Liberty ID-WSF Discovery Service Specification

Version: 2.0-12

Editors:
John Beatty, Sun Microsystems, Inc.
Jonathan Sergent, Sun Microsystems, Inc.
Jeff Hodges, NeuStar, Inc.

Contributors:
Robert Aarts, Nokia Corporation
Carolina Canales-Valenzuela, Ericsson
Conor Cahill, America Online, Inc.
Darryl Champagne, IEEE-ISTO
Gary Ellison, Sun Microsystems, Inc.
Jukka Kainulainen, Nokia Corporation
John Kemp, Nokia Corporation
Paul Madsen, Entrust, Inc.
Greg Whitehead, Trustgenix, Inc.
Emily Xu, Sun Microsystems, Inc.

Abstract:
This specification defines mechanisms for describing and discovering identity web services.

Filename: draft-liberty-idwsf-disco-svc-v2.0-12.pdf
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.

Copyright © 2005 Adobe Systems; America Online, Inc.; American Express Company; Amsoft Systems Pvt Ltd.;
Avatier Corporation; Axalto; Bank of America Corporation; BIPAC; BMC Software, Inc.; Computer Associates
International, Inc.; DataPower Technology, Inc.; Diversinet Corp.; Enosis Group LLC; Entrust, Inc.; Epok, Inc.;
Ericsson; Fidelity Investments; Forum Systems, Inc.; France Télécom; French Government Agence pour le
développement de l’administration électronique (ADAE); Gamefederation; Gemplus; General Motors; Giesecke &
Devrient GmbH; GSA Office of Governmentwide Policy; Hewlett-Packard Company; IBM Corporation; Intel
Corporation; Intuit Inc.; Kantega; Kayak Interactive; MasterCard International; Mobile Telephone Networks (Pty)
Ltd; NEC Corporation; Netegrity, Inc.; NeuStar, Inc.; Nippon Telegraph and Telephone Corporation; Nokia
Corporation; Novell, Inc.; NTT DoCoMo, Inc.; OpenNetwork; Oracle Corporation; Ping Identity Corporation;
Reactivity Inc.; Royal Mail Group plc; RSA Security Inc.; SAP AG; Senforce; Sharp Laboratories of America;
Sigaba; SmartTrust; Sony Corporation; Sun Microsystems, Inc.; Supremacy Financial Corporation; Symlabs, Inc.;
Telecom Italia S.p.A.; Telefónica Móviles, S.A.; Trusted Network Technologies; Trustgenix; UTI; VeriSign, Inc.;
Vodafone Group Plc.; Wave Systems Corp. All rights reserved.
1. Introduction

This specification defines a framework for describing and discovering identity services. The conceptual model and terminology is first provided to set the context for the rest of the specification. Next, the data types for the information maintained by a Discovery Service are specified. Then the Discovery Service itself is specified.

1.1. Conceptual Model and Terminology

An identity web service is defined as a type of web service whose operations are indexed by identity. Such services maintain information about, or on behalf of, Principals — as represented by their identities — and/or perform actions on behalf of Principals. They are also sometimes referred to as simply identity services.

There are various types of identity services, each of which is assigned a unique service type identifier, encoded as a URI (Uniform Resource Identifier). This service type URI 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 [WSDLv1.1].

An example of a type of identity web service is a Principal’s "calendar service," which could be identified by a URI such as urn:example:services:calendar.

A service instance is a deployed physical instantiation of a particular type of identity service. A service provider may deploy one or more concrete service instances in the act of deploying an identity service.

A service instance may be described by a concrete WSDL document (including at least the <wsdl:binding>, <wsdl:service>, and <wsdl:port> elements) which contains the protocol endpoint and additional information necessary for a client to communicate with a particular service instance. An example of such "additional information" is communication security policy information.

A service instance is hosted by some provider, identified by a URI. An example of a service instance is a SOAP-over-HTTP endpoint offering a calendar service, being hosted by some provider.

Thus, a service instance exposes a protocol interface to a set of logical resources, nominally indexed by Principal. A resource in this specification is either data related to some Principal’s identity or a service acting on behalf of some Principal. An example of a resource is a calendar containing appointments for a particular Principal. When a client sends a request message to a service instance, information in the message serves to implicitly identify the resource being acted upon. This is accomplished in one of the following fashions:

- Implicitly (e.g. PAOS exchange [LibertyPAOS]).
- Via a <TargetIdentity> header block [LibertySOAPBinding].
- Via supplied security token: it is presumed that a resource of the security token subject, i.e. the Principal itself, is to be accessed.
- Via the endpoint. A service may choose to offer different endpoints for every resource. The simplest case of this is to represent the resource as a part of the query string.
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", or "manage", the resource). The access control policies associated with 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:

- Specifies service instance endpoint description and enumeration via a profile of W3C Web Services Addressing [WSAv1.0].
- Specifies a Discovery Service facilitating discovery and invocation of service instances.
- A SAML (see [SAMLCore2]) <Attribute> element defined such that a Endpoint Reference (EPR) for the discovery service itself can be conveyed via SAML assertions. This is known as a Discovery EPR or DS EPR and also colloquially as the discovery bootstrap.

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 [RFC2119]. These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

1.3.1. XML Namespaces

The following XML namespaces are referred to in this document:

- The prefix ds: represents the Discovery Service namespace. This namespace is the default for instance fragments, type names, and element names in this document. In schema listings, and in examples of discovery service messages and fragments thereof, this is the default namespace when no prefix is shown:
  
  urn:liberty:disco:2005-06

- The prefix saml2: stands for the SAMLv2 assertion namespace [SAMLCore2]:
  
  urn:oasis:names:tc:SAML:2.0:assertion

- The prefix samlp2: stands for the SAMLv2 protocol namespace [SAMLCore2]:
  
  urn:oasis:names:tc:SAML:2.0:protocol

- The prefix sec: stands for the Liberty Security Mechanisms namespace [LibertySecMech]:
  
  urn:liberty:sec:2005-11
The prefix `wsa:` stands for the W3C Web Services Addressing (WSA) namespace [WSAv1.0]:

```
http://www.w3.org/2002/ws/addressing
```

The prefix `wsdl:` stands for the primary WSDL v1.1 namespace [WSDLv1.1]:

```
http://schemas.xmlsoap.org/wsdl/
```

The prefix `wsdlsoap:` stands for the namespace of the WSDL-SOAP binding [WSDLv1.1]:

```
http://schemas.xmlsoap.org/wsdl/soap/
```

