Abstract:

The Liberty Identity Web Services Framework (ID-WSF) allows the creation of identity services - web service interfaces exposed on behalf of a user.

There exist many circumstances where a user wishes to access the identity service of another user. Some examples include: a parent wishing to discover the current location of a child, someone wishing to share photographs stored at some service with their friends, or allowing one game-player to determine whether another player is available.

In such cases, it is necessary for one user to be able to obtain an identifier for another user from that user’s Identity Provider, and to convey that identifier to this second user’s identity services.

Additionally, users will often desire to grant access rights to their browser-based resources (e.g. online photos) to friends and colleagues- this implies that the privileges can be assigned to a relevant identifier for that friend as known by an appropriate identity provider.

This document describes an architecture for allowing secure, privacy-respecting access by one user to another’s identity information (both browser-based and programmatic services), and normatively defines the Liberty ID-WSF People Service, one component of such an architecture.

Filename: draft-liberty-idwsf-people-service-v1.0-08.pdf
1 Contents

2 1. Introduction ................................................................. 4
3  1.1. Overview ............................................................... 4
4  1.2. Notation ................................................................. 4
5  1.3. Terminology ............................................................. 5
6  1.4. Namespaces ............................................................... 5
7  2. Data Model ................................................................. 6
8  2.1. <Object> Element ....................................................... 6
9  2.1.1. NodeType Attribute ............................................... 6
10  2.1.2. CreatedDateTime Attribute ..................................... 6
11  2.1.3. ModifiedDateTime Attribute ..................................... 6
12  2.2. <ObjectID> Element .................................................. 7
13  2.3. <DisplayName> Element ............................................. 7
14  2.4. <Tag> Element .......................................................... 7
15  2.5. <ObjectRef> Element ................................................ 7
16  2.6. <Token> Element ....................................................... 8
17  3. People Service ........................................................... 9
18  3.1. Overview ............................................................... 9
19  3.2. Service Type .......................................................... 9
20  3.3. Request and Response Abstract Types ............................. 9
21  3.3.1. Complex Type RequestAbstractType .......................... 9
22  3.3.2. Complex Type ResponseAbstractType .......................... 9
23  3.4. Status ................................................................. 10
24  3.5. Identity Token Policy ................................................ 11
25  3.6. Success & Failure ..................................................... 11
26  3.7. Subscription and Notification ...................................... 11
27  3.7.1. <Subscription> Element ......................................... 12
28  3.7.2. Notify and NotifyResponse Messages ......................... 12
29  3.7.3. <Notification> Element ...................................... 12
30  3.8. Adding an Entity ..................................................... 13
31  3.8.1. AddEntityRequest Message ..................................... 13
32  3.8.2. AddEntityResponse Message .................................... 14
33  3.8.3. Processing Rules .................................................. 15
34  3.9. Adding a Known Entity ............................................... 16
35  3.9.1. AddKnownEntityRequest Message ............................... 16
36  3.9.2. AddKnownEntityResponse Message ............................. 17
37  3.9.3. Processing Rules .................................................. 18
38  3.10. Removing an Entity ................................................ 19
39  3.10.1. RemoveUser Message .......................................... 19
