Liberty Discovery Service Specification

Version: 1.0-06

Editors:
Jonathan Sergent

Contributors:
John Beatty

Abstract:

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

Notice

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

Revision History

Revision: 03 **Date:** 21 Mar 2003

progress on the following bugs: 96: implied resource definition is missing, 102 element provider in service instance

Revision: 04 **Date:** 28 Mar 2003

Bug 113 (Editorial issues - Service Instance Description): Rearranged schema for WsdlRef and BriefSoapHttpDescription. Clarified text on when BriefSoapHttpDescription may be used instead of WsdlRef. Added references to help describe concrete WSDL versus abstract WSDL.

Bug 114 (Editorial issues - Discovery Lookup): Added useful cross-references to options. Clarify language about service instances that do not register options. Clarify entryId uniqueness constraint.

Revision: 05 **Date:** 6 Apr 2003

Worked on alignment with the security spec (also bugs 115, 117).

Bug 144 (id attribute needed...): Added id attribute for use by usage directives.

Revision: 06 **Date:** 13 Apr 2003

Changed namespace URNs from "isf" to "wsf" ; legal notice modified

Contents

1. Introduction

This draft 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.

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 (*TODO: ref*) 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]) (*TODO:* need proper reference here) 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 ds: 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 lu: stands for the Liberty (working) namespace (urn:liberty:util:1.0).
- 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. The `ServiceType` element is used to identify a service type. (*TODO:* the URIs used should be restricted to follow some general principles)

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

The `Resource` element is used to identify a particular resource. (*TODO:* the URIs used should be restricted to follow some general principles) One special value of `Resource` 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 elimiate the need for contacting the discovery service to access the resource.

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

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:element name="ServiceInstance" type="ServiceInstanceType"/>
<xs:complexType name="ServiceInstanceType">
   <xs:sequence>
 <xs:element ref="ServiceType"/>
 <xs:element name="ProviderID" type="xs:anyURI"/>
 <xs:choice>
    <xs:group ref="WsdlRef"/>
    <xs:group ref="BriefSoapHttpDescription"/>
 </xs:choice>
   </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 [5]), 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; thus, the `ServiceInstance` element contains the appropriate `ProviderID` element given the context (e.g., the identity of the anticipated requester). `ProviderID` MUST correspond with the constraints for the `entityIDType` data type as specified in [LibertyMetadata].

The necessary concrete service description (see Section 1.1 [5]) is included 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 NOT be chosen if it is not 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. (*TODO*: should this be "MUST be chosen if and only if" instead of "MUST only be chosen"? In other words, should WSDL be allowed when BriefSoapHttpDescription is sufficient?)

3.1. WsdlRef Group

`WsdlURI` provides a URI to a WSDL resource containg the service description. This must be concrete WSDL, not abstract WSDL. (TODO: add reference to definitions of these terms above.) 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:string"/>
   </xs:sequence>
```

```
</xs:group>
```

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. (*TODO*: provide example of this equivalence?)

`Endpoint` contains the URI of the SOAP-over-HTTP endpoint. The URI scheme MUST be 'http'. `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): (*TODO*: need to say more about wsdl:service)

- The `wsdl:binding` contains a `wsdlsoap:binding` element. This specifies that the SOAP binding for WSDL is being used.
- The `wsdl:binding` element provides bindings for the operations specified in the `wsdl:portType` for the abstract WSDL corresponding to the `ServiceType`.
- The `wsdlsoap:transport` attribute of the `wsdlsoap:binding` element is *http://schemas.xmlsoap.org/soap/http*.
- The `wsdlsoap:use` attribute of the `wsdlsoap:body` elements is "literal".
- The `wsdlsoap:soapAction` attribute of `wsdlsoap:operation` is equal to `SoapAction`.
- The `wsdlsoap: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>
```

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:element ref="Resource"/>
 <xs:element ref="ServiceInstance"/>
 <xs:element ref="Options" minOccurs="0"/>
 <xs:element name="Abstract" type="xs:string" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
```

The `Resource` element provides the URI for the resource that the requester can use in a request to the described `ServiceInstance`. 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.

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).