The prefix `xs:` stands for the W3C XML schema namespace [Schema1]:

```
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. Discovery Service Information Model

This section describes the Discovery Service information model. This model comprises the various data types, and thus information, that are maintained and managed by the Discovery Service, as well as the manner and format in which this information is exchanged between the Discovery Service and its clients.

First, there is a brief non-normative overview describing how service instances are referenced, as well as the interactions between the Discovery Service and the various other roles donned by system entities in the ID-WSF framework. Next are the normative definitions of the various elements defined in this specification and used in referencing service instances. Lastly is the Discovery Service WSA Profile, which normatively defines WSA EPRs profiled for use in referencing ID-WSF service instances.

2.1. Overview of Discovery Service Information Model

A service instance is a web service at a distinct protocol endpoint. Information about service instances needs to be communicated in various contexts. This specification defines a profile of WSA Endpoint References (EPRs) [WSAv1.0][WSAv1.0-SOAP] such that they can be used to convey service instance information needed by entities wishing to communicate with said service instances. Such profiled EPRs are termed "ID-WSF EPRs" in the remainder of this specification.

The general model for ID-WSF system entity interactions from a Principal’s perspective is as follows:

• A Principal wielding some user agent interacts with some service provider and is authenticated in some Liberty-compliant fashion, such that the service provider obtains possession of a discovery bootstrap assertion for the Principal. This assertion contains a pointer to the Principal’s discovery service instance in the form of an ID-WSF EPR.

• Now, the service provider, acting as a web service consumer (WSC) and using the ID-WSF EPR obtained above, queries the Principal’s discovery service for a pointer to some other desired service of the Principal—e.g. the Principal’s Profile Service or Calendar Service.

• The discovery service returns one or more ID-WSF EPRs to the querying WSC, pointing to the Principal’s service instance(s), of the requested type, if any.

• The WSC now employs the returned ID-WSF EPR(s) to interact with the identified service instance(s), which themselves will be acting in the role of a web service provider. The WSC returns results as appropriate to the Principal’s user agent.

Note

There are various permutations of this general interaction model. For example, the Principal’s user agent may itself act in the role of a WSC. Or, a Principal may not be actively involved in a given interaction—a WSC is simply interacting with a WSP on a Principal’s behalf. For example, it may be renewing some contract, such as a magazine subscription.

In order to enable the above Principal’s-perspective model, there is a parallel model from the web service provider’s (WSP) perspective, which is as follows:

• A service instance(s), acting as a WSP, is deployed at some addressable endpoint(s). In this example, the WSP is providing some service(s) on behalf of one or more Principals, e.g. a profile or calendar service.

• The WSP registers itself, or is somehow registered, with the Discovery Service. This is accomplished by minting ID-WSF EPRs describing the WSP’s service instances and using the appropriate Discovery Service protocol operations (defined later in this specification) to "insert" them into the Discovery Service.

• The above Principal’s-perspective model is now enabled.
Note
There are various permutations of this general WSP-perspective service instance registration model. For example, the same administrative entity may be deploying the both the Discovery Service and the other services and so may employ alternative means, e.g. bulk configuration, to effect service instance registration with their discovery service.

It is worthwhile to note that the Discovery Service as described in the models above is itself an identity service. It provides requesters enumerations of ID-WSF EPRs, each of which serves to describe a particular service instance.

The general form of an EPR is illustrated in Example 1.

Example 1. General Form of an EPR

The EPRs are profiled, as specified below in Section 2.3, by placing Liberty-specific elements in the <wsa:Metadata> elements of the EPR. These Liberty-specific elements are defined below in Section 2.2: EPR Profiling Elements. These profiled EPRs are referred to as "ID-WSF EPRs", Example 2 illustrates an ID-WSF EPR.

Note
Note that use of these profiled EPRs does not necessarily replace WSDL; rather, they may be used either in conjunction with WSDL or without. This is also described in Section 2.3.
<wsa:EndpointReference ...>
  <wsa:Address>
    http://profile-provider.com/profiles/someFoobarProfile
  </wsa:Address>
  <wsa:Metadata>
    <ds:Abstract>
      This is a personal profile containing common name information.
    </ds:Abstract>
    <ds:ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
    <ds:NotOnOrAfter>
      2005-08-15T23:18:56Z
    </ds:NotOnOrAfter>
    <ds:SecurityContext>
      <ds:SecurityMechID>
      </ds:SecurityMechID>
      <sec:Token>
        <!-- some security token goes here -->
      </sec:Token>
    </ds:SecurityContext>
    <ds:Options>
      <ds:Option>urn:liberty:id-sis-pp</ds:Option>
      <ds:Option>urn:liberty:id-sis-pp:cn</ds:Option>
      <ds:Option>urn:liberty:id-sis-pp:can</ds:Option>
      <ds:Option>urn:liberty:id-sis-pp:can:cn</ds:Option>
    </ds:Options>
  </wsa:Metadata>
</wsa:EndpointReference>

Example 2. An Instantiated ID-WSF EPR

2.2. EPR Profiling Elements

This section defines the elements that are used to profile EPRs as defined below in Section 2.3: ID-WSF Web Services Addressing EPR Profile. The full Discovery Service schema is given in Appendix A: Discovery Service Version 2.0 XSD.

All elements defined in this section feature an optional id attribute of type xs:ID. This attribute MAY be used as necessary for identifying element instances for referencing purposes.

2.2.1. Abstract — Service Instance Description

The <Abstract> element (Figure 1) is used for conveying a textual, natural language description of the service instance.

<!-- Abstract: natural-language description of service -->
<xs:element name="Abstract" type="xs:string"/>
2.2.2. Service Type

The `<ServiceType>` element (Figure 2) is used to identify a service type. This URI needs to 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.

```xml
<!-- Service Type -->
<xsd:element name="ServiceType" type="ServiceTypeType"/>
<xsd:complexType name="ServiceTypeType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:anyURI">
      <xsd:attribute name="id" type="xsd:ID" use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Some examples of possible ServiceType URIs:

- urn:liberty:disco:2005-06
- urn:liberty:id-sis-pp:2003-08
- http://example.com/my-service-wsdl-ns
- http://example.com/wsdl/my-service.wsdl

2.2.3. NotOnOrAfter

The `<NotOnOrAfter>` element states the expiration timestamp for the EPR with which it is associated (Figure 3). See Example 2, above, for an instantiated EPR example.

Values of the `<NotOnOrAfter>` element MUST be expressed in UTC form, with no time zone component.

Liberty system entities SHOULD NOT rely on time resolution finer than milliseconds. Implementations MUST NOT generate time instants that specify leap seconds.

```xml
<!-- EPR Expiration Timestamp -->
<xsd:element name="NotOnOrAfter" type="NotOnOrAfterType"/>
<xsd:complexType name="NotOnOrAfterType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:dateTime">
      <xsd:attribute name="id" type="xsd:ID" use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```
2.2.4. SecurityContext

The <SecurityContext> element (Figure 4) is a container in which <SecurityMechID> elements and <sec:Token> elements are placed and thus associated with an ID-WSF EPR. The <sec:Token> element is used to either directly contain security tokens, or to reference them [LibertySecMech].

Therefore, the <SecurityContext> element serves to denote the security context necessary for interacting with the service instance represented by the containing ID-WSF EPR.

A ID-WSF EPR MAY contain more than one <SecurityContext> element. This serves to denote mutually-exclusive groupings of <SecurityMechID>s and <sec:Token>s, and thus different security contexts.

See Section 2.3: ID-WSF Web Services Addressing EPR Profile, below, for the precise specification of the mapping of <SecurityContext>, and its contents, to ID-WSF EPRs.

<!-- Security Context Container -->
<xs:element name="SecurityContext">
<xs:complexType>
  <xs:sequence>
    <xs:element ref="SecurityMechID"
      minOccurs="1"
      maxOccurs="unbounded"/>
    <xs:element ref="sec:Token"
      minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>
</xs:element>

Figure 4. <SecurityContext> — Schema Fragment

See Example 2, above, for an instantiated ID-WSF EPR example, containing a <SecurityContext> element, itself containing <SecurityMechID> and <sec:Token> elements.

2.2.5. SecurityMechID

The <SecurityMechID> element (Figure 5) specifies the security mechanism(s) supported by the service instance represented by the ID-WSF EPR. These security mechanisms are represented as URIs, and are defined in [Liberty-SecMech].

The <SecurityMechID> element is used within the <SecurityContext> element described above. This is detailed in the Section 2.3: ID-WSF Web Services Addressing EPR Profile section below.
<!-- Security Mechanism ID -->
<xs:complexType name="SecurityMechIDType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

Figure 5. <SecurityMechID> — Schema Fragment

Some examples of possible SecurityMechID URI values (from [LibertySecMech]):

urn:liberty:security:2004-12:ClientTLS:SAMLV2
urn:liberty:security:2003-08:ClientTLS:SAML

See Example 2, above, for an instantiated ID-WSF EPR example, containing a <SecurityContext> element containing <SecurityMechID> and <sec:Token> elements.

2.2.6. WsdlURI and ServiceNameRef

The <WsdlURI> and <ServiceNameRef> elements (Figure 6) are used to reference a WSDL resource containing a service description.

<!-- For WSDL-specified protocol binding -->
<xs:element name="WsdlURI" type="WsdlURIType"/>
<xs:complexType name="WsdlURIType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

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

Figure 6. WsdlURI and ServiceNameRef — Schema Fragments

2.2.7. Options
The <Options> element (Figure 7) expresses the "options" supported by a service instance. Thus they provide hints to a potential requester whether certain data or operations may be available with a particular service instance.

For example, an option may be provided stating that home contact information is available. If no Options element is present, it means only that the service instance does not advertise whether any options are available. Options may, in fact, be employed by the service instance. 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 is empty, it means that the service instance explicitly advertises that no options are available.

The <Options> element contains zero or more <Option> elements, each of which contains a URI identifying a particular option. The set of possible URIs for an <Option> element should be defined by the service type. For example, a person profile service specification would specify a set of options particular to its own domain. However, one common <Option> flag related to security, and thus common to ID-WSF services, is defined in Section 3.5: Option Value for Response Authentication.

### 2.3. ID-WSF Web Services Addressing EPR Profile

This section specifies the profile of WSA Endpoint References (EPRs). Profiling an EPR, yielding an ID-WSF EPR, is accomplished by placing various of the elements defined in Section 2.2: EPR Profiling Elements, above, into the EPR’s <wsa:Metadata> element according to the rules defined below. All ID-WSF EPRs must adhere to the per-element rules in Section 2.2, and thereupon adhere to the rules defined in the following sections, depending upon the intended usage scenario for the ID-WSF EPR being minted.

For reference, the general form of an instantiated EPR is illustrated above in Example 1, and the <wsa:EndpointReference> schema fragment [WSAv1.0-SOAP] is illustrated below in Figure 8.

An ID-WSF EPR is normatively defined as a <wsa:EndpointReference> profiled as per this section.
Note

Except for the `<wsa:Address>` and `<wsa:ReferenceParameters>` elements, all elements discussed in the below sections are denoted as either being "absent" or "present" as content of the `<wsa:Metadata>` element of the ID-WSF EPR being minted.

