Liberty ID-WSF Discovery Service Specification

Version: 1.2

Editors:
Jonathan Sergent, Sun Microsystems, Inc.

Contributors:
Robert Aarts, Nokia Corporation
John Beatty, Sun Microsystems, Inc.
Conor Cahill, AOL Time Warner, Inc.
Gary Ellison, Sun Microsystems, Inc.
Jukka Kainulainen, Nokia Corporation
John Kemp, IEEE-ISTO
Paul Madsen, Entrust, Inc.
Greg Whitehead, Trustgenix, Inc.
Emily Xu, Sun Microsystems, Inc.

Abstract:

Specification from the Liberty Alliance Project Identity Web Services Framework for describing and discovering identity services.

Filename: liberty-idwsf-disco-svc-v1.2.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

This specification defines a framework for describing and discovering identity services. A conceptual model with terminology is first provided to set the context for the rest of the specification.

1.1. Conceptual Model and Terminology

An *identity service* is an abstract notion of a web service that acts upon some resource to either retrieve information about an identity, update information about an identity, or perform some action for the benefit of some identity.

There are different types of identity services, each of which has a unique *service type*, identified by a URI. This service type identifier maps to exactly one *abstract WSDL* definition of a service, which contains the `wsdl:types`, `wsdl:message`, and `wsdl:portType` elements of a WSDL 1.1 description. An example of a service type is a "calendar service," which could be identified by a URI such as *urn:example:services:calendar*.

A service instance is the physical instantiation of a particular type of identity service. A service instance maps to a *concrete WSDL* document (which includes at least the `wsdl:binding`, `wsdl:service`, and `wsdl:port` elements) that contains the *protocol endpoint* and additional information necessary for a client to communicate with the particular service instance (e.g., this information may include security policy information). Each service instance is hosted by some provider, which is identified by a URI. An example of a service instance is a SOAP-over-HTTP endpoint offering a calendar service.

A service instance exposes a protocol interface to a set of resources. A *resource* in this specification is either data related to some identity or a service acting for the benefit of some identity. An example of a resource is a calendar containing appointments for a particular identity. When a client sends a request message to a service instance, it includes the resource identifier (i.e., a URI) for the resource it wishes the service instance to act upon.

A resource commonly has access control policies associated with it. These access control policies are typically under the purview of the entity or entities associated with the resource (in common language, the entity or entities could be said to "own" the resource). The access control policies on a resource must be enforced by the service instance.

The discovery service defined here is not intended to be exclusive. Some identity services meeting the conceptual model may be exposed via other discovery mechanisms. For example, [LibertyPAOS] defines an equivalent discovery mechanism.

1.2. Scope

This specification contains:

- Schemas for service instance enumeration and resource offering description.
- Specification of a discovery service that facilitates discovery and invocation of resource offerings.
- A SAML (see [SAMLCore]) attribute designator so that a resource offering for the discovery service itself can be conveyed via SAML assertions.

1.3. Notation and Conventions

This specification uses schema documents conforming to W3C XML Schema (see [Schema1]) and normative text to describe the syntax and semantics of XML-encoded 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 RFC 2119.

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.

The following namespaces are referred to in this document:

- The prefix disco: stands for the Discovery Service namespace. This namespace is the default for instance fragments, type names, and element names in this document. In schema listings, this is the default namespace and no prefix is shown.
- The prefix ds11: stands for the Discovery Service 1.1 extensions namespace. This namespace is used for new elements that were added in version 1.1 of this specification for use in Extension elements from the base schema.
- The prefix wsdl: stands for the primary WSDL namespace (http://schemas.xmlsoap.org/wsdl/).
- The prefix wsdlsoap: stands for the namespace of the WSDL-SOAP binding (http://schemas.xmlsoap.org/wsdl/soap/).
- The prefix xs: stands for the W3C XML schema namespace (http://www.w3.org/2001/XMLSchema).
- The prefix xsi: stands for the W3C XML schema instance namespace (http://www.w3.org/2001/XMLSchema-instance).

2. Common Types

Several XML Schema ComplexType and element declarations are used throughout this specification.

2.1. ServiceType

The `ServiceType` element is used to identify a service type. This URI needs be constant across all implementations of a service to enable interoperability. Therefore, it is RECOMMENDED that this URI be the same as the targetNamespace URI of the abstract WSDL description for the service.

```
<xs:element name="ServiceType" type="xs:anyURI"/>
```

Some example of possible ServiceType URIs:

```
urn:liberty:disco:2003-08
urn:liberty:id-sis-pp:2003-08
http://example.com/my-service-wsdl-ns
http://example.com/wsdl/my-service.wsdl
```

2.2. ResourceID

The `ResourceID` element contains a URI used to identify a particular resource. Resource identifiers can be registered with the discovery service. Queries on the discovery service return the resource identifiers of matching resources (along with a service instance description that describes how to access the resource). The format of resource IDs will vary from service instance to service instance. Clients should never need to construct resource IDs; clients will obtain them from the discovery service.

The following constraints are imposed on the acceptable values for `ResourceID`:

- It MUST not be a relative URI (see [RFC2396].)
- The resource URI SHOULD contain a domain name which is owned by the provider hosting the resource.
- If a resource is exposed via multiple `ResourceOffering` elements (for example, if there are multiple service instances used as a front end to the same data), all such `ResourceOffering` elements SHOULD have the same `ResourceID` value.

One special value of `ResourceID` is defined: "urn:liberty:isf:implied-resource". This resource identifier is to be used in circumstances where the resource in question is implicitly identified because there is only one resource that could be operated upon at the service instance being contacted. In some circumstances, the use of this resource identifier can eliminate the need for contacting the discovery service to access the resource. (See [LibertyPAOS] for some example uses.)

The ResourceID element is optional (minOccurs="0"). If the ResourceID element is not present, the message should be processed the same way as if the ResourceID element was present and its value was "urn:liberty:isf:implied-resource". Note that in version 1.0 of this specification ResourceID was not optional and so older implementations may not process messages which have omitted ResourceID instead of including it with this value.

```
<xs:complexType name="ResourceIDType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="EncryptedResourceIDType">
  <xs:sequence>
    <xs:element ref="xenc:EncryptedData"/>
    <xs:element ref="xenc:EncryptedKey"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="ResourceID" type="ResourceIDType"/>
<xs:element name="EncryptedResourceID" type="EncryptedResourceIDType"/>

<!--
    if not present, equivalent to
    <ResourceID>urn:liberty:isf:implied-resource</ResourceID>
    (see specification text for details)
  -->

<xs:group name="ResourceIDGroup">
  <xs:sequence>
    <xs:choice minOccurs="0" maxOccurs="1">
      <xs:element ref="ResourceID"/>
      <xs:element ref="EncryptedResourceID"/>
    </xs:choice>
  </xs:sequence>
</xs:group>
```

Some examples of possible resource IDs are:

```
http://example.com/disco/d0CQF8elJTDLmzEo
http://profile-provider.com/profiles/l4m0B82k15csaUxs
urn:liberty:isf:implied-resource
```

2.3. EncryptedResourceID

The schema also defines an element `EncryptedResourceID` for the transport of obfuscated resource identifiers. The `EncryptedResourceID` contains a `ResourceID` that has been encrypted using XML encryption, and an encrypted key that was used to encrypt the `ResourceID`. Use of `EncryptedResourceID` is often necessary for privacy reasons.

If a non-predictable nonce is used for the EncryptedKey, each discovery service client will get a different identifier. This will prevent discovery service clients from colluding with other discovery service clients about the identity of the

Principal on the basis of the ResourceID for one of the Principal's services. (Note that strictly speaking, the key does not need to be a nonce, rather just unique for each client.)

