUST ALWAYS attempt direct dereferencing of the providerID.

Resolving applications MUST subsequently order the result set according to the order field, and MAY order the result set based on the preference set. Resolvers are NOT REQUIRED to follow the ordering of the preferences field.

The resulting NAPTR resource record(s) are operated on iteratively (based on the order flag) until a terminal NAPTR resource record is reached.

The result will be a well formed, fully qualified URL, which will then be used to retrieve the metadata document.

4.1.2.1. Post Processing Operations

When service specific metadata is sought, resolvers MAY filter the NAPTR result set based on more specific resource records with service identifiers which match the service(s) sought.

4.1.3. Obtaining Metadata via the "Well-Known Location Method"

Consumers of published metadata MAY attempt retrieval via the well-known-location method by directly dereferencing the providerID. Other forms of well-known location MAY be agreed upon by a group of Liberty entities, however, it is STRONGLY SUGGESTED that publication in the DNS be employed as well, to allow for interactions with other Liberty implementations. The resulting XML document MUST describe the metadata for the `providerID` entity only. If other entities need to be described, the `AdditionalMetaLocation` MUST be used. There may be only one location, although this document MAY point to other document locations using the `AdditionalMetaLocation` element.

5. Post Processing of the Metadata Document

5.1. Processing of ds:Signature and General Trust Processing

Metadata processing provides several mechanisms for trust negotiation for both the metadata itself and for the trust ascribed to the entity described by such metadata:

- Trust derived from the signature of the zone from which the metadata location URI was resolved, ensuring accuracy of the metadata document location(s)
- Trust derived from signature processing of the metadata document itself, ensuring the integrity of the XML document
- Trust derived from the SSL/TLS negotiation of the metadata delivery URI, ensuring the identity of the publisher of the metadata.

Post processing of the metadata document MUST include the signature processing at the XML-document level and MAY include one of the other two processes. Specifically, the relying party MAY choose to trust any of the cited authorities in the resolution and parsing process. Publishers of metadata MUST employ a document-integrity mechanism and MAY employ any of the other two processing profiles to establish trust of the subject of the metadata document, governed by implementation policies.

5.1.1. Processing Signed DNS Zones

Verification of zone signature SHOULD be processed, if present, as described in [RFC2535]

5.1.2. Processing Signed Documents and Fragments

Published metadata documents SHOULD be signed, as described in [XMLDsig], either by a certificate issued to the subject of the document, or by another trusted party. Publishers MAY consider signatures of other parties as a means of trust conveyance.

Consumers MUST validate signatures, when present, on the metadata document on initial retrieval as described by [XMLDsig].

5.1.3. Processing Server Authentication in Metadata Retrieval via TLS/SSL

It is STRONGLY RECOMMENDED that publishers implement TLS URL's; therefore, consumers SHOULD consider the trust inherited from the issuer of the TLS/SSL certificate. Publication URLs may not always be located in the domain of the provider of the subject of the metadata document; therefore, consumers SHOULD NOT expect certificates whose subject is the provider, as it may be hosted at another trusted party.

As the basis of this trust may not be available against a cached document, other mechanisms SHOULD be used under such circumstances.

5.2. Metadata Location and Document Caching

Location caching based on DNS profiles MUST NOT exceed the TTL of the DNS zone from which the location was derived. Resolvers MUST obtain a fresh copy of the Metadata location upon reaching the expiration of the TTL of the zone.

A publishers of Metadata documents should carefully consider the TTL of the zone when making updates to its metadata document location. Should such a location change occur, a publisher MUST either keep the document at both the old and new location until all conforming resolvers are certain to have the updated location (e.g.: time of zone change + TTL), or provide an `HTTP Redirect` [RFC2616] to the new location.

Document caching MUST NOT exceed the `validUntil` attribute of the subject element(s) and the `cacheDuration` attribute. If fragments have parents which contain caching policies, the parent fragment ALWAYS takes precedence.

To properly process the `cacheDuration` attributes on fragments and documents consumers MUST retain the *dateTime* when the document was retrieved.

When a document or fragment has expired the consumer MUST retrieve a fresh copy, which may require a refresh of the document location(s). Consumers SHOULD process document cache processing according to [RFC2616] section 13, and MAY request the Last-Modified dateTime from the HTTPS server. Publishers SHOULD ensure acceptable cache processing as described in [RFC2616] (Section 10.3.5 304 Not Modified).

5.3. Handling of HTTPS Redirects

Publishers MAY issue an HTTP Redirect (301 Moved Permanently, or 307 Temporary Redirect) [RFC2616], and user agents MUST follow the specified URL in the Redirect response. Redirects SHOULD be to a TLS/SSL protected resource, and SHOULD be of the same protocol as the initial request.