```xml
<xsd:element name="EndpointReference" type="tns:EndpointReferenceType"/>
<xsd:complexType name="EndpointReferenceType">
  <xsd:sequence>
    <xsd:element name="Address" type="tns:AttributedURIType"/>
    <xsd:element name="ReferenceParameters" type="tns:ReferenceParametersType" minOccurs="0"/>
    <xsd:element ref="tns:Metadata" minOccurs="0"/>
    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
```

Figure 8. `<wsa:EndpointReference>` — Schema Fragment

### 2.3.1. ID-WSF EPR Prerequisites

The following rules MUST be observed when constructing an ID-WSF EPR per the following sections:

- The value of the `<wsa:Address>` element MUST be set as follows:

1. If the parties ultimately relying upon the ID-WSF EPR being registered are expected to do so without the use of WSDL (at runtime), then the `<wsa:Address>` element MUST contain the endpoint address of the service instance being described by this EPR. Additionally, the `<Wsd1URI>` and `<ServiceNameRef>` elements MUST NOT be present. This literally-addressed form of ID-WSF EPR is useful in order to ease the burden of WSCs from having to retrieve and parse WSDL in common cases. Additionally, the rules specified in Section 2.3.5: Literally-addressed ID-WSF EPR Specifics MUST be adhered to.

2. However, if WSCs of this service instance are expected to use WSDL to construct and send their request messages to this service instance, then the value of:

   `urn:liberty:disco:2005-06:dsepr:address:getOrComputeAddress`

   MUST be specified as the value of the `<wsa:Address>` element, and the `<Wsd1URI>` and `<ServiceNameRef>` elements MUST be present and contain appropriate values. This form of ID-WSF EPR is referred to as a WSDL-dependent ID-WSF EPR. The rules specified in Section 2.3.6: WSDL-dependent ID-WSF EPR Specifics MUST also be adhered to.

- Exactly one `<Abstract>` element MAY be present.
• Exactly one <ProviderID> element MUST be present.

• Exactly one <ServiceType> element MUST be present.

• Optionally, one or more <Options> element(s). These are discussed in detail above, in Section 2.2.7.

• Implementations, in general, MAY include a <wsa:ReferenceParameters> element, and subelements thereof, in an ID-WSF EPR. In certain situations, implementations MUST include a <wsa:ReferenceParameters> element with specific contents, as stipulated below in Section 2.3.3, and Section 2.3.4.

2.3.2. ID-WSF EPRs Minted by Web Service Providers for Registration with the Discovery Service

The following rules MUST be followed when a system entity acting in a Web Service Provider (WSP) role mints an ID-WSF EPR or ID-WSF EPRs for registration with the Discovery Service.

Note that the rules specified above in Section 2.3.1: ID-WSF EPR Prerequisites MUST also be adhered to.

1. Exactly one <NotOnOrAfter> element MAY be present. If present, it denotes the time after which the registering WSP believes any ID-WSF EPRs subsequently issued by the Discovery Service to relying parties should be considered stale.

Absence of the <NotOnOrAfter> element, or its having a value of 1970-01-01T00:00:00Z, indicates that the Discovery Service MAY subsequently mint ID-WSF EPRs referring to the registrant having a <NotOnOrAfter> element (i.e. a expiration timestamp) whose value is established by policy local to the Discovery Service.

2. One or more <SecurityContext> elements SHOULD be present. Each <SecurityContext> element MUST contain at least one <SecurityMechID> element, and SHOULD NOT contain any <sec:Token> elements. The rules specified below in Section 2.3.7: Security Mechanism Specifics MUST also be adhered to.

3. A <wsa:ReferenceParameters> element SHOULD NOT be present. If present, the element’s contents and the subsequent behavior of the Discovery Service are undefined by this or other ID-WSF specifications.

2.3.3. ID-WSF EPRs Minted by the Discovery Service for WSC Consumption

This section describes the rules that MUST be followed when a Discovery Service mints an ID-WSF EPR, or list of ID-WSF EPRs, for return to a requesting Web Service Consumer (WSC).

Note that the rules specified above in Section 2.3.1: ID-WSF EPR Prerequisites MUST also be adhered to.

1. Exactly one <NotOnOrAfter> element MAY be present in each ID-WSF EPR. If absent, or if it has a value of 1970-01-01T00:00:00Z, it means the Discovery Service is not stipulating an expiration time for this ID-WSF EPR, and that its wielder is obliged to follow its own local policy for refreshing any cached copies. If present, and if a value had been supplied at registration time, then the value at ID-WSF EPR issuance time MAY be the registration time value, or it MAY be some other value — set by the Discovery Service according to local policy, for example.

2. A <wsa:ReferenceParameters> element MAY be present in each ID-WSF EPR. When present, this element MUST contain header blocks culled from [LibertySOAPBinding], as stipulated in Section 2.3.8: wsa:ReferenceParameters, Invocation Identity, and Target Identity Specifics. It MAY contain other elements not defined in this or other ID-WSF specifications, though the cautions outlined in [WSAv1.0-SOAP] SHOULD be carefully observed.

3. One or more <SecurityContext> elements SHOULD be present in each ID-WSF EPR. If so they, and their content, MUST adhere to the rules below, as well as the additional specific rules in Section 2.3.7: Security Mechanism Specifics.
a. If no security tokens are to be embedded, then place all the supported security mechanisms, denoted by

b. Else, if a security token or tokens are to be embedded or referenced (via <sec:Token> elements), then one
MUST group corresponding <SecurityMechID> elements and corresponding <sec:Token> elements in
separate <SecurityContext> elements.

c. Security tokens MAY be embedded directly within the <sec:Token> element within a
<SecurityContext> element.

Also, a security token embedded in a <sec:Token> in a given ID-WSF EPR’s <SecurityContext>
element MAY be referenced from other <SecurityContext> elements, whether the other
<SecurityContext> elements are contained within the given ID-WSF EPR or whether they are in
another ID-WSF EPR in the list of ID-WSF EPRs being constructed.

Such referencing is accomplished by using the sec:Ref attribute of a <sec:Token> element. When
constructing such a reference, the referencing <sec:Token> MUST reference the <sec:Token> element
containing the target embedded security token, as specified in [LibertySecMech].

2.3.4. ID-WSF EPRs Minted by Web Service Consumers or Providers for SOAP

Header Block Inclusion

This section describes the rules that MUST be followed when a system entity, acting in either a WSC or WSP
role, mints an ID-WSF EPR for inclusion in a SOAP Header Block. Examples of such header blocks are the
<wsa:ReplyTo> and the <wsa:FaultTo>, as discussed in [LibertySOAPBinding]. Note that the rules specified
above in Section 2.3.1: ID-WSF EPR Prerequisites MUST also be adhered to.

1. Exactly one <NotOnOrAfter> element MAY be present. If absent, or if it has a value of 1970-01-01T00:00:00Z,
it means the ID-WSF EPR minter is not stipulating an expiration time for this ID-WSF EPR, and that the
subsequent EPR wielder is obliged to follow its own local policy for refreshing any cached copies.

2. A <wsa:ReferenceParameters> element MAY be present in each ID-WSF EPR. When present, this
element MUST contain header blocks culled from [LibertySOAPBinding], as stipulated in Section 2.3.8:
wsa:ReferenceParameters, Invocation Identity, and Target Identity Specifics. It MAY contain other elements
not defined in this or other ID-WSF specifications, though the cautions outlined in [WSAv1.0-SOAP] SHOULD
be carefully observed.

3. One or more <SecurityContext> elements MAY be present in each ID-WSF EPR. If so they, and their content,
MUST adhere to the rules below, as well as the additional specific rules in Section 2.3.7: Security Mechanism
Specifics:

a. If no security tokens are to be embedded, then place all the supported security mechanisms, denoted by

b. Else, if a security token or tokens are to be embedded or referenced (via <sec:Token> elements), then
group corresponding <SecurityMechID> elements and corresponding <sec:Token> elements in separate
<SecurityContext> elements.