- The `xenc:EncryptedData`, when successfully decrypted, MUST contain a `ResourceID` element.
- The `xenc:EncryptedData` MUST be encrypted using the `xenc:EncryptedKey` that is present in the same element.
- The key used to encrypt the key in `xenc:EncryptedKey` MUST be the public key of the provider that hosts the resource.
- [LibertyMetadata] defines a mechanism to retrieve the provider's public key if the Provider ID is known; the Provider ID is available in the resource's service instance description (see Section 3). [LibertyMetadata] can also be used to ensure that the encryption algorithm being used is supported by both parties.
- The `xenc:EncryptedKey` MUST exhibit nonce-like semantics, so that it does not circumvent the privacy requirement that the `EncryptedResourceID` mechanism is intended to address. However, an `xenc:EncryptedKey` MAY be used multiple times with the same client, so long as the same encrypted identifier is never delivered to more than one party.
- Because the `xenc:EncryptedKey` element is used for key transport, the `xenc:Algorithm` attribute of the `xenc:EncryptionMethod` element must be one of the URIs designated for key transport as defined in [xmlenc-core].

2.4. Status Codes

The following status code QNames are defined in the Discovery Service namespace:

- *OK*: message processing succeeded
- *Failed*: general failure code
- *RemoveEntry*: an entry being removed does not exist
- *Forbidden*: the request was denied based on policy
- *NoResults*: no results could be found
- *Directive*: a directive was supplied in InsertEntry that was not understood or not supported

These QNames are expected to appear in the "code" attribute of `Status` elements used in Discovery Service protocol messages. Specific uses for the status codes are defined in the processing rules for individual messages. The "ref" attribute on the `Status` element is not used in this specification, so it MUST not appear on Status elements in Discovery Service protocol messages. The contents of the "comment" attribute are not defined by this specification, but it may be used for additional descriptive text intended for human consumption (for example, to carry information that will aid debugging).

3. Service Instance Description

A service instance is a running web service at a distinct protocol endpoint. Information about service instances needs to be communicated in various contexts. For example, the Discovery Service defined in this specification is an identity service which provides an enumeration of resource offerings (each of which includes a service instance description). This specification defines a schema for service instance description that can be used in a variety of protocol interactions. Note that this description schema does not replace WSDL; rather, it is to be used in conjunction with WSDL. In essence, it wraps WSDL with additional information and allows for enumeration of various service instances described with WSDL.

```
<xs:complexType name="DescriptionType">
  <xs:sequence>
    <xs:element name="SecurityMechID" type="xs:anyURI" minOccurs="1" maxOccurs="unbounded"/>
    <xs:element name="CredentialRef" type="xs:IDREF" minOccurs="0" maxOccurs="unbounded"/>
    <xs:choice>
      <xs:group ref="WsdlRef"/>
      <xs:group ref="BriefSoapHttpDescription"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>

<xs:complexType name="ServiceInstanceType">
  <xs:sequence>
    <xs:element ref="ServiceType"/>
    <xs:element name="ProviderID" type="md:entityIDType"/>
    <xs:element name="Description" type="DescriptionType" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

`ServiceType` contains the URI defining the type of service this service instance implements. Because `ServiceType` maps one-to-one with abstract WSDL (see Section 1.1), the `ServiceType` element enables a potential requester to know the abstract WSDL implemented by the service instance.

`ProviderID` contains the URI of the provider of the service instance. This is useful for resolving metadata (e.g., trust metadata) necessary for invoking the service instance. Note that a single physical provider may have multiple provider IDs. At registration time, the provider can ensure that the `ServiceInstance` element contains the appropriate `ProviderID` element given the context (for example, based on the circle of trust that the discovery service instance is in). `ProviderID` MUST correspond with the constraints for the `entityIDType` data type as specified in [LibertyMetadata].

The `Description` element contains the necessary concrete service description (see Section 1.1) is included in the Description via a choice between a `WsdlRef` group (which references an external concrete WSDL resource) or a `BriefSoapHttpDescription` group (which provides inline the information necessary to invoke basic SOAP-over-HTTP-based service instances without using WSDL). The BriefSoapHttpDescription group MUST be chosen if and only if it is possible to logically compute the concrete WSDL from the abstract WSDL (as referred to by the service type URI) and the information contained in the BriefSoapHttpDescription. (See Section 3.2 below.) Otherwise, the `WsdlRef` group must be used. The purpose of having the `BriefSoapHttpDescription` choice is to ease the burden of ServiceInstance processors from having to retrieve and parse WSDL in common cases.

The `Description` contains an optional *id* attribute. The id attribute need not be specified unless it is necessary to refer to the description elsewhere. In particular, it may be needed to refer to a particular `Description` from a directive (see Section 5.2.1.1) in the `Modify` request.

The `Description` contains one or more `SecurityMechID` URIs. These URIs identify the security mechanisms supported by the service instance. Other specifications, such as [LibertySecMech] define the actual security mechanisms along with their identifiers. These security mechanisms refer to the way a web service client authenticates to the web service provider. The service instance description SHOULD list of all of the security mechanisms that the service

instance supports. The client SHOULD pick the first mechanism (in the order listed) that it supports; the description SHOULD list them in order of preference, to avoid situations where the client fails to gain access to the service because it picked the wrong security mechanism.

The `Description` may also contain `CredentialRef` elements. These elements refer to security credentials (which are contained elsewhere; see Section 5.1.2) which the client may need in order to invoke the service using the given security mechanism.

Multiple `Description` elements are allowed in case the WSDL binding varies between security mechanisms. (For example, many web servers will require a different endpoint URI to be used for SOAP/HTTP clients authenticating using client TLS certificates than for clients which do not authenticate using client TLS certificates.) Any single `SecurityMechID` URI MUST NOT appear in more than one Description in a particular service instance description. In other words, each service instance may only specify one WSDL binding per supported security mechanism. The descriptions SHOULD appear in the order of the service's preference, and the client SHOULD use the first description in the list that it is capable of accessing.

3.1. WsdlRef Group

`WsdlURI` provides a URI to a WSDL resource containing the service description. This must be concrete WSDL (see Section 1.1), not abstract WSDL (see Section 1.1). The `ServiceNameRef` references a `wsdl:service` element within the WSDL resource such that `ServiceNameRef` is equal to the `wsdl:name` attribute of the proper `wsdl:service` element. The specified `ServiceNameRef` MUST refer to a `wsdl:service` that implements bindings to the portTypes defined by the `ServiceType` URI. The processor of the `ServiceInstance` chooses the proper `wsdl:service` element by this means. The specified WSDL resource MUST contain a `wsdl:service` with a `wsdl:name` attribute equal to the specified `ServiceNameRef`.

```
  <xs:group name="WsdlRef">
    <xs:sequence>
      <xs:element name="WsdlURI" type="xs:anyURI"/>
      <xs:element name="ServiceNameRef" type="xs:QName"/>
    </xs:sequence>
  </xs:group>
```

An example service instance description using a WSDL reference is:

```
   <ServiceInstance xmlns="urn:liberty:disco:2003-08">
      <ServiceType>http://example.com/wsdl/my-service.wsdl</ServiceType>
      <ProviderID>http://example.com/</ProviderID>
      <Description>
         <SecurityMechID>urn:liberty:security:2003-08:TLS:null</SecurityMechID>
         <SecurityMechID>urn:liberty:security:2005-02:TLS:X509</SecurityMechID>
         <WsdlURI>http://example.com/wsdl/my-service.wsdl</WsdlURI>
         <ServiceNameRef xmlns:m="http://example.com/wsdl/my-service.wsdl">
m:MyService</ServiceNameRef>
      </Description>
   </ServiceInstance>
```

3.2. BriefSoapHttpDescription Group

The information contained in this group is sufficient for making invocations for some service instances. In other words, the information contained in this group together with the abstract WSDL specified by the ServiceType URI is sufficient to logically compute concrete WSDL with the rule set specified below. If the service instance exposes an endpoint that is different from the logically generated concrete WSDL, the WsdlRef group MUST be used instead.

`Endpoint` contains the URI of the SOAP-over-HTTP endpoint. The URI scheme MUST be "http" or "https". `SoapAction` contains the equivalent of the `wsdlsoap:soapAction` attribute of the `wsdlsoap:operation` element in WSDL-based description.

Use of this group implies `wsdl:binding` and `wsdl:service` elements according to the following rules (i.e., the concrete WSDL can be logically computed given the abstract WSDL and the BriefSoapHttpDescription group):

- The `wsdl:binding` contains a `wsdlsoap:binding` element. This specifies that the SOAP binding for WSDL is being used.
- The `style` attribute of the `wsdlsoap:binding` element is *document*.
- The `transport` attribute of the `wsdlsoap:binding` element is *http://schemas.xmlsoap.org/soap/http*.
- The abstract WSDL corresponding to the `ServiceType` MUST contain a single `portType` element. The `wsdl:binding` element provides bindings for the operations specified in this `wsdl:portType`. Each operation binding includes an input element and an output element, each containing a single `wsdlsoap:body` element. The `use` attribute of the `wsdlsoap:body` elements is "literal".
- The `soapAction` attribute of `wsdlsoap:operation` is equal to `SoapAction` if provided, otherwise it is omitted.
- The `location` attribute of `wsdlsoap:address` is equal to `Endpoint`.
- All other optional elements and attributes are not specified and thus default to the SOAP binding of WSDL.