```
<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>
```

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. (TODO: expand on this and what exactly it means).

The Discovery Service is defined by the abstract WSDL in this document (TODO: insert ref to WSDL), 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, accomodating inserts and removals of resource offerings.

The Discovery Service supports "proxy" functionality whereby a resource offering can be registered for a given service type such that the resource offering proxies for all other resource offerings of the same service type. Such a proxy will communicate with one or more of the underlying resource offerings to fulfill the requests. This type of resource offering is termed a *proxy resource offering*. Registration of proxies and lookups involving service types that are being proxied for is discussed below.

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:wsf:disco:1.0*.

5.1. Operation: *DiscoveryLookup*

The DiscoveryLookup operation enables a requester to obtain an enumeration of `ResourceOffering` elements. The requester sends a `DiscoveryLookupRequest` and receives a `DiscoveryLookupResponse` 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 `DiscoveryLookupResponse`.

5.1.1. `DiscoveryLookupRequest`

The `DiscoveryLookupRequest` minimally contains the `Resource` element, which describes which discovery resource is being requested. A request with only the `Resource` 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 [10]) 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.

```
<xs:element name="DiscoveryLookupRequest" type="DiscoveryLookupRequestType"/>
<xs:complexType name="DiscoveryLookupRequestType">
   <xs:sequence>
 <xs:element ref="Resource"/>
 <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>
```

Requesters SHOULD construct a DiscoveryLookupRequest 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.

5.1.2. `DiscoveryLookupResponse`

Responses to the lookup request are wrapped in a `Results` element, which contains each response in a `Result` element. The `ResourceOffering` is contained within the `Result`. Each `Result` element MUST contain an *entryId* attribute, to be used in the *DiscoveryUpdateRequest* 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 `Result` are referenced with a set of zero of more `CredentialIdRef` elements.

```
<xs:element name="DiscoveryLookupResponse" type="DiscoveryLookupResponseType"/>
<xs:complexType name="DiscoveryLookupResponseType">
   <xs:sequence>
 <xs:element name="Results" minOccurs="0">
    <xs:complexType>
 <xs:sequence>
    <xs:element name="Result" maxOccurs="unbounded">
 <xs:complexType>
    <xs:sequence>
 <xs:element ref="ResourceOffering"/>
 <xs:element name="CredentialIdRef"
    type="xs:IDREF" minOccurs="0"
    maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="entryId" type="IDType"
 use="optional"/>
 </xs:complexType>
    </xs:element>
 </xs:sequence>
    </xs:complexType>
 </xs:element>
 <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:complexType>
```

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 provided, all entries in the discovery resource with the specified service type match.
- If the `RequestedServiceType` has a child `Options` element, a `ResourceOffering` in the discovery resource matches the query if either the intersection between the set of `Option` element values in the `RequestedServiceType` and those in the `ResourceOffering` is non-empty, or the `ResourceOffering` has no child `Options` element. (See Section 4 [10].)

If there is a proxy resource offering registered for a service type, the Discovery Service must follow these rules in determining the subset result set related to that service type:

- If the identity of the requester is not the identity of the provider of the proxy resource offering, the result set for that service type MUST contain only the proxy resource offering and all other resource offerings for which the requester is the provider.
- If the identity of the requester is the provider of the proxy resource offering, the result set MUST contain all resource offerings for the specified service type, including the proxy resource offering. Additionally, the directives for all instances of the requested service type must be aggregated when formulating the credentials, as the proxying entity will need these credentials to fulfill the request.

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 `Result` 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.

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. `DiscoveryUpdateRequest`

The `DiscoveryUpdateRequest` contains a set of zero or more `InsertEntry` elements, each containing exactly one `ResourceOffering` element.