6. Security Considerations

6.1. Trust Establishment

Cryptographic signatures are used to establish identity and tamper evidence in several locations within the metadata specification. While valid signatures convey some level of trust in the resulting document, extreme care should be taken as to the validity of the URIs described within the document itself. Relying parties should carefully inspect agreements and statements made by the signing authorities of the subject certificates or keys.

7. Metadata XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:liberty:metadata:2003-08"
  xmlns="urn:liberty:metadata:2003-08"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="1.0">
  <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd"/>
  <xs:import namespace="urn:oasis:names:tc:SAML:1.0:assertion"
    schemaLocation="oasis-sstc-saml-schema-assertion-1.1.xsd"/>
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xs:include schemaLocation="liberty-utility-v1.1.xsd"/>
  <xs:annotation>
    <xs:documentation>

      XML Schema fom Metadata description and discovery protocols

      The source code in this XSD file was excerpted verbatim from:

      Liberty Metadata Description and Discovery Specification
      Version 1.1
      14 December 2004

      Copyright (c) 2004-2005 Liberty Alliance participants, see
      https://www.projectliberty.org/specs/idwsf_1_1_copyrights.php

    </xs:documentation>
  </xs:annotation>
  <xs:simpleType name="entityIDType">
    <xs:restriction base="xs:anyURI">
      <xs:maxLength value="1024" id="maxlengthid"/>
    </xs:restriction>
  </xs:simpleType>
  <!--
  <xs:attribute name="libertyPrincipalIdentifier" type="entityIDType"/>
  <xs:attribute name="providerID" type="entityIDType"/>
  <xs:attribute name="validUntil" type="xs:dateTime"/>
  <xs:attribute name="cacheDuration" type="xs:duration"/>
  -->
  <xs:complexType name="additionalMetadataLocationType">
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
        <xs:attribute name="namespace" type="xs:anyURI"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="organizationType">
    <xs:sequence>
      <xs:element name="OrganizationName" type="organizationNameType" maxOccurs="unbounded"/>
      <xs:element name="OrganizationDisplayName" type="organizationDisplayNameType"
maxOccurs="unbounded"/>
      <xs:element name="OrganizationURL" type="localizedURIType" maxOccurs="unbounded"/>
      <xs:element ref="Extension" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="organizationNameType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute ref="xml:lang"/>
      </xs:extension>
    </xs:simpleContent>
```

```
  </xs:complexType>
  <xs:complexType name="organizationDisplayNameType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute ref="xml:lang" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="localizedURIType">
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
        <xs:attribute ref="xml:lang" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="contactType">
    <xs:sequence>
      <xs:element name="Company" type="xs:string" minOccurs="0"/>
      <xs:element name="GivenName" type="xs:string" minOccurs="0"/>
      <xs:element name="SurName" type="xs:string" minOccurs="0"/>
      <xs:element name="EmailAddress" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="TelephoneNumber" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="Extension" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="libertyPrincipalIdentifier" type="entityIDType" use="optional"/>
    <xs:attribute name="contactType" type="attr.contactType" use="required"/>
  </xs:complexType>
  <xs:simpleType name="attr.contactType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="technical"/>
      <xs:enumeration value="administrative"/>
      <xs:enumeration value="billing"/>
      <xs:enumeration value="other"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="keyTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="encryption"/>
      <xs:enumeration value="signing"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="providerDescriptorType">
    <xs:sequence>
      <xs:element name="KeyDescriptor" type="keyDescriptorType"
        minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="SoapEndpoint" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="SingleLogoutServiceURL" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="SingleLogoutServiceReturnURL"
        type="xs:anyURI" minOccurs="0"/>
      <xs:element name="FederationTerminationServiceURL"
        type="xs:anyURI" minOccurs="0"/>
      <xs:element name="FederationTerminationServiceReturnURL"
        type="xs:anyURI" minOccurs="0"/>
      <xs:element name="FederationTerminationNotificationProtocolProfile"
        type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="SingleLogoutProtocolProfile"
        type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="RegisterNameIdentifierProtocolProfile"
        type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="RegisterNameIdentifierServiceURL"
        type="xs:anyURI" minOccurs="0"/>
      <xs:element name="RegisterNameIdentifierServiceReturnURL"
        type="xs:anyURI" minOccurs="0"/>
      <xs:element name="NameIdentifierMappingProtocolProfile"
        type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="NameIdentifierMappingEncryptionProfile"
        type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
```