```
<xs:group name="BriefSoapHttpDescription">
  <xs:sequence>
    <xs:element name="Endpoint" type="xs:anyURI"/>
    <xs:element name="SoapAction" type="xs:anyURI" minOccurs="0"/>
  </xs:sequence>
</xs:group>
```

An example ServiceInstance using the BriefSoapHttpDescription is:

```
<ServiceInstance xmlns="urn:liberty:disco:2003-08">
   <ServiceType>urn:liberty:id-sis-pp:2003-08</ServiceType>
   <ProviderID>http://profile-provider.com/</ProviderID>
   <Description>
      <SecurityMechID>urn:liberty:security:2003-08:TLS:null</SecurityMechID>
      <SecurityMechID>urn:liberty:security:2005-02:TLS:X509</SecurityMechID>
      <Endpoint>https://soap.profile-provider.com/soap/</Endpoint>
   </Description>
   <Description>
      <SecurityMechID>urn:liberty:security:2003-08:ClientTLS:null</SecurityMechID>
      <Endpoint>https://soap-auth.profile-provider.com/soap/</Endpoint>
   </Description>
</ServiceInstance>
```

4. Resource Offering Description

A *resource offering* is the association of a resource and a service instance. This association is necessary as there is a many-to-many relationship between resources and service instances. Typically, a single service instance will serve many resources. For example, a personal profile service provider would typically serve up many profiles behind a single service instance, as having a separate protocol endpoint for each profile would be impractical. Thus, a `ResourceOffering` element is defined to associate a resource with a service instance that provides access to that resource.

```
<xs:element name="ResourceOffering" type="ResourceOfferingType"/>
<xs:complexType name="ResourceOfferingType">
  <xs:sequence>
    <xs:group ref="ResourceIDGroup"/>
    <xs:element name="ServiceInstance" type="ServiceInstanceType"/>
    <xs:element ref="Options" minOccurs="0"/>
    <xs:element name="Abstract" type="xs:string" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="entryID" type="IDType" use="optional"/>
</xs:complexType>
```

The `ResourceID` element provides the URI for the resource that the requester can use in a request to the described `ServiceInstance`. The `EncryptedResourceID` MAY appear in place of the `ResourceID`. Therefore, service instances need to handle the case where the client presents an `EncryptedResourceID` instead of a `ResourceID`. The `ServiceInstance` contains the description of the service instance that is providing access to the resource.

The `Abstract` element contains a human-readable description of the resource offering.

The `Options` element expresses the "options" available for the resource offering, which provides hints to a potential requester whether certain data or operations may be available with a particular resource offering. For example, an option may be provided stating that home contact information is available. If no Options element is present, it means that the service instance does not advertise whether any options are available (for example, it may be a simple service that is not capable of updating its entry in the discovery service when the available options change, so it avoids listing them at all.) If the Options element is present, but it is empty, it means that the service instance explicitly advertises that none of the options are available.

The `Options` element contains zero or more `Option` elements, each of which contain a URI identifying the particular option. The set of possible URIs for an `Option` element should be defined by the service type (e.g., a person profile service specification standardize a set of options). However, one common `Option` flag related to security and common to ID-WSF services is defined in Section 7.

```
<xs:element name="Options" type="OptionsType"/>
<xs:complexType name="OptionsType">
  <xs:sequence>
    <xs:element name="Option" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

An example ResourceOffering is:

```
<ResourceOffering xmlns="urn:liberty:disco:2003-08">
   <ResourceID>http://profile-provider.com/profiles/l4m0B82k15csaUxs</ResourceID>
   <ServiceInstance xmlns="urn:liberty:disco:2003-08">
      <ServiceType>urn:liberty:id-sis-pp:2003-08</ServiceType>
      <ProviderID>http://profile-provider.com/</ProviderID>
      <Description>
         <SecurityMechID>urn:liberty:security:2003-08:TLS:null</SecurityMechID>
         <SecurityMechID>urn:liberty:security:2005-02:TLS:X509</SecurityMechID>
         <Endpoint>https://soap.profile-provider.com/soap/</Endpoint>
```

```
        </Description>
        <Description id="saml-profile-description">
           <SecurityMechID>urn:liberty:security:2005-02:TLS:SAML</SecurityMechID>
           <Endpoint>https://soap.profile-provider.com/soap/</Endpoint>
        </Description>
        <Description>
           <SecurityMechID>urn:liberty:security:2003-08:ClientTLS:null</SecurityMechID>
           <Endpoint>https://soap-auth.profile-provider.com/soap/</Endpoint>
        </Description>
     </ServiceInstance>
     <Options>
        <Option>urn:liberty:id-sis-pp</Option>
        <Option>urn:liberty:id-sis-pp:cn</Option>
        <Option>urn:liberty:id-sis-pp:can</Option>
        <Option>urn:liberty:id-sis-pp:can:cn</Option>
     </Options>
     <Abstract>
        This is a personal profile containing common name information.
     </Abstract>
  </ResourceOffering>
```

5. Discovery Service

The Discovery Service is an identity service that allows requesters to discover resource offerings. Thus, the Discovery Service is essentially a web service interface for "discovery resources", each of which can be viewed as a registry of resource offerings. Entities can place resource offerings in a discovery resource, and this will allow other entities to discover these resource offerings. A common use case is that a user places his or her personal profile, calendar, and so on a discovery resource so that these other resources can be discovered by other entities.

Note that the Discovery Service itself is an identity service like any other. Also note that other discovery mechanisms are possible; this specification formalizes one particular mechanism that can be used in a wide variety of applications.

The Discovery Service is meant to be used in conjunction with the lower-layer ID-WSF specifications. For example, security mechanisms are not specified here, because they are defined in [LibertySecMech]. At the same time, the Discovery Service is specified such that it can be used with other security mechanisms, not yet defined.

The Discovery Service is defined by the abstract WSDL in this document (see [WSDLv1.1]), which defines two `wsdl:operation` definitions. The *DiscoveryLookup* operation returns an enumeration of `ResourceOffering` elements given search criteria. The *DiscoveryUpdate* operation enables maintenance of a discovery resource, accommodating inserts and removals of resource offerings.

The discovery service MAY return `EncryptedResourceID` elements instead of the `ResourceID` in the `QueryResponse` (except in the case of the `EncryptResourceID` directive (see Section 5.2.1.1). Note that it may decide to encrypt the identifier based on policy, or it may decide not to do so because of implementation limitations.

However, some identity services may not wish to register with discovery services that do not encrypt resource IDs for whatever reason. The `EncryptResourceID` directive is provided to allow the identity service to require the discovery service to either always encrypt the resource ID or refuse the registration. The intent is that, in general, the discovery service that is describing the resource makes the decision, not the service instance hosting the resource in question, so even if this directive is not present, the discovery service may still decide to encrypt the resource ID.