```
<xs:element name="DiscoveryUpdateRequest" type="DiscoveryUpdateRequestType"/>
<xs:complexType name="DiscoveryUpdateRequestType">
   <xs:sequence>
 <xs:element ref="Resource"/>
 <xs:element name="InsertEntry" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
 <xs:sequence>
    <xs:element ref="ResourceOffering"/>
    <xs:any namespace="##any" processContents="lax" minOccurs="0"
 maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="proxyForServiceType" type="xs:boolean"
    use="optional" default="0"/>
    </xs:complexType>
 </xs:element>
```

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

The *proxyForServiceType* attribute specifies that the resource offering will proxy for all other registered resource offerings with the same service type. Thus, this resource offering will be a proxy resource offering.

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.

Two policy-related directives are also defined: `AuthorizeRequester` and `AuthenticateSessionContext`. If the `AuthorizeRequester` directive is specified for a resource, that the discovery service provider SHOULD include a SAML assertion containing a Resource Access Statement (as defined in [ID-WSF-Security]) in any future `DiscoveryLookupResponse` messages for the resource. (This may potentially be a Proxy Resource Access Statement, as well–but note that this is potentially a different meaning of "proxy" than that used in the "proxy resource offering" defined in this specification.) The `AuthenticateSessionContext` directive is identical to the `AuthorizeRequester` directive except that the appropriate statement is a `SessionContextStatement`. If credentials are provided in response to these directives, they MUST comply with the processing rules defined in [ID-WSF-Security].

```
<element name="AuthorizeRequester" type="EmptyType"/>
<element name="AuthenticateSessionContext" type="EmptyType"/>
```

5.2.2. `DiscoveryUpdateResponse`

The response contains only a `Status` element.

```
<xs:element name="DiscoveryUpdateResponse" type="DiscoveryUpdateResponseType"/>
<xs:complexType name="DiscoveryUpdateResponseType">
   <xs:sequence>
      <xs:element ref="Status"/>
   </xs:sequence>
   <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>
```

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.

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:wsf:disco:1.0".
- A single saml:AttributeValue element MUST be included which contains 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.

An example AttributeStatement that might be found in a Liberty ID-FF AuthnResponse is included:

```
<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:wsf:disco:1.0">
      <AttributeValue>
         <ResourceOffering xmlns="urn:liberty:wsf:disco:1.0">
            <Resource>http://example.com/disco/d0CQF8elJTDLmzEo</Resource>
            <ServiceInstance>
               <ServiceType>urn:liberty:wsf:disco:1.0</ServiceType>
               <ProviderID>http://example.com/</ProviderID>
               <Description>
                  <BriefSoapHttpDescription>
                     <Endpoint>http://soap.example.com/</Endpoint>
                     <SoapAction>urn:liberty:wsf:disco:1.0</SoapAction>
                  </BriefSoapHttpDescription>
               </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. (*TODO*: do we need to say anything about the Resource attribute on AttributeQuery? seems like it's best to avoid talking about that here.)

7. XSD

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

  <xs:include schemaLocation="lib-arch-iwsf-utility.xsd"/>

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

  <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="ServiceInstance" type="ServiceInstanceType"/>
  <xs:complexType name="ServiceInstanceType">
     <xs:sequence>
        <xs:element ref="ServiceType"/>
        <xs:element name="ProviderID" type="xs:anyURI"/>
    <xs:choice>
 <xs:group ref="WsdlRef"/>
 <xs:group ref="BriefSoapHttpDescription"/>
    </xs:choice>
     </xs:sequence>
  </xs:complexType>

  <xs:group name="WsdlRef">
     <xs:sequence>
        <xs:element name="WsdlURI" type="xs:anyURI"/>
        <xs:element name="ServiceNameRef" type="xs:string"/>
     </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:element ref="Resource"/>
        <xs:element ref="ServiceInstance"/>
        <xs:element ref="Options" minOccurs="0"/>
        <xs:element name="Abstract" type="xs:string" minOccurs="0"/>
     </xs:sequence>
  </xs:complexType>

  <xs:element name="DiscoveryLookupRequest" type="DiscoveryLookupRequestType"/>
  <xs:complexType name="DiscoveryLookupRequestType">
     <xs:sequence>
        <xs:element ref="Resource"/>
    <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="DiscoveryLookupResponse" type="DiscoveryLookupResponseType"/>
  <xs:complexType name="DiscoveryLookupResponseType">
     <xs:sequence>
```