c. Security tokens MAY be embedded directly within the <sec:Token> element within a <SecurityContext> element.

Also, a security token embedded in a <sec:Token> in a given ID-WSF EPR’s <SecurityContext> element MAY be referenced from other <SecurityContext> elements, whether the other <SecurityContext> elements are contained within the given ID-WSF EPR or whether they are in another ID-WSF EPR in the list of ID-WSF EPRs being constructed. Such referencing is accomplished by using the sec:Ref attribute of a <sec:Token> element. When constructing such a reference, the referencing <sec:Token> MUST reference the <sec:Token> element containing the target embedded security token, as specified in [LibertySecMech].

2.3.5. Literally-addressed ID-WSF EPR Specifics

The information contained in a literally-addressed ID-WSF EPR 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 WSDL-dependent ID-WSF EPR form (defined above, specifics are below) MUST be used instead.

The <wsa:Address> element of the ID-WSF EPR 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 <wsdl:binding> element in a WSDL-based description.

Use of this addressing form 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 a literally-addressed ID-WSF EPR):

- 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 <wsa:Address>.
- All other optional elements and attributes are not specified and thus default to the SOAP binding of WSDL.
2.3.6. WSDL-dependent ID-WSF EPR Specifics

<WsdlURI> provides a URI to a WSDL resource containing the service description. This must be concrete WSDL, not abstract WSDL (see Section 1.1: Conceptual Model and Terminology).

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 ID-WSF EPR 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>.

2.3.7. Security Mechanism Specifics

With respect to <SecurityMechID> URIs: these URIs denote the security mechanisms supported by the service instance described by the ID-WSF EPR. Other specifications, such as [LibertySecMech] define the actual security mechanisms along with their indentifying URIs. These security mechanisms refer to the way a WSC authenticates to a WSP ("peer-entity authentication") and/or provides message security ("data-origin authentication"). A ID-WSF EPR SHOULD list all of the security mechanisms that the service instance supports in order of preference. I.e. the most preferred security mechanism is first in the list, the next is the second-most preferred, and so on.

In the case that the set of supported security mechanisms varies with respect to endpoint address(es) and/or WSDL binding, the system entity constructing the ID-WSF EPRs MUST construct multiple ID-WSF EPRs with each ID-WSF EPR separately representing each supported mapping.

Also, any single <SecurityMechID> URI MUST NOT appear in more than one of the <SecurityContext> elements of any of the ID-WSF EPRs so constructed. In other words, each service instance may only specify one WSDL binding per supported security mechanism. If a sequence of ID-WSF EPRs is constructed, then the ID-WSF EPRs SHOULD appear in the order of the constructor’s preference, and the <SecurityContext> elements within each should be in order of preference, as should the <SecurityMechID> elements within them—with the most preferred item listed first in each case.

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 authenticate in some other fashion. See Example 3.

2.3.8. wsa:ReferenceParameters, Invocation Identity, and Target Identity Specifics

The <wsa:ReferenceParameters> element is used in ID-WSF EPRs to convey ID-WSF messaging-specific header blocks, as overall defined in [LibertySOAPBinding], and refined with the stipulations of this section of this specification, from the ID-WSF EPR minter to the ID-WSF EPR wielder. Typically, the former is the Discovery Service and the latter is a WSC.

At this time, there are two messaging-specific header blocks that are candidates for inclusion in <wsa:ReferenceParameters>:

• <sb:TargetIdentity>

• <sb:InvocationIdentity>
These header blocks are used to convey identity tokens, which are a means for conveying the identity of a Principal involved in an ID-WSF interaction, by means of stipulating one of the Principal’s identifiers, as well as (typically) an ID-WSF EPR denoting the Principal’s Discovery Service.

The rules for when and how the above header blocks are included in <wsa:ReferenceParameters>, when the ID-WSF EPR is minted by the Discovery Service (in response to a DiscoveryQuery operation, see Section 3.3), are as follows:

- If the Principal, whose discovery resource is being queried, is the same as the invocation identity of the Discovery-Query operation — i.e. there is not a <sb:TargetIdentity> header block on the <Query> message — then the same effective invocation identity MUST be expressed by the combination of the Discovery Service’s resultant selected security tokens (which are embedded in <Token> element(s) in the <SecurityContext> element in the ID-WSF EPR’s <wsa:Metadata> element) and possibly a <sb:InvocationIdentity> header block (per [LibertySOAPBinding]) containing an identity token (denoting the invocation identity; per [LibertySecMech] and [LibertySecMech20SAML]) included in the <wsa:ReferenceParameters> element of the ID-WSF EPR.

The Discovery Service will decide whether including the <sb:InvocationIdentity> header block is necessary based on whether the security tokens to be included are capable of expressing the invocation identity. For example, if the generated security tokens will be X.509-based, then it will be necessary to include a <sb:InvocationIdentity> header block containing an identity token, but if the generated security tokens will be SAML-based, then inclusion of a <sb:InvocationIdentity> header block is not necessary.

Note

The Discovery Service MUST NOT generate ID-WSF EPRs containing a <sb:InvocationIdentity> header block in the <wsa:ReferenceParameters> element where the generated security tokens adequately convey the invocation identity on their own. If such a conflict appears imminent, the Discovery Service MAY mint separate ID-WSF EPRs where each individual ID-WSF EPR meets this requirement.

- Else, if the Principal, whose discovery resource is being queried, is not the same as the invocation identity of the DiscoveryQuery operation — i.e. a <sb:TargetIdentity> header block appears in the header of the <Query> message — then the invocation identity to be conveyed in the ID-WSF EPR is expressed as denoted in the bullet item above, and additionally, a <sb:TargetIdentity> header block (per [LibertySOAPBinding]) containing an identity token (denoting the target identity; per [LibertySecMech] and [LibertySecMech20SAML]) is included in the <wsa:ReferenceParameters> element of the ID-WSF EPR.

The rules for when and how the above header blocks are included in <wsa:ReferenceParameters>, when the ID-WSF EPR is minted by a WSC or WSP (refer to Section 2.3.4, above, for context), are as follows:

- If the intended target identity is to be the same as that as the intended invocation identity, then the intended invocation identity MUST be expressed in the minted ID-WSF EPR as detailed in the rules above (first bullet item).

- If the intended target identity is to be different than the intended invocation identity, then the intended invocation identity and the intended target identity both MUST be expressed in the minted ID-WSF EPR as detailed in the rules above (second bullet item).
<wsa:EndpointReference entryID="_12345">
  <wsa:Address>
    http://profile-provider.com/profiles/someFoobarProfileEndpointAddr
  </wsa:Address>
  <wsa:Metadata>
    <ds:Abstract>
      This is a personal profile containing common name information.
    </ds:Abstract>
    <ds:ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
    <ds:NotOnOrAfter>
      2005-08-15T23:18:56Z
    </ds:NotOnOrAfter>
    <ds:SecurityContext id="_1">
      <ds:SecurityMechID>
        urn:liberty:security:2004-12:ClientTLS:SAMLV2
      </ds:SecurityMechID>
      <ds:SecurityMechID>
        urn:liberty:security:2005-02:ClientTLS:X509
      </ds:SecurityMechID>
      <sec:Token id="_10">
        <!-- some security token goes here -->
      </sec:Token>
    </ds:SecurityContext>
    <ds:SecurityContext id="_2">
      <ds:SecurityMechID>
        urn:liberty:security:2005-02:ClientTLS:X509