Therefore, identity services which are registered with the discovery service MUST be prepared to accept `EncryptedResourceID` from their clients instead of `ResourceID`. (Identity services which do not wish to expose the `ResourceID` to the discovery service unless it will encrypt it for privacy reasons may do so by using the `EncryptResourceID` directive (see Section 5.2.1.1).

To enforce access control policies, security credentials may need to be presented by the client. While the definition of these security credentials is outside the scope of this specification, it is common in many cases for the same entity that is hosting the discovery service to also be the entity that generates the credentials necessary to access the service. To avoid extra network round-trips, arrangements are made here so that credentials may be provided as part of the discovery service lookup response.

The Discovery Service service type URI is *urn:liberty:disco:2003-08*.

5.1. Operation: DiscoveryLookup

The DiscoveryLookup operation enables a requester to obtain an enumeration of `ResourceOffering` elements. The requester sends a `Query` and receives a `QueryResponse` in return. Also, because a provider hosting a Discovery Service may also be playing other roles for an identity (such as a *Policy Decision Point* or an *Authentication Authority*), the *DiscoveryLookup* operation can also function as a security credential service, providing the requester with an efficient means of obtaining credentials that may be necessary to invoke service instances described in the `QueryResponse`.

5.1.1. Query

The `Query` minimally contains the `ResourceID` element, which describes which discovery resource is being requested. A request with only the `ResourceID` element indicates the requester is requesting all available resource offerings. The set of results is dependant upon local access control policy of the discovery resource.

The request can be qualified with a set of `RequestedServiceType` elements, which enables the requester to specify that all resource offerings returned must be offered via a service instance complying with one of the specified service types. For each `ServiceType` specified, the requester can also specify `Options` (see Section 4) the returned resource offering should support. Note that returned resource offerings are not guaranteed to support the requester-specified options, as some discovery service instances and/or resource offering registrations may not support options registration.

Requesters SHOULD construct a Query to be as qualified as possible, as the discovery service provider may have to perform significant work for each result in the response, especially if credentials are going to be generated.

```
<xs:element name="Query" type="QueryType"/>
<xs:complexType name="QueryType">
  <xs:sequence>
    <xs:group ref="ResourceIDGroup"/>
    <xs:element name="RequestedServiceType" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="ServiceType"/>
          <xs:element ref="Options" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>
```

An example SOAP message containing a Query follows:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <Correlation id="e3Bvm4gNw_IrVJiEpU50"
         soap:mustUnderstand="1"
         soap:actor="http://schemas.xmlsoap.org/soap/actor/next"
         messageId="NK44V79NdfPaE5jCwlk_"
         xmlns="urn:liberty:sb:2003-08"/>
   </soap:Header>
   <soap:Body>
      <Query xmlns="urn:liberty:disco:2003-08">
         <ResourceID>http://example.com/disco/d0CQF8elJTDLmzEo</disco:ResourceID>
         <RequestedServiceType>
            <ServiceType>urn:liberty:id-sis-pp:2003-08</disco:ServiceType>
         </RequestedServiceType>
      </Query>
   </soap:Body>
</soap:Envelope>
```

5.1.2. QueryResponse

The `QueryResponse` element contains the set of results of the query as a set of `ResourceOffering` elements. Each `ResourceOffering` element MUST contain an *entryID* attribute, to be used in the *Modify* message. This *entryID* MUST be unique across all entries in the discovery resource being queried.

A set of credentials may be provided within the `Credentials` element in the response. All credentials MUST have an attribute of type *ID*, so that they can be referred to via an IDREF. The relevant credentials for each `ResourceOffering` are referenced with a set of zero of more `CredentialRef` elements contained in the service instance description (see Section 3).

A status code is also included in the response.

```
<xs:element name="QueryResponse" type="QueryResponseType"/>
<xs:complexType name="QueryResponseType">
  <xs:sequence>
    <xs:element ref="Status"/>
    <xs:element ref="ResourceOffering" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Credentials" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>
```

An example SOAP message containing a QueryResponse follows. This example includes a credential and a ResourceOffering which references the credential. Parts of the credential have been omitted due to size.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <Correlation id="HE-XCr00O5_3rXLDqKWM"
         soap:mustUnderstand="1"
         soap:actor="http://schemas.xmlsoap.org/soap/actor/next"
         refToMessageId="NK44V79NdfPaE5jCwlk_"
         messageId="XIl-dyCnQzfRW-BD9Ngx"
         xmlns="urn:liberty:sb:2003-08" />
   </soap:Header>
   <soap:Body>
      <QueryResponse xmlns="urn:liberty:disco:2003-08">
         <Status code="OK"/>
         <ResourceOffering entryID="1">
            <ResourceID>http://example.com/pip/bob</ResourceID>
            <ServiceInstance>
               <ServiceType>urn:liberty:id-sis-pp:2003-08</ServiceType>
               <ProviderID>http://example.com/</ProviderID>
               <Description>
                  <SecurityMechID>urn:liberty:security:2005-02:TLS:SAML</SecurityMechID>
                  <CredentialRef>2sxJu9g/vvLG9sAN9bKp/8q0NKU=</CredentialRef>
                  <Endpoint>https://soap.example.com:443/soap</Endpoint>
               </Description>
            </ServiceInstance>
            <Abstract>Bob's personal profile</Abstract>
         </ResourceOffering>
         <Credentials>
            <saml:Assertion
               xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
               AssertionID="2sxJu9g/vvLG9sAN9bKp/8q0NKU="
               Issuer="idp.example.com"
               IssueInstant="2003-09-09T16:58:33.173Z">

               <saml:AuthenticationStatement
                  AuthenticationMethod="urn:ietf:rfc:2246"
                  AuthenticationInstant="2003-09-09T16:57:30.000Z">
                  <saml:Subject>
                     <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
                        http://serviceprovider.com/
                     </saml:NameIdentifier>
                  </saml:Subject>
               </saml:AuthenticationStatement>

               <ResourceAccessStatement
                  xmlns="urn:liberty:sec:2003-08"
```

```
                    xmlns:disco="urn:liberty:disco:2003-08">
                    <saml:Subject>
                       <saml:NameIdentifier format="urn:liberty:iff:nameid:entityID">
                          http://serviceprovider.com/
                       </saml:NameIdentifier>
                       <saml:SubjectConfirmation>
                          <saml:ConfirmationMethod>
                             urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
                          </saml:ConfirmationMethod>
                          <ds:KeyInfo>
                             <ds:KeyName>
                                CN=serviceprovider.com,OU=Services R US,O=Service Nation,...
                             </ds:KeyName>
                             <ds:KeyValue>...</ds:KeyValue>
                          </ds:KeyInfo>
                       </saml:SubjectConfirmation>
                    </saml:Subject>

                    <disco:ResourceID>http://example.com/pip/bob</disco:ResourceID>
                    <SessionContext xmlns="urn:liberty:sec:2003-08"
                       AuthenticationInstant="" AssertionIssueInstant="">
                       <SessionSubject>
                          <saml:NameIdentifier ID="..."
                             Format="urn:liberty:iff:nameid:encrypted">
                             HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw
                             A23B45C569UkjjLLA6nNvBX7mY00TZhwBdFNDElgscSXZ5Ekw89XR3==
                          </saml:NameIdentifier>
                       </SessionSubject>
                       <AuthnContext xmlns="http://www.projectliberty
.org/schemas/authctx/2002/05">
                          ...
                       </AuthnContext>
                       <ProviderID>http://serviceprovider.com/</ProviderID>
                    </SessionContext>
                 </ResourceAccessStatement>
                 <ds:Signature>...</ds:Signature>
              </saml:Assertion>
           </Credentials>
        </QueryResponse>
     </soap:Body>
  </soap:Envelope>
```

5.1.3. Processing Rules

The discovery service provider returns entries based on the requester's criteria, the policies of the discovery resource, and the contents of the discovery resource. For each `RequestedServiceType`, the following matching rules MUST be followed in determining the subset of result that will be returned to the requester:

- If no `Options` element is present in the ResourceOffering, all entries in the discovery resource with the specified service type match.
- A `ResourceOffering` in the discovery resource matches the query if each `Option` element value in the `RequestedServiceType` is also present in the `ResourceOffering`. Note that this means that if there are no `Option` elements in the `RequestedServiceType`, the resource matches. (See Section 4.)