40  3.10.2. RemoveEntityResponse Message .............................. 20
41  3.10.3. Processing Rules ................................................ 21
42  3.11. Adding a Collection ............................................... 21
43  3.11.1. AddCollectionRequest Message ............................... 21
44  3.11.2. AddCollectionResponse Message ............................. 22
45  3.11.3. Processing Rules ................................................ 23
46  3.12. Removing a Collection ............................................. 23
47  3.12.1. RemoveCollectionRequest Message ............................ 23
48  3.12.2. RemoveCollectionResponse Message .......................... 24
49  3.12.3. Processing Rules ................................................ 24
50  3.13. Adding to a Collection ............................................. 25
51  3.13.1. AddToCollectionRequest Message .............................. 25
52  3.13.2. AddToCollectionResponse Message ............................ 25
53  3.13.3. Processing Rules ................................................ 26
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.14.</td>
<td>Removing from a Collection</td>
<td>26</td>
</tr>
<tr>
<td>3.14.1.</td>
<td>RemoveFromCollectionRequest Message</td>
<td>26</td>
</tr>
<tr>
<td>3.14.2.</td>
<td>RemoveFromCollectionResponse Message</td>
<td>27</td>
</tr>
<tr>
<td>3.14.3.</td>
<td>Processing Rules</td>
<td>27</td>
</tr>
<tr>
<td>3.15.</td>
<td>Listing Members</td>
<td>28</td>
</tr>
<tr>
<td>3.15.1.</td>
<td>ListMembersRequest Message</td>
<td>28</td>
</tr>
<tr>
<td>3.15.2.</td>
<td>ListMembersResponse Message</td>
<td>29</td>
</tr>
<tr>
<td>3.15.3.</td>
<td>Examples</td>
<td>30</td>
</tr>
<tr>
<td>3.15.4.</td>
<td>Processing Rules</td>
<td>31</td>
</tr>
<tr>
<td>3.16.</td>
<td>Retrieving Info</td>
<td>32</td>
</tr>
<tr>
<td>3.16.1.</td>
<td>GetObjectInfoRequest Message</td>
<td>32</td>
</tr>
<tr>
<td>3.16.2.</td>
<td>GetObjectInfoResponse Message</td>
<td>33</td>
</tr>
<tr>
<td>3.16.3.</td>
<td>Processing Rules</td>
<td>33</td>
</tr>
<tr>
<td>3.17.</td>
<td>Updating Info</td>
<td>34</td>
</tr>
<tr>
<td>3.17.1.</td>
<td>SetObjectInfoRequest Message</td>
<td>34</td>
</tr>
<tr>
<td>3.17.2.</td>
<td>SetObjectInfoResponse Message</td>
<td>35</td>
</tr>
<tr>
<td>3.17.3.</td>
<td>Processing Rules</td>
<td>35</td>
</tr>
<tr>
<td>3.18.</td>
<td>Querying Objects</td>
<td>35</td>
</tr>
<tr>
<td>3.18.1.</td>
<td>QueryObjectsRequest Message</td>
<td>35</td>
</tr>
<tr>
<td>3.18.2.</td>
<td>QueryObjectsResponse Message</td>
<td>36</td>
</tr>
<tr>
<td>3.18.3.</td>
<td>Processing Rules</td>
<td>37</td>
</tr>
<tr>
<td>3.19.</td>
<td>Testing Membership</td>
<td>38</td>
</tr>
<tr>
<td>3.19.1.</td>
<td>TestMembershipRequest Message</td>
<td>38</td>
</tr>
<tr>
<td>3.19.2.</td>
<td>TestMembershipResponse Message</td>
<td>39</td>
</tr>
<tr>
<td>3.19.3.</td>
<td>Processing Rules</td>
<td>40</td>
</tr>
<tr>
<td>3.20.</td>
<td>Resolving Objects</td>
<td>40</td>
</tr>
<tr>
<td>3.20.1.</td>
<td>ResolveIdentifierRequest Message</td>
<td>40</td>
</tr>
<tr>
<td>3.20.2.</td>
<td>ResolveIdentifierResponse Message</td>
<td>41</td>
</tr>
<tr>
<td>3.20.3.</td>
<td>Processing Rules</td>
<td>41</td>
</tr>
<tr>
<td>4.</td>
<td>Interaction with Users</td>
<td>43</td>
</tr>
<tr>
<td>4.1.</td>
<td>Model (Informative)</td>
<td>43</td>
</tr>
<tr>
<td>4.2.</td>
<td>Processing Rules</td>
<td>43</td>
</tr>
<tr>
<td>5.</td>
<td>Sequence Examples</td>
<td>44</td>
</tr>
<tr>
<td>5.1.</td>
<td>Policy definition</td>
<td>44</td>
</tr>
<tr>
<td>5.2.</td>
<td>AccessControl</td>
<td>44</td>
</tr>
<tr>
<td>5.3.</td>
<td>Group Operation</td>
<td>46</td>
</tr>
<tr>
<td>6.</td>
<td>Security Considerations</td>
<td>49</td>
</tr>
<tr>
<td>7.</td>
<td>XML Schema for ID-WSF People Service</td>
<td>50</td>
</tr>
<tr>
<td>8.</td>
<td>Abstract WSDL</td>
<td>58</td>
</tr>
<tr>
<td>9.</td>
<td>References</td>
<td>62</td>
</tr>
</tbody>
</table>
1. Introduction

1.1. Overview

A user’s People Service (PS) is an interface into those other users with which the owning user wishes to (or has already) interact with in some online fashion - these other users possibly categorized into arbitrary groups. The PS provides a flexible, privacy respecting framework by which a user can manage/track the people they know and how these other users are related.