```
    <xs:element name="Organization" type="organizationType" minOccurs="0"/>
    <xs:element name="ContactPerson" type="contactType"
     minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="AdditionalMetaLocation"
     type="additionalMetadataLocationType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Extension" minOccurs="0"/>
    <xs:element ref="ds:Signature" minOccurs="0"/>
  </xs:sequence>
  <!--xs:attribute ref="providerID" use="required"/-->
  <xs:attribute name="protocolSupportEnumeration" type="anyURIListType" use="required"/>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
  <xs:attribute name="validUntil" type="xs:dateTime"/>
  <xs:attribute name="cacheDuration" type="xs:duration"/>
</xs:complexType>
<xs:simpleType name="anyURIListType">
  <xs:list itemType="xs:anyURI"/>
</xs:simpleType>
<!--added-->
<xs:element name="KeyDescriptor" type="keyDescriptorType"/>
<xs:complexType name="keyDescriptorType">
  <xs:sequence>
    <xs:element name="EncryptionMethod" type="xs:anyURI" minOccurs="0"/>
    <xs:element name="KeySize" type="xs:integer" minOccurs="0"/>
    <xs:element ref="ds:KeyInfo" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="use" type="keyTypes" use="optional"/>
</xs:complexType>
<!-- -->
<xs:element name="EntityDescriptor" type="entityDescriptorType"/>
<xs:group name="providerGroup">
  <xs:sequence>
    <xs:element name="IDPDescriptor" type="IDPDescriptorType"
     minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="SPDescriptor" type="SPDescriptorType"
     minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:group>
<xs:complexType name="entityDescriptorType">
  <xs:sequence>
    <xs:choice>
      <xs:group ref="providerGroup"/>
      <xs:element name="AffiliationDescriptor" type="affiliationDescriptorType"/>
    </xs:choice>
    <xs:element name="ContactPerson" type="contactType" minOccurs="0"/>
    <xs:element name="Organization" type="organizationType" minOccurs="0"/>
    <xs:element ref="Extension" minOccurs="0"/>
    <xs:element ref="ds:Signature" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="providerID" type="entityIDType" use="required"/>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
  <xs:attribute name="validUntil" type="xs:dateTime"/>
  <xs:attribute name="cacheDuration" type="xs:duration"/>
</xs:complexType>
<xs:complexType name="SPDescriptorType">
  <xs:complexContent>
    <xs:extension base="providerDescriptorType">
      <xs:sequence>
        <xs:element name="AssertionConsumerServiceURL" maxOccurs="unbounded">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:anyURI">
                <xs:attribute name="id" type="xs:ID" use="required"/>
                <xs:attribute name="isDefault" type="xs:boolean" default="false"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
```