The discovery service provider SHOULD provide credentials in the response if it knows those credentials are necessary based on the directives provided when the resources being discovered were registered.

The discovery service provider MAY order `ResourceOffering` elements as it sees fit. If the discovery service is rank ordering the entries, it MUST use descending rank order. This enables the requester to assume that if the results were ordered, the first result is the most relevant.

The following rules specify the status code in the response:

- If request processing succeeded, the top-level status code MUST be *OK*. Otherwise, the top-level status code MUST be *Failed*.

- If the top-level status code is *Failed*, the response MAY also contain *Forbidden* or *NoResults* as a second-level status code. *Forbidden* MUST be only be used if the request was denied based on policy such that no future Modify request would put the resource in a state that any results could be returned. *NoResults* MUST only be used if there are no ResourceOffering elements in the response, but there might be some ResourceOffering elements if an entry matching the criteria was later inserted.
  The service may not wish to reveal the reason for failure, in which case no second-level status code will appear. Other second-level status codes from the Discovery Service namespace MUST NOT be used. Other second-level status codes from other namespaces MAY appear. Clients MAY ignore status codes from other namespaces if they are not understood.

5.2. Operation: DiscoveryUpdate

The DiscoveryUpdate operation enables a requester to insert new resource offering entries into a discovery resource and remove existing entries from a discovery resource. The DiscoveryUpdate allows multiple insertions and removals to be made in a single request. Updates to existing entries are performed by removing an existing entry and inserting a new entry in a single operation.

5.2.1. Modify

The `Modify` element contains a set of zero or more `InsertEntry` elements, each containing exactly one `ResourceOffering` element, and a set of zero or more `RemoveEntry` elements, each containing an *entryID* attribute.

`ResourceOffering` elements being inserted MUST NOT contain entryID attributes.

Note that the `InsertEntry` definition contains an `any` element. This allows the requester to include directives about the `ResourceOffering` being inserted. For example, access control policy for the resource offering could be specified. This specification defines several standard directives that can be used in this placeholder.

```
<xs:complexType name="InsertEntryType">
  <xs:sequence>
    <xs:element ref="ResourceOffering"/>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="RemoveEntryType">
  <xs:attribute name="entryID" type="IDReferenceType" use="required"/>
</xs:complexType>

<xs:element name="Modify" type="ModifyType"/>
<xs:complexType name="ModifyType">
  <xs:sequence>
    <xs:group ref="ResourceIDGroup"/>
    <xs:element name="InsertEntry" type="InsertEntryType" minOccurs="0" maxOccurs="unbounded"/>
```

```
      <xs:element name="RemoveEntry" type="RemoveEntryType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="optional"/>
  </xs:complexType>
```

An example SOAP message containing a Modify follows. This request removes an existing resource (see the QueryResponse example) and replaces it with a different one.

```
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
      <Correlation id="CkXlpwJuNxn9ssDsEt5l"
        soap:mustUnderstand="1"
        soap:actor="http://schemas.xmlsoap.org/soap/actor/next"
        messageId="DNtVj92yV1iICR47w1QV"
        xmlns="urn:liberty:sec:2003-08"/>
    </soap:Header>
    <soap:Body>
      <Modify xmlns="urn:liberty:disco:2003-08">
        <ResourceID>http://example.com/disco/d0CQF8elJTDLmzEo</disco:ResourceID>
          <InsertEntry>
            <ResourceOffering xmlns="urn:liberty:disco:2003-08">
              <ResourceID>http://profile-provider.com/profiles/l4m0B82k15csaUxs</Resour
ceID>
              <ServiceInstance xmlns="urn:liberty:disco:2003-08">
                <ServiceType>urn:liberty:id-sis-pp:2003-08</ServiceType>
                <ProviderID>http://profile-provider.com/</ProviderID>
                <Description id="clientTLS">
                  <SecurityMechID>urn:liberty:security:2003-08:Client
TLS:null</SecurityMechID>
                  <Endpoint>https://soap-auth.profile-provider.com/soap/</Endpoint>
                </Description>
                <Description id="saml">
                  <SecurityMechID>urn:liberty:security:2005-02:TLS:SAML</SecurityMechID>
                  <Endpoint>https://soap.profile-provider.com/soap/</Endpoint>
                </Description>
                <Description>
                  <SecurityMechID>urn:liberty:security:2005-02:TLS:X509</SecurityMechID>
                  <SecurityMechID>urn:liberty:security:2003-08:TLS:null</SecurityMechID>
                  <Endpoint>https://soap.profile-provider.com/soap/</Endpoint>
                </Description>
                <Description>
                  <SecurityMechID>urn:liberty:security:2003-08:null:null</SecurityMechID>
                  <Endpoint>http://soap.profile-provider.com/soap/</Endpoint>
                </Description>
              </ServiceInstance>
              <Options>
                <Option>urn:liberty:id-sis-pp</Option>
                <Option>urn:liberty:id-sis-pp:cn</Option>
                <Option>urn:liberty:id-sis-pp:can</Option>
                <Option>urn:liberty:id-sis-pp:can:cn</Option>
              </Options>
              <Abstract>
                This is a personal profile containing common name information.
              </Abstract>
            </ResourceOffering>
            <AuthenticateRequester descriptionIDRefs="saml"/>
            <AuthorizeRequester descriptionIDRefs="saml clientTLS"/>
          </InsertEntry>
          <RemoveEntry entryID="1"/>
      </Modify>
    </soap:Body>
  </soap:Envelope>
```

5.2.1.1. Directives

Six policy-related directives are also defined: `AuthenticateRequester`, `AuthorizeRequester`, `AuthenticateSessionContext`, `EncryptResourceID`, `ds11:SendSingleLogOut`, and `ds11:GenerateBearerToken`.

The directives all contain an optional descriptionIDRefs attribute. If the *descriptionIDRefs* attribute is not present in a directive element, the directive is to be taken to apply to all `Description` elements provided in the `ResourceOffering`. If the *descriptionIDRefs* attribute is present in a directive element, it MUST contain a list of IDREFs which refer to `Description` elements in the ResourceOffering the directive is associated with. If the attribute is present, the directive MUST be taken to apply only to those descriptions referred to in the descriptionIDRefs list. This may be useful if certain directives are incompatible with certain security mechanisms.

If the `AuthenticateRequester` directive is specified for a resource, the discovery service provider SHOULD include a SAML assertion containing an AuthenticationStatement in any future `QueryResponse` messages for the resource to enable the client sending the Query message to authenticate to the service instance hosting the resource. If the client which sends the Query authenticates using a `ProviderID`, the statement SHOULD be a `saml:AuthenticationStatement` as defined in [LibertySecMech]. In this case, the statement's subject is the client's ProviderID. If the client which sends the Query is a user agent which does not authenticate using a `ProviderID` (that is, if no Provider header is present on the request message), the AuthenticationStatement included SHOULD be an ID-FF `lib:AuthenticationStatement`. In this case, the statement should identify the principal using the user agent using an ID-FF name identifier. (Note that since provider metadata cannot be accessed in this case, the statement will not be useful with the holder of key confirmation method and therefore not useful with [LibertySecMech]'s SAML security mechanism. However, it might be used with the bearer security mechanism in conjunction with the ds11:GenerateBearerToken directive.)

If the `AuthorizeRequester` directive is specified for a resource, the discovery service provider SHOULD include a SAML assertion containing a ResourceAccessStatement (as defined in [LibertySecMech]) in any future `QueryResponse` messages for the resource. The `AuthenticateSessionContext` directive is identical to the `AuthorizeRequester` directive except that the appropriate statement is a `SessionContextStatement`.