The first generation of online transactions/interactions were single-user, eg. online banking, travel booking, shopping etc. More and more however, our online interactions involve other users than just ourselves. Whether it is communication, commerce, sharing, self-expression, or collaboration being enabled - all these interactions build on a social layer that connects individuals to others. Unfortunately, the current situation is that each of these applications generally builds its view of a given individual’s complete social network. This can result in duplication and undesirable management burden on those individuals, forced to maintain these multiple views.

Many interesting interactions will involve those individuals who are both explicit and direct. For instance, a user may wish to share their online photos with their family, or they may need to determine the network presence of their colleagues.

Enabling such direct interactions between users and their circle of friends is straightforward when both maintain an account at the same provider. On many online photo sites for instance, users share their photos with others but only once they have established an account at the same provider. If the first user already knows the account name of the other, all that need happen is for that name to be supplied. If they don’t know it, they might search existing accounts or, if necessary, have an invite sent to their friend encouraging them to create an account.

There are two significant implications of this model:

1. Both users must maintain or establish accounts at the same provider. Typically, the result of this requirement is that the friend being invited to interact (e.g. View vacation photos, etc) is forced to create an account (with associated logins and passwords to remember) at a provider where they might not otherwise choose to do so.

2. If some connection between two friends is established in the context of the photo site, it can’t be leveraged in some other context (e.g. Calendar sharing) unless that provider happens to host both services.

Enabling such cross-user interactions such that the above two implications are addressed is the goal of the Liberty Alliance’s People Service. The People Service provides a flexible, privacy respecting framework by which one user can manage/track the people they know - typically but not exclusively in order to assign them certain privileges for accessing certain resources owned by the first user. Providers query/manipulate this information through standardized interfaces.

Additionally, to satisfy the requirement for informing a user of another’s intent to add them to their PS resource, an invitation model by which user’s can be informed of such and establish the necessary federations between providers is defined.

This document is the Liberty Identity Web Services Framework (ID-WSF) People Service Specification that normatively specifies the People Service.

1.2. Notation

This specification uses schema documents conforming to W3C XML Schema (see [Schema]) and normative text to describe the syntax and semantics of XML-encoded protocol messages. Note: Phrases and numbers in brackets [ ] refer to other documents; details of these references may be found at the end of this document.

The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," "MAY NOT," and "OPTIONAL." in this specification are to be interpreted as described...
in [RFC2119]: "they MUST only be used where it is actually required for interoperability or to limit behavior which has potential for causing harm (e.g., limiting retransmissions)."

These keywords are thus capitalized when used to specify, unambiguously, 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.

This specification uses the following typographical conventions in text: <Element>, <ns:ForeignElement>, attribute, Datatype, and OtherCode.

Definitions for Liberty-specific terms may be found in [LibertyGlossary].

### 1.3. Terminology

The Liberty terms Service Provider and Web Service Consumer, and their respective abbreviations, SP and WSC, refer to different roles that may be assumed by the same website. Generally, an SP is some website that provides online services to users through HTTP interactions. In interactions with other providers not mediated by a user’s browser, websites assume the role of a WSC in order to send SOAP-based requests. For clarity, this specification uses the SP abbreviation to refer to both these roles, distinguishing where appropriate.

### 1.4. Namespaces

The following namespaces are used in the schema definitions:

  This namespace is the default for instance fragments, type names, and element names in this document.

- The prefix xs: stands for the W3C XML schema namespace (http://www.w3.org/2001/XMLSchema) [Schema1].

- The prefix xml: stands for the W3C XML namespace (http://www.w3.org/XML/1998/namespace) [XML].

- The prefix saml: stands for the OASIS SSTC SAML2.0 Assertion namespace (urn:oasis:names:tc:SAML:2.0:assertion) [SAMLCore2].

- The prefix samlp: stands for the OASIS SSTC SAML2.0 Protocol namespace (urn:oasis:names:tc:SAML:2.0:protocol) [SAMLCore2].

- The prefix tns: stands for the W3C Web Services Addressing (WS-Addressing) 1.0 namespace (http://www.w3.org/2005/08/addressing) [WSAv1.0].