```
        </xs:element>
        <xs:element name="AuthnRequestsSigned" type="xs:boolean"/>
       </xs:sequence>
     </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="IDPDescriptorType">
    <xs:complexContent>
     <xs:extension base="providerDescriptorType">
       <xs:sequence>
        <xs:element name="SingleSignOnServiceURL" type="xs:anyURI"/>
        <xs:element name="SingleSignOnProtocolProfile" type="xs:anyURI" maxOccurs="unbounded"/>
        <xs:element name="AuthnServiceURL" type="xs:anyURI" minOccurs="0"/>
       </xs:sequence>
     </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="EntitiesDescriptor" type="entitiesDescriptorType"/>
  <xs:complexType name="entitiesDescriptorType">
    <xs:sequence>
     <xs:element ref="EntityDescriptor" minOccurs="2" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="affiliationDescriptorType">
    <xs:sequence>
     <xs:element name="AffiliateMember" type="entityIDType" maxOccurs="unbounded"/>
     <xs:element ref="Extension" minOccurs="0"/>
     <xs:element name="KeyDescriptor" type="keyDescriptorType" minOccurs="0"
maxOccurs="unbounded"/>
     <xs:element ref="ds:Signature" minOccurs="0"/>
    </xs:sequence>
    <!-- <xs:attribute name="affiliationID" type="entityIDType" use="required"/> -->
    <xs:attribute name="affiliationOwnerID" type="entityIDType" use="required"/>
    <xs:attribute name="validUntil" type="xs:dateTime"/>
    <xs:attribute name="cacheDuration" type="xs:duration"/>
    <xs:attribute name="id" type="xs:ID" use="optional"/>
  </xs:complexType>
</xs:schema>
```

References

Normative

[LibertyAuthn] Hodges, Jeff, Aarts, Robert, eds. " Liberty ID-WSF Authentication Service Specification ," Version 1.1, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs/ [http://www.projectliberty.org/specs/]*

[LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification," Version 1.2-errata-v3.0, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*

[LibertyProtSchema1.1] Beatty, John, Kemp, John, eds. "Liberty Protocols and Schema Specification," Version 1.1, Liberty Alliance Project (January 2003). *http://www.projectliberty/specs [January 2003]*.

[LibertyBindProf] Cantor, Scott, Kemp, John, Champagne, Darryl, eds. "Liberty ID-FF Bindings and Profiles Specification," Version 1.2-errata-v2.0, Liberty Alliance Project (12 September 2004). *http://www.projectliberty.org/specs*

[RFC1034] Mockapetris, P., eds. (November 1987). "DOMAIN NAMES - CONCEPTS AND FACILITIES," RFC 1510, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc1034.txt [November 1987]*.

[RFC2119] Bradner, S., eds. "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet Engineering Task Force (March 1997). *http://www.ietf.org/rfc/rfc2119.txt [March 1997]*.

[RFC2246] Dierks, T., Allen, C., eds. (January 1999). "The TLS Protocol," Version 1.0 RFC 2246, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2246.txt [January 1999]*.

[RFC2396] Berners-Lee, T., Fielding, R., Masinter, L., eds. (August 1998). "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2396.txt [August 1998]*.

[RFC2535] Eastlake, D., eds. (March 1999). "Domain Name System Security Extensions," RFC 2535, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2535.txt [March 1999]*.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., eds. (June 1999). "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2616.txt [June 1999]*.

[RFC2732] Hinden, R., Carpenter, B., Masinter, L., eds. (December 1999). "Format for Literal IPv6 Addresses in URL's," RFC 2732, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2732.txt [December 1999]*.

[RFC2915] Mealling, M., Daniel, R., eds. (September 2000). "The Naming Authority Pointer (NAPTR) DNS Resource Record," RFC 2915, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc2915.txt [September 2000]*.

[RFC3401] Mealling, M., eds. (October 2002). "Dynamic Delegation Discovery System (DDDS) - Part One: The Comprehensive DDDS," RFC 3401, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc3401.txt [October 2002]*.

[RFC3403] Mealling, M., eds. (October 2002). "Dynamic Delegation Discovery System (DDDS) - Part Three: The Domain Name System (DNS) Database," RFC 3403, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc3403.txt [October 2002]*.

[RFC3404] Mealling, M., eds. (October 2002). "Dynamic Delegation Discovery System (DDDS) - Part Four: The Uniform Resource Identifiers (URI) Resolution Application," RFC 3404, Internet Engineering Task Force *http://www.ietf.org/rfc/rfc3404.txt [October 2002]*.

[RFC3546] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., Wright, T., eds. (June 2003). "Transport Layer Security (TLS) Extensions," RFC 3546, The Internet Engineering Task Force *http://www.ietf.org/rfc/rfc3546.txt [June 2003]*.

[SAMLCore11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (27 May 2003). "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1," OASIS Committee Specification, version 1.1, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/documents.php?wg_abbrev=security*

[Schema1] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (May 2002). "XML Schema Part 1: Structures," Recommendation, World Wide Web Consortium *http://www.w3.org/TR/xmlschema-1/*

[Schema2] Biron, Paul V., Malhotra, Ashok, eds. (May 2002). "XML Schema Part 2: Datatypes," Recommendation, World Wide Web Consortium *http://www.w3.org/TR/xmlschema-2/*

[WSDLv1.1] "Web Services Description Language (WSDL) 1.1," Christensen, Erik, Curbera, Francisco, Meredith, Greg, Weerawarana, Sanjiva, eds. World Wide Web Consortium W3C Note (15 March 2001). *http://www.w3.org/TR/2001/NOTE-wsdl-20010315*

[XML] Bray, Tim, Paoli, Jean, Sperberg-McQueen, C.M., Maler, Eve, eds. (Oct 2000). "Extensible Markup Language (XML) 1.0 (Second Edition)," Recommendation, World Wide Web Consortium *http://www.w3.org/TR/2000/REC-xml-20001006*

[XMLDsig] Eastlake, Donald, Reagle, Joseph, Solo, David, eds. (12 Feb 2002). "XML-Signature Syntax and Processing," Recommendation, World Wide Web Consortium *http://www.w3.org/TR/xmldsig-core*

[xmlenc-core] Eastlake, Donald, Reagle, Joseph, eds. (December 2002). "XML Encryption Syntax and Processing," W3C Recommendation, World Wide Web Consortium *http://www.w3.org/TR/xmlenc-core/*

Informative

[LibertyIDFFOverview] Wason, Thomas, eds. "Liberty ID-FF Architecture Overview," Version 1.2-errata-v1.0, Liberty Alliance Project (12 September 2004). *http://www.projectliberty.org/specs*