However, if both `AuthorizeRequester` and `AuthenticateSessionContext` are present, the discovery service provider SHOULD NOT generate both a ResourceAccessStatement and a SessionContextStatement; instead, it SHOULD generate a SAML assertion containing a ResourceAccessStatement, and the ResourceAccessStatement SHOULD contain a SessionContext element.

If credentials are provided in response to these directives, they MUST comply with the processing rules defined in [LibertySecMech].

The `AuthenticateRequester` directive MUST be used with any descriptions including the security mechanisms from [LibertySecMech] which use SAML for message authentication.

If the `EncryptResourceID` directive is included, the discovery service MUST NOT reveal the unencrypted resource ID to clients (i.e. when returning it in a QueryResponse). If the discovery service is not willing to honor the directive (for example, because it is not willing to encrypt the resource ID for policy reasons, or because it does not support resource ID encryption), it MUST fail the `Modify` request.

The `ds11:SendSingleLogOut` directive is provided for use with the `AuthenticateSessionContext` directive. The credentials received as a result of `AuthenticateSessionContext` may be used by some service instances to establish session state for principal. This directive allows the service instance to request that the provider issuing the credential for `AuthenticateSessionContext` deliver an ID-FF LogoutRequest message when the session state being conveyed is no longer valid. If the `ds11:SendSingleLogOut` directive is included, the discovery service MUST cause an ID-FF LogoutRequest to be delivered to the ID-FF Service Provider identified by the ProviderID in the ResourceOffering any time a session ends for which SessionContext has been sent to the provider by way of returning credentials in a Query for the provider.

The `ds11:GenerateBearerToken` directive is provided for use with the Bearer Token Authentication mechanism defined in [LibertySecMech]. It modifies the directives which are defined for use with the SAML authentication mechanism; in particular, the AuthenticateRequester, AuthorizeRequester, and AuthenticateSessionContext directives. If the `ds11:GenerateBearerToken` directive is specified for a resource, any SAML statements generated for the above directives MUST have ConfirmationMethod set to "urn:oasis:names:tc:SAML:1.0:cm:bearer". The resulting assertion is to be used as a token with the Bearer Token Authentication mechanism defined in [LibertySecMech]. (Note that the rules defined for the SAML Assertion Message Authentication mechanism do not apply to the token generated when `ds11:GenerateBearerToken` is present, even though it happens to be a SAML assertion.)

```
<xs:complexType name="DirectiveType">
  <xs:attribute name="descriptionIDRefs" type="xs:IDREFS" use="optional"/>
</xs:complexType>
<xs:element name="AuthenticateRequester" type="DirectiveType"/>
<xs:element name="AuthorizeRequester" type="DirectiveType"/>
<xs:element name="AuthenticateSessionContext" type="DirectiveType"/>
<xs:element name="EncryptResourceID" type="DirectiveType"/>


 <xs:element name="SendSingleLogOut" type="disco:DirectiveType"/>


 <xs:element name="GenerateBearerToken" type="disco:DirectiveType"/>
```

5.2.2. ModifyResponse

The response contains the following elements and attributes:

- `Status`: Contains status code; see processing rules.
- *newEntryIDs*: If the status is OK, and InsertEntry was present in the Modify request, the *newEntryIDs* attribute MUST contain the list of entry IDs assigned to the new entries. The list MUST be in the same order that the InsertEntry elements were in.
- `Extension`: Contains future extensions in other namespaces. One such extension is defined (see Section 8).

```
<xs:element name="ModifyResponse" type="ModifyResponseType"/>
<xs:complexType name="ModifyResponseType">
  <xs:sequence>
    <xs:element ref="Status"/>
<xs:element ref="Extension" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
  <xs:attribute name="newEntryIDs" use="optional">
    <xs:simpleType>
      <xs:list itemType="IDReferenceType"/>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
```

An example modify response follows.

```
 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
       <Correlation id="YhgnUXx9EMVACSWZiDlL"
          soap:mustUnderstand="1"
          soap:actor="http://schemas.xmlsoap.org/soap/actor/next"
          refToMessageId="DNtVj92yV1iICR47w1QV"
          messageId="AykqNGoN-K5kEmBzqPOU"
          xmlns="urn:liberty:soap-bind:2003-08"/>
```

```
      </soap:Header>
      <soap:Body>
         <ModifyResponse xmlns="urn:liberty:disco:2003-08" newEntryIDs="2">
            <Status code="OK"/>
         </ModifyResponse>
      </soap:Body>
   </soap:Envelope>
```

5.2.3. Processing Rules

- The transaction unit for this operation is the entire set of `InsertEntry` and `RemoveEntry` elements; they either all succeed or all fail. The discovery service provider MUST enforce this atomicity. (This is so that a client can easily update a ResourceOffering by removing the old one and inserting a new one.)

- For each InsertEntry element, the service instance MAY store the resource offering provided such that it can be retrieved (subject to policy) by future Query operations on the same ResourceID. If the service instance does not store the resource, it MUST return a Failed status code for the operation (and therefore not insert or remove any of the other entries provided).

- For each RemoveEntry element, the service instance searches for a previously inserted ResourceOffering with the given entryID. If the entryID cannot be located, the service instance MUST return a Failed status code for the operation. If the entryID can be located, the service instance SHOULD remove the entry if policy allows. If the entry is not removed, the service instance MUST return a Failed status code for the operation. (Note that if a Failed status code is returned, none of the RemoveEntry elements in the message may be processed.)

- Resource offerings contained in InsertEntry SHOULD NOT contain entryID values. If a resource offering to be inserted contains an entryID value, the discovery service MUST ignore the value. The discovery service is to pick a new entryID. The entryID MUST be unique within the discovery resource, but it MUST NOT be usable as a pseudonym for the user, for privacy reasons (it should not be possible to correlate two discovery responses for the same user from the entryIDs).

- If request processing succeeded, the top-level status code MUST be *OK*. Otherwise, the top-level status code MUST be *Failed*.

- If the top-level status code is *Failed*, the response MAY also contain *RemoveEntry*, *Directive* or *Forbidden* as a second-level status code. The service may not wish to reveal the reason for failure, in which case no second-level status code will appear. Other second-level status codes from the Discovery Service namespace MUST NOT be used. Other second-level status codes from other namespaces MAY appear. Clients MAY ignore status codes from other namespaces if they are not understood.

- If any directives are present in a request that the discovery service does not understand, or there are directives present that the discovery service understands but does not support, the discovery service SHOULD reject the entire request and SHOULD include the second-level status code `Directive` to indicate the reason for failure.

6. SAML AttributeDesignator for Discovery ResourceOffering

Entities which authenticate Principals using SAML may need to discover the location of the discovery service containing identity services for that Principal. This can be accomplished using the existing `saml:AttributeStatement` mechanism. To include a ResourceOffering for a Principal's discovery service in a SAML assertion, an AttributeStatement SHOULD be included according to the following rules:

- The `saml:AttributeName` MUST be "DiscoveryResourceOffering".
- The `saml:AttributeNamespace` MUST be "urn:liberty:disco:2003-08".
- One or more saml:AttributeValue element MUST be included which each contain a single `ResourceOffering` element for the Discovery Service specified above. The discovery service must contain identity services for the Principal identified in the Subject element inside the AttributeStatement.
- The `ResourceOffering` that is inside the `AttributeStatement` may contain CredentialRef elements referring to credentials that are necessary to access the discovery service. These IDs SHOULD resolve to an XML element contained within the SAML `Advice` element of the same `Assertion`.