      </ds:SecurityMechID>
      <sec:Token id="_30">
        <!-- some other security token goes here -->
      </sec:Token>
    </ds:SecurityContext>
    <ds:Options>
      <ds:Option>urn:liberty:id-sis-pp</ds:Option>
      <ds:Option>urn:liberty:id-sis-pp:cn</ds:Option>
      <ds:Option>urn:liberty:id-sis-pp:can</ds:Option>
      <ds:Option>urn:liberty:id-sis-pp:can:cn</ds:Option>
    </ds:Options>
  </wsa:Metadata>
</wsa:EndpointReference>

<wsa:EndpointReference entryID="_23456">
  <wsa:Address>
    http://profile-provider.com/profiles/anotherFoobarProfileEndpointAddr
  </wsa:Address>
  <wsa:Metadata>
    <ds:Abstract>
      This is a personal profile containing common name information.
    </ds:Abstract>
    <ds:ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
    <ds:NotOnOrAfter>
      2005-08-15T23:18:56Z
    </ds:NotOnOrAfter>
    <ds:SecurityContext id="_1">
      <ds:SecurityMechID>
        urn:liberty:security:2004-12:ClientTLS:SAMLV2
      </ds:SecurityMechID>
      <ds:SecurityMechID>
        urn:liberty:security:2005-02:ClientTLS:X509
      </ds:SecurityMechID>
      <sec:Token id="_10">
        <!-- some security token goes here -->
      </sec:Token>
    </ds:SecurityContext>
    <ds:SecurityContext id="_2">
      <ds:SecurityMechID>
        urn:liberty:security:2005-02:ClientTLS:X509
      </ds:SecurityMechID>
      <sec:Token id="_30">
        <!-- some other security token goes here -->
      </sec:Token>
    </ds:SecurityContext>
    <ds:Options>
      <ds:Option>urn:liberty:id-sis-pp</ds:Option>
      <ds:Option>urn:liberty:id-sis-pp:cn</ds:Option>
      <ds:Option>urn:liberty:id-sis-pp:can</ds:Option>
      <ds:Option>urn:liberty:id-sis-pp:can:cn</ds:Option>
    </ds:Options>
  </wsa:Metadata>
</wsa:EndpointReference>
Example 3. Instantiated List of ID-WSF EPRs Illustrating Multiple <SecurityContext> Elements with both Embedded and Referenced <SecurityMechID> Elements
3. Discovery Service

A Discovery Service is an identity service facilitating requesters’ discovery of identity service instances on a per-identity basis, and acquisition of ID-WSF Endpoint References (ID-WSF EPRs) "pointing" to the discovered service instances. These ID-WSF EPRs provide requesters with the information necessary to invoke discovered service instances.

Thus in an abstract sense, the Discovery Service is essentially a web service interface to per-identity "discovery resources", each of which can be viewed as a registry of ID-WSF EPRs. The notion of "discovery resources" is an abstract way of referring to what are concretely "identity-indexed discovery service instances".

Entities can register ID-WSF EPRs, pointing to their identity services, with a discovery resource, and this will allow other entities to discover them. A common use case is that a Principal places references (aka ID-WSF EPRs) to his or her personal profile, calendar, and so on, in a discovery resource so that they may be discovered by other entities, e.g. web service providers who wish to provide the Principal with value-added services.

The act of discovering service instances is implicitly on a per-identity basis. This occurs in at least two fashions in ID-WSF:

- The first is that when a Principal authenticates to a service provider using via a SAMLv2 profile (or similarly via ID-FF), the identity provider conveys an ID-WSF EPR pointing explicitly to the Principal’s discovery service resource, which the SP may then use to discover the Principal’s various services.

- The second is similar in that if a Principal’s (LUAD-)WSC authenticates via the Single Sign-On Service (see ), then the identity provider will likely return an ID-WSF EPR for the Principal’s discovery service resource along with the Principal’s credentials, represented as security tokens.

To summarize, the discovery service is itself an identity service, and thus is identified by ID-WSF EPRs, which themselves have been crafted (typically by an identity provider) such that they identify the discovery service resource (aka discovery service instance) mapped to the Principal in question.

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 intended to be used in conjunction with other 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 could be used with other security mechanisms, not yet defined.

The Discovery Service is designed to be describable by WSDL [WSDLv1.1], and an abstract WSDL definition is included in this document, see Appendix B: Discovery Service WSDL. This WSDL document defines two "WSDL operations" for the Discovery Service. The first is the DiscoveryQuery operation. This operation returns an enumeration of ID-WSF EPRs for a given search criteria. The DiscoveryModify operation enables maintenance of a discovery resource, accommodating inserts and removals of ID-WSF EPRs.

[TODO: Need some elaboration here and also in the Implementor’s Guide w.r.t. registering principal-identifying, or principal-traceable data with the DS. E.g. one should be using "generic endpoints" rather than principal-specific ones.]

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

3.1. The Discovery Service Service Type URI
The service type URI of the Discovery Service is:

`urn:liberty:disco:2005-06`

### 3.2. Status Codes

The following status code strings are defined:

- **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 strings are expected to appear in the "code" attribute of `<Status>` elements used in SOAP-bound 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).

### 3.3. Operation: DiscoveryQuery

The *DiscoveryQuery* WSDL operation enables a requester to obtain an enumeration of ID-WSF EPRs (see Section 2: Discovery Service Information Model) — the requester sends a `<Query>` message and receives a `<QueryResponse>` message in return. Also, because a provider hosting a Discovery Service may also be playing other roles on behalf of Principals (such as a Policy Decision Point or an Authentication Authority), the *DiscoveryQuery* operation can also function as a security token service, providing the requester with an efficient means of obtaining security tokens that may be necessary to invoke service instances described in the `<QueryResponse>`.

#### 3.3.1. `<Query>` Message

A `<Query>` request message is empty in the minimal case. Such a request indicates the requester is requesting all available ID-WSF EPRs for the Principal, regardless of security mechanisms or service types. The result set is dependant upon the local access control policies of the discovery service instance.

Alternatively, a request can be qualified with a set of `<RequestedServiceType>` elements, which enables the requester to specify that all ID-WSF EPRs 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 2.2.7: Options) the returned ID-WSF EPRs should support. Note that returned ID-WSF EPRs are not guaranteed to support the requester-specified options, as some discovery service instances and/or ID-WSF EPR registrations may not support options registration.

Additionally, the requester MAY indicate which security mechanisms it is prepared to use on a per requested service type basis. If present, the discovery service compares this information with service instances’ registered security mechanism information and finds an intersection, if any. The server will return the intersection maintaining original set ordering with respect to how the service instance originally registered the security mechanism set. The client will
select from these returned values when subsequently invoking the service instance. Note that the returned set may be
distributed across multiple ID-WSF EPRs (see Section 2.3: ID-WSF Web Services Addressing EPR Profile).

[TODO: discuss <ProviderID> being present in the /Query/RequestedServiceType — if one wants/needs to obtain
all ID-WSF EPRs for given ServiceType at the Disco instance, obviating the implicit Principal context of the Disco
Query, then perhaps need an attribute or element in the Query message indicating this, since with simple presence of
the ProviderID element may usefully connote "please give me all services of this principal at this provider (modulo
any expressed ServiceTypes and/or SecurityMechIDs). See bug#726]