```
         <xs:element name="Results" minOccurs="0">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Result" maxOccurs="unbounded">
                     <xs:complexType>
                        <xs:sequence>
                           <xs:element ref="ResourceOffering"/>
   <xs:element name="CredentialIdRef"
type="xs:IDREF" minOccurs="0"
maxOccurs="unbounded"/>
                        </xs:sequence>
<xs:attribute name="entryId" type="IDType"
   use="optional"/>
                     </xs:complexType>
                  </xs:element>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <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:element name="DiscoveryUpdateRequest" type="DiscoveryUpdateRequestType"/>
  <xs:complexType name="DiscoveryUpdateRequestType">
     <xs:sequence>
        <xs:element ref="Resource"/>
        <xs:element name="InsertEntry" minOccurs="0" maxOccurs="unbounded">
           <xs:complexType>
              <xs:sequence>
                 <xs:element ref="ResourceOffering"/>
<xs:any namespace="##any" processContents="lax" minOccurs="0"
   maxOccurs="unbounded"/>
              </xs:sequence>
   <xs:attribute name="proxyForServiceType" type="xs:boolean"
use="optional" default="0"/>
           </xs:complexType>
        </xs:element>
        <xs:element name="RemoveEntry" minOccurs="0" maxOccurs="unbounded">
           <xs:complexType>
              <xs:attribute name="entryId" type="IDReferenceType"/>
           </xs:complexType>
        </xs:element>
     </xs:sequence>
<xs:attribute name="id" type="xs:ID" use="optional"/>
  </xs:complexType>

  <xs:element name="AuthorizeRequester" type="EmptyType"/>
  <xs:element name="AuthenticateSessionContext" type="EmptyType"/>

  <xs:element name="DiscoveryUpdateResponse" type="DiscoveryUpdateResponseType"/>
  <xs:complexType name="DiscoveryUpdateResponseType">
     <xs:sequence>
        <xs:element ref="Status"/>
     </xs:sequence>
<xs:attribute name="id" type="xs:ID" use="optional"/>
  </xs:complexType>
</xs:schema>
```

8. WSDL

```
<?xml version="1.0"?>
<definitions name="disco-svc" targetNamespace="urn:liberty:wsf:disco:1.0:wsdl"
   xmlns:typens="urn:liberty:wsf:disco:1.0:wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:disco="urn:liberty:wsf:disco:1.0"
   xmlns:sb="urn:liberty:wsf:soap-bind:1.0">
   <!- Types for messages ->
   <types>
 <xsd:schema>
    <xsd:include schemaLocation="lib-arch-soap-bind.xsd"/>
    <xsd:include schemaLocation="lib-arch-disco-svc.xsd"/>
 </xsd:schema>
   </types>
   <!- Messages for core identity services ->
   <message name="DiscoveryLookupRequest">
 <part name="wsf-header" type="sb:Correlation"/>
 <part name="body" type="disco:DiscoveryLookupRequest"/>
   </message>
   <message name="DiscoveryLookupResponse">
 <part name="wsf-header" type="sb:Correlation"/>
 <part name="body" type="disco:DiscoveryLookupResponse"/>
   </message>
   <message name="DiscoveryUpdateRequest">
 <part name="wsf-header" type="sb:Correlation"/>
 <part name="body" type="disco:DiscoveryUpdateRequest"/>
   </message>
   <message name="DiscoveryUpdateResponse">
 <part name="wsf-header" type="sb:Correlation"/>
 <part name="body" type="disco:DiscoveryUpdateResponse"/>
   </message>
   <!- Ports for core identity services ->
   <portType name="DiscoveryPort">
 <operation name="DiscoveryLookup">
    <input message="typens:DiscoveryLookupRequest"/>
    <output message="typens:DiscoveryLookupResponse"/>
 </operation>
 <operation name="DiscoveryUpdate">
    <input message="typens:DiscoveryUpdateRequest"/>
    <output message="typens:DiscoveryUpdateResponse"/>
 </operation>
   </portType>
</definitions>
```