An example AttributeStatement that might be found in a Liberty ID-FF AuthnResponse follows. Note that it does not include any credentials. If credentials are needed, they should be placed in the Advice element of the assertion containing the AttributeStatement. In this case, the Description should contain CredentialRef elements whose values are set to the IDs of the credentials.

```
<AttributeStatement xmlns="urn:oasis:names:tc:SAML:1.0:assertion">
   <Subject>
      <NameIdentifier Format="urn:liberty:iff:nameid:federated">
         d0CQF8elJTDLmzEo
      </NameIdentifier>
   </Subject>
   <Attribute AttributeName="DiscoveryResourceOffering"
      AttributeNamespace="urn:liberty:disco:2003-08">
      <AttributeValue>
         <ResourceOffering xmlns="urn:liberty:disco:2003-08">
            <ResourceID>http://example.com/disco/d0CQF8elJTDLmzEo</ResourceID>
            <ServiceInstance>
               <ServiceType>urn:liberty:disco:2003-08</ServiceType>
               <ProviderID>http://example.com/</ProviderID>
               <Description>
                  <SecurityMechID>urn:liberty:security:2005-02:TLS:X509</SecurityMechID>
                  <SecurityMechID>urn:liberty:security:2003-08:TLS:null</SecurityMechID>
                  <Endpoint>https://soap.example.com/</Endpoint>
                  <SoapAction>urn:liberty:disco:2003-08</SoapAction>
               </Description>
            </ServiceInstance>
            <Abstract>Discovery service</Abstract>
         </ResourceOffering>
      </AttributeValue>
   </Attribute>
</AttributeStatement>
```

In all cases, this AttributeStatement MUST carry a resource offering for the Liberty discovery service defined in this specification. Any other resource offerings are to be discovered by contacting the discovery service.

7. Option Value for Response Authentication

The service instance description provides a way for services to indicate to clients what mechanisms are necessary for the client to authenticate itself to the service via the SecurityMechID element. The SecurityMechID values defined by [LibertySecMech] also indicate whether the service uses peer entity authentication (for example, server-side SSL/TLS). However, a web service client may need to know whether the service will use message authentication (that is, whether the service will sign the response message) and may not be willing to use a service which does not sign its responses.

To avoid situations where a client requests data and then discovers it does not trust it because it is not signed, an `Option` value is defined: *urn:liberty:disco:2005-02:options:security-response-x509*

If a service instance always authenticates its response messages according to the "X.509v3 Certificate Message Authentication" mechanism in [LibertySecMech], descriptions of the service SHOULD include this option value. Otherwise, its description MUST NOT include this option value. Clients MAY include this option value in Query messages in order to locate only services which always authenticate their response messages. A service MAY authenticate its response messages even if this option value was not included in its description at the discovery service.

In case the service also supported the previous version of this specification [LibertySecMech11], it should be able to register two different endpoints at the discovery service, each of them with different Options values (one according to this version of the specification, another one according to the URN value specified in [LibertySecMech11]. This information would aid the client to determine which version of the WS-SMS specification ([wss-sms-draft] and/or [wss-sms]) is supported by the service, and the service will act accordingly, depending on the endpoint used by the client. Note that this behaviour only applies to the case when the client's request does not use message authentication mechanisms. Otherwise, it should be possible for the service to determine the version of the WS-SMS specification supported by the client by simply analyzing the `<wsse:Security>` header present in the request.

In general, it is recommended that services do not sign their responses unless they positively know that clients are able to perform message authentication and are aware of the version of the WS-SMS spec used by that client

8. Including keys in ModifyResponse

Some directives from the Modify request (see Section 5.2.1) may cause the Discovery service instance to generate signed credentials in QueryResponse messages for the resource offerings in question. When the service instance which included the directives receives the signed credentials from a client, it needs to be able to verify the Discovery service instance's signature on the credentials. Typically the metadata (see [LibertyMetadata]) for the Discovery service instance is sufficient for such information. In certain situations, such as when the Discovery service instance is hosted on a LUAD (see [LibertyClientProfiles]), it may not be feasible to assign the LUAD a ProviderID with which to obtain metadata. However, the key material still needs to be made available to service instances which register services with the discovery service and include such directives.

The Discovery service instance may include a ds11:Keys extension in the ModifyResponse in order to provide such keys. The Discovery service instance SHOULD NOT include the extension unless necessary (that is, in cases where the Discovery service instance has no ProviderID). The Discovery service instance SHOULD include the ds11:Keys extension in ModifyResponse messages if it has no ProviderID and the Modify message included a ResourceOffering for which the Discovery service instance intends to generate signed credentials.

```
<!-- For use in ModifyResponse Extension field -->
<xs:element name="Keys" type="ds11:KeysType"/>
<xs:complexType name="KeysType">
  <xs:sequence>
    <xs:element ref="md:KeyDescriptor" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The ds11:Keys element appears as a child of the Extension element. It contains one or more KeyDescriptor elements.

9. Version 1.2 Core XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:liberty:disco:2003-08"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:md="urn:liberty:metadata:2003-08"
   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
   xmlns="urn:liberty:disco:2003-08"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified">

  <xs:include schemaLocation="liberty-idwsf-utility-v1.1.xsd"/>

  <xs:import namespace="urn:liberty:metadata:2003-08"
   schemaLocation="liberty-metadata-v1.1.xsd"/>

  <xs:import namespace="http://www.w3.org/2001/04/xmlenc#"
   schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd"/>
  <xs:annotation>
    <xs:documentation>

      XML Schema from Liberty Discovery Service Specification.

      ### NOTICE ###

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

    </xs:documentation>
  </xs:annotation>
  <xs:element name="ServiceType" type="xs:anyURI"/>

  <xs:complexType name="ResourceIDType">
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
        <xs:attribute name="id" type="xs:ID" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="EncryptedResourceIDType">
    <xs:sequence>
      <xs:element ref="xenc:EncryptedData"/>
      <xs:element ref="xenc:EncryptedKey"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="ResourceID" type="ResourceIDType"/>
  <xs:element name="EncryptedResourceID" type="EncryptedResourceIDType"/>

  <!--
      if not present, equivalent to
      <ResourceID>urn:liberty:isf:implied-resource</ResourceID>
      (see specification text for details)
    -->

  <xs:group name="ResourceIDGroup">
    <xs:sequence>
      <xs:choice minOccurs="0" maxOccurs="1">
        <xs:element ref="ResourceID"/>
        <xs:element ref="EncryptedResourceID"/>
      </xs:choice>
    </xs:sequence>
  </xs:group>

  <xs:complexType name="DescriptionType">
    <xs:sequence>
```

```
      <xs:element name="SecurityMechID" type="xs:anyURI" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element name="CredentialRef" type="xs:IDREF" minOccurs="0" maxOccurs="unbounded"/>
      <xs:choice>
        <xs:group ref="WsdlRef"/>
        <xs:group ref="BriefSoapHttpDescription"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="optional"/>
  </xs:complexType>

  <xs:complexType name="ServiceInstanceType">
    <xs:sequence>
      <xs:element ref="ServiceType"/>
      <xs:element name="ProviderID" type="md:entityIDType"/>
      <xs:element name="Description" type="DescriptionType" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:group name="WsdlRef">
    <xs:sequence>
      <xs:element name="WsdlURI" type="xs:anyURI"/>
      <xs:element name="ServiceNameRef" type="xs:QName"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="BriefSoapHttpDescription">
    <xs:sequence>
      <xs:element name="Endpoint" type="xs:anyURI"/>
      <xs:element name="SoapAction" type="xs:anyURI" minOccurs="0"/>
    </xs:sequence>
  </xs:group>
  <xs:element name="ResourceOffering" type="ResourceOfferingType"/>
  <xs:complexType name="ResourceOfferingType">
    <xs:sequence>
      <xs:group ref="ResourceIDGroup"/>
      <xs:element name="ServiceInstance" type="ServiceInstanceType"/>
      <xs:element ref="Options" minOccurs="0"/>
      <xs:element name="Abstract" type="xs:string" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="entryID" type="IDType" use="optional"/>
  </xs:complexType>
  <xs:element name="Options" type="OptionsType"/>
  <xs:complexType name="OptionsType">
    <xs:sequence>
      <xs:element name="Option" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Query" type="QueryType"/>
  <xs:complexType name="QueryType">
    <xs:sequence>
      <xs:group ref="ResourceIDGroup"/>
      <xs:element name="RequestedServiceType" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="ServiceType"/>
            <xs:element ref="Options" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="optional"/>
  </xs:complexType>
  <xs:element name="QueryResponse" type="QueryResponseType"/>
  <xs:complexType name="QueryResponseType">
    <xs:sequence>
      <xs:element ref="Status"/>
      <xs:element ref="ResourceOffering" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Credentials" minOccurs="0">
        <xs:complexType>
```

```
            <xs:sequence>
              <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:complexType>
    <xs:complexType name="InsertEntryType">
      <xs:sequence>
        <xs:element ref="ResourceOffering"/>
        <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="RemoveEntryType">
      <xs:attribute name="entryID" type="IDReferenceType" use="required"/>
    </xs:complexType>

    <xs:element name="Modify" type="ModifyType"/>
    <xs:complexType name="ModifyType">
      <xs:sequence>
        <xs:group ref="ResourceIDGroup"/>
        <xs:element name="InsertEntry" type="InsertEntryType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="RemoveEntry" type="RemoveEntryType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:complexType>
    <xs:complexType name="DirectiveType">
     <xs:attribute name="descriptionIDRefs" type="xs:IDREFS" use="optional"/>
    </xs:complexType>
    <xs:element name="AuthenticateRequester" type="DirectiveType"/>
    <xs:element name="AuthorizeRequester" type="DirectiveType"/>
    <xs:element name="AuthenticateSessionContext" type="DirectiveType"/>
    <xs:element name="EncryptResourceID" type="DirectiveType"/>
    <xs:element name="ModifyResponse" type="ModifyResponseType"/>
    <xs:complexType name="ModifyResponseType">
      <xs:sequence>
        <xs:element ref="Status"/>
   <xs:element ref="Extension" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="optional"/>
      <xs:attribute name="newEntryIDs" use="optional">
        <xs:simpleType>
          <xs:list itemType="IDReferenceType"/>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
</xs:schema>
```