Requesters SHOULD construct a Query to be as qualified as possible, as the discovery service instance may have to
perform significant work for each item in the result set, especially if security tokens will be generated.

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

Figure 9. Query Message — Schema Fragment
Example 4. SOAP message containing a Query

3.3.2. QueryResponse

A `<QueryResponse>` message conveys the results of the query as a set of ID-WSF EPRs, i.e. profiled `<wsa:EndpointReference>` elements (see Section 2.3: ID-WSF Web Services Addressing EPR Profile). Each `<wsa:EndpointReference>` element MUST feature an `entryID` attribute, to be used, for example, in any subsequent Modify message. This `entryID` MUST be unique across all entries in the discovery resource being queried.

As specified in Section 2.3, security tokens, appropriate for subsequent invocation(s) of the service instances represented by the returned ID-WSF EPRs, MAY be provided within the ID-WSF EPRs in the response.

A status code is also included in the response.

Figure 10. `<QueryResponse>` — Schema Fragment
An example SOAP message containing a `<QueryResponse>` message is illustrated in Example 5. This example includes a security token embedded in the returned ID-WSF EPR. Parts of the security token have been omitted due to size.
Example 5. SOAP-bound <QueryResponse> Message with Embedded Security Token

3.3.3. DiscoveryQuery Processing Rules

The discovery service returns entries based on the requester’s criteria, the policies of the discovery resource, and the contents of the discovery resource. For each <RequestedService> element in a <Query> message, the following matching rules MUST be followed in determining the ID-WSF EPRs that will be returned to the requester:

- All ID-WSF EPR entries in the discovery resource with the specified <ServiceType> value and <SecurityMechID> value (if the latter is specified in the <RequestedService>), and lacking an <Options>, match. This is regardless whether <Options> were specified for that <RequestedService> or not.

- Also, an ID-WSF EPR in the discovery resource matches the query if each <Option> element value in the <RequestedService> element is also present in the ID-WSF EPR in the discovery resource.

Note that this means that if there are no <Option> elements in the <RequestedService> element, the ID-WSF EPR matches.

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

The discovery service MAY order <wsa:EndpointReference> 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 <wsa:EndpointReference> elements in the response but there might be some <wsa:EndpointReference> elements if an entry matching the criteria is 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.
3.4. Operation: DiscoveryModify

The DiscoveryModify operation enables a requester to insert new ID-WSF EPR entries into a discovery resource, as well as remove existing entries from a discovery resource. The DiscoveryModify 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. Entries are denoted by placing an entryID attribute on <wsa:EndpointReference> elements.

Below, the <Modify> message and attendant directives are first presented. Next, the <ModifyResponse> message is specified. Lastly, the processing rules for the overall DiscoveryModify operation are presented.

3.4.1. Modify Message

The <Modify> element contains a set of zero or more <InsertEntry> elements, each containing exactly one <wsa:EndpointReference> element, and a set of zero or more <RemoveEntry> elements, each containing an entryID attribute.

<wsa:EndpointReference> elements being inserted MUST NOT contain entryID attributes.

Note that the <InsertEntryType> definition contains an <any> element. This allows the requester to include directives about the <wsa:EndpointReference> being inserted. For example, access control policy for the ID-WSF EPR could be specified. This specification defines several standard directives, see Section 3.4.1.1: Directives.

The processing rules are given below in Section 3.4.3: DiscoveryModify Processing Rules.
An example SOAP message containing a `<Modify>` message follows. This request removes an existing ID-WSF EPR entry (see the `<QueryResponse>` example) and replaces it with a different one.
Example 6. SOAP-bound <Modify> Message

3.4.1.1. Directives

The below policy-related "directives" are defined. They are to be specified when registering ID-WSF EPRs with the Discovery Service, on an ID-WSF EPR basis.
AuthenticateRequester
AuthorizeRequester
GenerateBearerToken

Schematically, the directives are represented as elements given by the schema illustrated in Figure 12. Thus when we use the term "directive" or "directive element" in this specification, we are referring to an element of type DirectiveType.

<!-- Directives -->
<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="GenerateBearerToken" type="DirectiveType"/>

Figure 12. Directive Type — Schema Fragment

Example 6, above, illustrates the use of directives.

3.4.1.1. Overall Directive Rules

The rules in this section apply to all directives.

The directives all contain an optional descriptionIDRefs attribute (Figure 12). If the descriptionIDRefs attribute is not present in a directive element, the directive is to be taken to apply to all <SecurityContext> elements provided in the ID-WSF EPR — i.e. a profiled <wsa:EndpointReference> — that the directive is grouped with inside of an <InsertEntry> element within a <Modify> message.

If the descriptionIDRefs attribute is present in a directive element, the attribute MUST contain a list of IDREFs, each referring to a <SecurityContext> element in the <wsa:EndpointReference> the directive is associated with. Furthermore, if the descriptionIDRefs attribute is present on a directive, the directive MUST be taken to apply only to those <SecurityContext> elements explicitly referred to in the descriptionIDRefs list. This may be useful if certain directives are incompatible with certain security mechanisms.

3.4.1.1.2. <AuthenticateRequester> Directive

If the <AuthenticateRequester> directive is specified for an ID-WSF EPR’s <SecurityContext>, the discovery service SHOULD include a SAML assertion containing an <saml2:AuthnStatement> in any future <QueryResponse> messages returning that ID-WSF EPR with that <SecurityContext> (and <SecurityMechID>). This will enable the client to subsequently authenticate to the service instance represented by the ID-WSF EPR(s).
The `<AuthenticateRequester>` directive MUST be used with any ID-WSF EPRs including the security mechanisms from [LibertySecMech] which use SAML for message authentication.

The formulation of the `<AuthnStatement>` is subject to the following rules:

- If the client which sends the Query authenticates using a `<ProviderID>`, the statement SHOULD be a `<saml2:AuthnStatement>` as defined in [LibertySecMech]. In this case, the statement’s subject is the client’s ProviderID.
- Otherwise, if the client which sends the Query is a user agent which does not authenticate using a `<ProviderID>` (that is, if no `<Sender>` header block is present on the request message [LibertySOAPBinding]), the `<AuthnStatement>` included SHOULD be a `<saml2:AuthnStatement>` as defined by [SAMLCore2], and the Principal wielding the user agent SHOULD be indentified via a name identifier.

Note 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 the SAML security mechanism as specified in [LibertySecMech]. However, it might be used with the bearer security mechanism in conjunction with the GenerateBearerToken directive.

3.4.1.1.3. `<AuthorizeRequester>` Directive

If the `<AuthorizeRequester>` directive is specified for an ID-WSF EPR, the discovery service MAY, in any future `<QueryResponse>` messages returning that ID-WSF EPR, include a SAML attribute statement identifying the resource, in any SAML assertion embedded as a security token in said ID-WSF EPR (as discussed in [LibertySecMech]).

3.4.1.1.4. `<GenerateBearerToken>` Directive

The `<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>` and `<AuthorizeRequester>` directives. If the `<GenerateBearerToken>` directive is specified for an ID-WSF EPR’s `<SecurityContext>`, any SAML statements generated for the above directives MUST have the Method attribute of the `<saml2:SubjectConfirmation>` element set to:

```
urn:oasis:names:tc:SAML:2.0:cm:bearer
```

The resulting assertion is to be used as a token with the Bearer Token Authentication mechanism defined in [LibertySecMech].
The rules defined for the SAML Assertion Message Authentication mechanism do not apply to the token generated when `<GenerateBearerToken>` is present, even though it happens to be a SAML assertion.

### 3.4.2. ModifyResponse Message

The response contains the following elements and attributes. The processing rules are given below in Section 3.4.3: DiscoveryModifyProcessingRules:

- `<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 3.6).

```xml
<!-- ModifyResponse Message Element & Type -->
<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>

Figure 13. `<ModifyResponse>` — Schema Fragment

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header/>
  ...
  </soap:Header>
  <soap:Body>
    <ModifyResponse xmlns="urn:liberty:disco:2005-06" newEntryIDs="2">
      <Status code="OK"/>
    </ModifyResponse>
  </soap:Body>
</soap:Envelope>
```
3.4.3. DiscoveryModify 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 MUST enforce this atomicity. This is so that a client can easily update a ID-WSF EPR by removing the old one and inserting a new one.

- For each `<InsertEntry>` element, the Discovery Service instance MAY store the ID-WSF EPR provided such that it can be retrieved (subject to policy) by future DiscoveryQuery operations for the logical resource. If the Discovery 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 Discovery Service instance searches for a previously inserted ID-WSF EPR with the given entryID. If the entryID cannot be located, the Discovery Service instance MUST return a Failed status code for the operation. If the entryID can be located, the discovery service instance SHOULD remove the entry if policy allows. If the entry is not removed, the discovery 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.

- ID-WSF EPRs contained in `<InsertEntry>` elements SHOULD NOT contain entryID values. If an ID-WSF EPR to be inserted contains an entryID value, the discovery service instance MUST ignore the value. The discovery service instance 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. I.e. it should not be possible to correlate two discovery responses for the same user from the entryID.

- 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 Discovery Service instance 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 instance does not understand, or there are directives present that the discovery service instance 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.

3.5. Option Value for Response Authentication

The ID-WSF EPR `<SecurityContext>` element 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:

```
```
If a service instance always authenticates its response messages according to the "X.509 v3 Certificate Message Authentication" mechanism in [LibertySecMech], registrations of ID-WSF EPRs describing the service instance SHOULD include this option value. Otherwise, its registered ID-WSF EPRs 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 instance.

In case the service also supports a previous version of the security mechanism 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 [LibertySecMech], the other one according to [LibertySecMech11]. This information will aid the client in determining which version of the WSS-SMS specification ([wss-sms-draft] and/or [wss-sms]) is supported by the service, and the service will act accordingly, depending on the ID-WSF EPR used by the client. Note that this behavior 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 WSS-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 WSS-SMS spec used by that client.

### 3.6. Including Keys in the ModifyResponse Message

Some directives from the <Modify> request (see Section 3.4.1: Modify Message) may cause the Discovery Service instance to generate signed security tokens in <QueryResponse> messages for the ID-WSF EPRs in question. When the Discovery Service instance which included the directives receives the signed security tokens from a client, it needs to be able to verify the Discovery service instance’s signature on the security tokens. Typically the metadata (see [SAMLMeta2]) 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 <Keys> element extension in the <ModifyResponse> in order to provide such keys. The Discovery service instance SHOULD NOT include the <Keys> element extension unless necessary—that is, in cases where the Discovery service instance has no <ProviderID>.

The Discovery Service instance SHOULD include the <Keys> element extension in <ModifyResponse> messages if it has no <ProviderID> and the <Modify> message included an ID-WSF EPR for which the Discovery Service instance intends to generate signed security tokens.

```xml
<xs:element name="Keys" type="KeysType"/>
<xs:complexType name="KeysType">
  <xs:sequence>
    <xs:element ref="md:KeyDescriptor" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

**Figure 14. <Keys> — Schema Fragment**

The <Keys> extension element appears as a child of the <Extension> element. It contains one or more <KeyDescriptor> elements.
4. ID-WSF EPR conveyed via SAML AttributeStatement

Entities which authenticate Principals using SAML may need to discover the location of the discovery service offering a Principal’s identity services. This can be accomplished using a SAMLv2 `<saml:AttributeStatement>` [SAMLCore2]. The remainder of this section specifies how this is done.

The following figure from [SAMLCore2] illustrates the SAMLv2 AttributeType and `<saml:Attribute>` element:

```xml
<element name="Attribute" type="saml:AttributeType"/>
<complexType name="AttributeType">
  <sequence>
    <element ref="saml:AttributeValue" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="Name" type="string" use="required"/>
  <attribute name="NameFormat" type="anyURI" use="optional"/>
  <attribute name="FriendlyName" type="string" use="optional"/>
  <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
```

Figure 15. SAMLv2 Attribute Element and Type

To include an ID-WSF EPR for a Principal’s discovery service in a SAMLv2 assertion, a `<saml2:AttributeStatement>` SHOULD be included according to the following rules:

- The Name attribute of the `<saml2:Attribute>` element MUST be: `urn:liberty:disco:2003-08:DiscoveryResourceOffering`
- The NameFormat attribute of the `<saml2:Attribute>` element MUST be: `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`
- One or more `<saml2:AttributeValue>` elements MUST be included which each containing a single `<wsa:EndpointReference>` element identifying a Discovery Service instance(s). These Discovery Service instances SHOULD offer identity services for the Principal identified in the Subject element inside the `<saml2:AttributeStatement>`.
- The `<wsa:EndpointReference>` that is inside the `<saml2:AttributeStatement>` MAY contain `<SecurityContext>` element(s) in turn containing `<sec:Token>` elements containing embedded security tokens, which are necessary to access the discovery service instance(s).
An example `<saml2:AttributeStatement>` that might be found in a SAMLv2 `<samlp2:Response>` message follows. Note that it does not include any security tokens. If security tokens are needed, they should be placed in the ID-WSF EPR's `<SecurityContext>` element(s) as specified above in Section 2.3: ID-WSF Web Services Addressing EPR Profile.

```xml
<AttributeStatement xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
  <Attribute Name="urn:liberty:disco:2003-08:DiscoveryResourceOffering"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
    <AttributeValue>
      <wsa:EndpointReference>
        <wsa:Address>http://example.com/disco/d0CQF8e1JTDlmzE0</wsa:Address>
        <wsa:Metadata>
          <Abstract>The Principal's Discovery Service Resource</Abstract>
          <ServiceType>urn:liberty:disco:2005-06</ServiceType>
          <ProviderID>http://example.com/</ProviderID>
          <SecurityContext>
          </SecurityContext>
        </wsa:Metadata>
      </wsa:EndpointReference>
    </AttributeValue>
  </Attribute>
</AttributeStatement>
```