10. Version 1.2 Extension XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:liberty:disco:2004-04"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:md="urn:liberty:metadata:2003-08"
   xmlns:ds11="urn:liberty:disco:2004-04"
   xmlns:disco="urn:liberty:disco:2003-08"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified">

  <xs:include schemaLocation="liberty-idwsf-utility-v1.1.xsd"/>
  <xs:import namespace="urn:liberty:metadata:2003-08"
   schemaLocation="liberty-metadata-v1.1.xsd"/>
  <xs:import namespace="urn:liberty:disco:2003-08"
   schemaLocation="liberty-idwsf-disco-svc-v1.2.xsd"/>
  <xs:annotation>
    <xs:documentation>

      XML Schema from Liberty Discovery Service Specification.

      ### NOTICE ###

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

    </xs:documentation>
  </xs:annotation>
   <!-- For use in ModifyResponse Extension field -->
   <xs:element name="Keys" type="ds11:KeysType"/>
   <xs:complexType name="KeysType">
     <xs:sequence>
       <xs:element ref="md:KeyDescriptor" minOccurs="1" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:complexType>
   <xs:element name="SendSingleLogOut" type="disco:DirectiveType"/>
   <xs:element name="GenerateBearerToken" type="disco:DirectiveType"/>
</xs:schema>

```

11. WSDL

```
<?xml version="1.0"?>
<definitions name="disco-svc"
  targetNamespace="urn:liberty:disco:2003-08"
  xmlns:typens="urn:liberty:disco:2003-08"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:sb="urn:liberty:sb:2003-08"
  xmlns:disco="urn:liberty:disco:2003-08">

  <!-- Abstract WSDL for Liberty Discovery Service Specification -->

     <xsd:documentation>

       XML Schema from Liberty Discovery Service Specification.

       ### NOTICE ###

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

     </xsd:documentation>

  <types>
    <xsd:schema>
      <xsd:import schemaLocation="liberty-idwsf-disco-svc-exts-v1.2.xsd"/>
      <xsd:import schemaLocation="liberty-idwsf-soap-binding-exts-v1.2.xsd"/>
      <xsd:import schemaLocation="liberty-idwsf-soap-binding-v1.2.xsd"/>
    </xsd:schema>
  </types>

  <message name="Query">
    <part name="body" element="disco:Query"/>
  </message>

  <message name="QueryResponse">
    <part name="body" element="disco:QueryResponse"/>
  </message>

  <message name="Modify">
    <part name="body" element="disco:Modify"/>
  </message>

  <message name="ModifyResponse">
    <part name="body" element="disco:ModifyResponse"/>
  </message>

  <message name="CorrelationHeader">
    <part name="Correlation" element="typens:Correlation"/>
  </message>

  <portType name="DiscoveryPort">

    <operation name="DiscoveryLookup">
      <input message="typens:Query"/>
      <output message="typens:QueryResponse"/>
    </operation>

    <operation name="DiscoveryUpdate">
      <input message="typens:Modify"/>
      <output message="typens:ModifyResponse"/>
    </operation>

  </portType>
```

```
  <!--
  An example of a binding and service that can be used with this
  abstract service description is provided below.
  -->

  <binding name="DiscoveryBinding" type="typens:DiscoveryPort">

    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

    <operation name="DiscoveryLookup">
      <soap:operation soapAction="http://example.com/DiscoveryLookup"/>

      <input>
        <soap:header message="typens:CorrelationHeader" part="Correlation" use="literal"/>
        <soap:body use="literal"/>
      </input>

      <output>
        <soap:header message="typens:CorrelationHeader" part="Correlation" use="literal"/>
        <soap:body use="literal"/>
      </output>

    </operation>

    <operation name="DiscoveryUpdate">
      <soap:operation soapAction="http://example.com/DiscoveryUpdate"/>

      <input>
        <soap:header message="typens:CorrelationHeader" part="Correlation" use="literal"/>
        <soap:body use="literal"/>
      </input>

      <output>
        <soap:header message="typens:CorrelationHeader" part="Correlation" use="literal"/>
        <soap:body use="literal"/>
      </output>

    </operation>

  </binding>

  <service name="DiscoveryService">

    <port name="DiscoveryPort" binding="typens:DiscoveryBinding">

      <!-- Modify with the REAL SOAP endpoint -->

      <soap:address location="http://example.com/discovery"/>

    </port>

  </service>

</definitions>
```

References

Normative

[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]*.

[SAMLCore] Hallam-Baker, Phillip, Maler, Eve, eds. (05 November 2002). "SAML Core Assertions and Protocols," OASIS Standard, version 1.0, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/security/#documents*

[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/*

[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*

[LibertyMetadata] Davis, Peter, eds. "Liberty Metadata Description and Discovery Specification," Version 1.1, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*

[LibertySecMech] Ellison, Gary, eds. "Liberty ID-WSF Security Mechanisms," Version 1.2, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*

[LibertySecMech11] Ellison, Gary, eds. "Liberty ID-WSF Security Mechanisms," Version 1.1, Liberty Alliance Project (18 April 2004). *http://www.projectliberty.org/specs*

[wss-sms] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (January, 2004). "Web Services Security: SOAP Message Security," OASIS Standard V1.0 [OASIS 200401], Organization for the Advancement of Structured Information Standards *http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf*

[wss-sms-draft] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (June 30, 2003). "Web Services Security: SOAP Message Security," Draft WSS-SOAPMessageSecurity-14-06-2003, Organization for the Advancement of Structured Information Standards *http://www.oasis-open.org/committees/download.php/2757/WSS-SOAPMessageSecurity-14-063003-merged.pdf*

[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

[LibertyPAOS] Aarts, Robert, eds. "Liberty Reverse HTTP Binding for SOAP Specification," Version 1.1, Liberty Alliance Project (14 December 2004). *http://www.projectliberty.org/specs*

[LibertyClientProfiles] Aarts, Robert, eds. (14 December 2004). "Liberty ID-WSF Profiles for Liberty enabled User Agents and Devices," version 1.1, Liberty Alliance Project *http://www.projectliberty.org/specs/*