Example 8. `<AttributeStatement>` that might be found in a SAMLv2 AuthnResponse

In all cases, this `<AttributeStatement>` MUST carry a ID-WSF EPR for the Liberty discovery service defined in this specification, or earlier versions of this specification. Any other ID-WSF EPRs are to be discovered by contacting the discovery service.
5. Acknowledgments

The overall design, and original specification of, the ID-WSF Discovery Service (v1.0) was originally conceived and written by John Beatty. Jonathan Sergent subsequently took on the editorship for the v1.1 revision, making several key enhancements. I (Jeff Hodges) have edited in some modest enhancements for the 2.0 revision, notably the use of WS-Addressing Endpoint References, although much of the thinking behind them was done by Robert Aarts, Conor Cahill, John Kemp, Gary Ellison, and Greg Whitehead.

Many people have made contributions to this specification, especially those listed as contributors, through all of the spec’s revisions. Thanks to all.

The docbook source code for this revision of this specification was hand set (via XEMACS psgml package) to the tunes of Weather Report, Miles Davis, John Coltrane, Alan Holdsworth, Chick Corea, Tony Williams, John McLaughlin, Oregon, Art Blakey, Bill Bruford, and various others. Thanks also to whatever deities are responsible for the existence of coffee, dark chocolate, and fermented cereals.
References

Normative


Informative


A. Discovery Service Version 2.0 XSD

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:liberty:disco:2005-11"
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:sec="urn:liberty:sec:2005-11"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:liberty:disco:2005-11"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

<xs:include schemaLocation="liberty-idwsf-utility-v2.0.xsd"/>
<xs:import namespace="http://www.w3.org/2005/08/addressing"
  schemaLocation="ws-addr-1.0.xsd"/>
<xs:import namespace="urn:oasis:names:tc:SAML:2.0:metadata"
  schemaLocation="saml-schema-metadata-2.0.xsd"/>
<xs:import namespace="urn:liberty:sec:2005-11"
  schemaLocation="liberty-idwsf-security-mechanisms-v2.0.xsd"/>
<xs:import namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
  schemaLocation="wss-secext-1.0.xsd"/>
<xs:annotation>
  <xs:documentation>
    XML Schema from Liberty Discovery Service Specification.
  </xs:documentation>
  <xs:documentation>### NOTICE ###</xs:documentation>
  Copyright (c) 2005 Liberty Alliance participants, see
  http://www.projectliberty.org/specs/idwsf_2_0_r2_copyrights.php
</xs:annotation>

<!-- **** Discovery Service Data Elements & Types **** -->
<!-- The data elements and types in this section are used to
  embellish WS-Addressing Endpoint References (EPRs).
  They are placed in the /wsa:EndpointReference/Metadata
  element. Specific usage and cardinalities are stipulated
  in the Discovery Service v2.0 Specification. -->
<!-- Abstract: natural-language description of service -->
<xs:element name="Abstract" type="xs:string"/>
<!-- Provider ID -->
<xs:element name="ProviderID" type="ProviderIDType"/>
<xs:complexType name="ProviderIDType">
  <xs:simpleContent>
    <xs:extension base="md:entityIDType">
      <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:extension>
  </xs:complexType>
```

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

<!-- EPR Expiration Timestamp -->
<xs:element name="NotOnOrAfter" type="NotOnOrAfterType"/>
<xs:complexType name="NotOnOrAfterType">
  <xs:simpleContent>
    <xs:extension base="xs:dateTime">
      <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<!-- Security Context Container -->
<xs:element name="SecurityContext">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="SecurityMechID"
        minOccurs="1"
        maxOccurs="unbounded"/>
      <xs:element ref="sec:Token"
        minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="optional"/>
  </xs:complexType>
</xs:element>

<!-- Security Mechanism ID -->
<xs:complexType name="SecurityMechIDType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<!-- For WSDL-specified protocol binding -->
<xs:element name="WsdlURI" type="WsdlURIType"/>
<xs:complexType name="WsdlURIType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:complexType>
  <xs:element name="ServiceNameRef" type="ServiceNameRefType"/>
</xs:complexType>

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

<!-- Options -->
<xs:complexType name="OptionsType">
  <xs:sequence>
    <xs:element name="Option" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
    <xs:attribute name="id" type="xs:ID" use="optional"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>

<!-- **** Discovery Service Protocol Messages Elements & Types **** -->
<xs:complexType name="RequestedServiceType">
  <xs:sequence>
    <xs:element ref="ServiceType" minOccurs="0"/>
    <xs:element ref="ProviderID" minOccurs="0"/>
    <xs:element ref="Options" minOccurs="0"/>
    <xs:element ref="SecurityMechID" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="QueryType">
  <xs:sequence>
    <xs:element name="RequestedService" type="RequestedServiceType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>

<xs:complexType name="QueryResponseType">
  <xs:sequence>
    <xs:element name="RequestQuery" type="QueryType"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>

<!-- Query Message Element & Type -->
<xs:element name="Query" type="QueryType"/>

<!-- QueryResponse Message Element & Type -->
<xs:element name="QueryResponse" type="QueryResponseType"/>
<xs:sequence>
  <xs:element ref="Status"/>
  <xs:element ref="wsa:EndpointReference"
    minOccurs="0"
    maxOccurs="unbounded"/>
</xs:sequence>

<xs:attribute name="id"
  type="xs:ID"
  use="optional"/>
</xs:complexType>

<!-- Modify Message Element & Types -->
<xs:element name="Modify" type="ModifyType"/>
<xs:complexType name="ModifyType">
  <xs:sequence>
    <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="InsertEntryType">
  <xs:sequence>
    <xs:element ref="wsa:EndpointReference"/>
    <!-- the below is used to include directives -->
    <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>

<!-- Directives -->
<xs:element name="AuthenticateRequester" type="DirectiveType"/>
<xs:element name="AuthorizeRequester" type="DirectiveType"/>
<xs:element name="GenerateBearerToken" type="DirectiveType"/>

<!-- ModifyResponse Message Element & Type -->
<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>

<!-- Keys Element - For use in ModifyResponse Extension field -->
<xs:element name="Keys" type="KeysType"/>
<xs:complexType name="KeysType">
  <xs:sequence>
    <xs:element ref="md:KeyDescriptor" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
B. Discovery Service WSDL

<?xml version="1.0"?>
<definitions name="disco-svc"
targetNamespace="urn:liberty:disco:2005-06"
xmlns:typens="urn:liberty:disco:2005-06"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmllns:xsd="http://www.w3.org/2001/XMLSchema"
xmllns:soap="http://schemas.xmlsoap.org/wsdl/soap/
xmlns:disco="urn:liberty:disco:2005-06">

<!-- Abstract WSDL for Liberty Discovery Service v2.0 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>
	<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="DiscoveryQuery">
	<input message="typens:Query"/>
	<output message="typens:QueryResponse"/>
	</operation>
	<operation name="DiscoveryModify">
	<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.

<!--

<!--

An example of a binding and service that can be used with this abstract service description is provided below.

-->