2. Data Model

A given user’s PS holds information about those other users with which the owning user may have established some online relationship. The owning user may also choose to organize these other users into groups (e.g. their teammates on a hockey team). The PS data model defines how these users and groups are represented.

2.1. <Object> Element

Both individual users and the groups to which they may belong are represented as <Object> elements - whether an <Object> refers to a group or a user (or perhaps some other individual entity) is distinguished by a NodeType attribute with values of urn:liberty:ps:collection or urn:liberty:ps:entity respectively (see Section 2.1.1 for exact definition).

The <Object> element has <DisplayName> elements to carry a human-readable name for the <Object> (see Section 2.3).

The <ObjectID> element uniquely labels each <Object> (see Section 2.2).

The optional CreatedDateTime and ModifiedDateTime attributes express the time at which an Object was created and last modified respectively (see Section 2.1.2).

To account for nested Objects, an <Object> element can have multiple <Object> and/or <ObjectRef> elements to refer to other Objects.

The schema model for the <Object> element is shown below.

```
<xs:element name="Object" type="ObjectType"/>
<xs:complexType name="ObjectType">
  <xs:sequence>
    <xs:element ref="ObjectID" minOccurs="0"/>
    <xs:element name="DisplayName" type="LocalizedDisplayNametype" minOccurs="1" maxOccurs="unbounded"/>
    <xs:element name="Tag" type="TagType" minOccurs="0"/>
    <xs:element ref="Object" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="ObjectRef" type="ObjectIDType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="NodeType" type="xs:anyURI" use="required"/>
  <xs:attribute name="CreatedDateTime" type="xs:dateTime" use="optional"/>
  <xs:attribute name="ModifiedDateTime" type="xs:dateTime" use="optional"/>
</xs:complexType>
```

2.1.1. NodeType Attribute

The NodeType attribute is defined such that the WSC can distinguish if an <Object> refers to a group or a user (or some other individual entity). For the values of the NodeType attribute, the following two URI’s are defined:

- urn:liberty:ps:collection: If an <Object> has this URI for the value of the NodeType attribute, it represents a collection that has zero or more <Object> as child elements. The child <Object> elements may have a NodeType of either urn:liberty:ps:collection or urn:liberty:ps:entity.  

- urn:liberty:ps:entity: If an <Object> has this URI for the value of the NodeType attribute, it represents a single entity (e.g. a user). An <Object> with a NodeType of urn:liberty:ps:entity MUST NOT itself contain any child <Object> elements.
2.1.2. CreatedDateTime Attribute

The CreatedDateTime attribute may be used by a PS provider to set the time when an <Object> is instantiated.

2.1.3. ModifiedDateTime Attribute

The ModifiedDateTime attribute may be used by a PS provider to set the time when the data or attributes that an <Object> has are changed.

2.2. <ObjectID> Element

The <ObjectID> element is defined so that the PS provider can scope each member’s identifier locally and uniquely.

```xml
<!-- Declaration of ObjectID element -->
<xs:element name="ObjectID" type="ObjectIDType"/>
<!-- Definition of ObjectIDType -->
<xs:complexType name="ObjectIDType">
    <xs:simpleContent>
        <xs:restriction base="xs:anyURI"/>
    </xs:simpleContent>
</xs:complexType>
```

Where privacy is a concern, PS providers MUST ensure that ObjectID’s do not create a privacy concern by allowing different WSCs to make inappropriate correlations about the users for which the Object identifiers stand. Unique identifiers for different WSCs (e.g. pairwise identifiers) and encrypted identifiers are potential mechanisms for addressing this concern.

2.3. <DisplayName> Element

The <DisplayName> element provides a human-readable friendly name for Objects. The value of this element SHOULD NOT be used to uniquely identify Objects; rather the ObjectID element SHOULD be used. (see Section 2.2).

```xml
<xs:complexType name="LocalizedDisplayNameType">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="Locale" type="xs:language" use="required"/>
            <xs:attribute name="IsDefault" type="boolean" use="optional"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

2.4. <Tag> Element

The <Tag> element allows users to add their own metadata to <Object> elements. For instance, a user might add a <Tag> element with a value of ‘sports’ for the Ref attribute to a group Object called 'Team' to denote the theme of that group (perhaps to distinguish it from another group with the same name for some work project).

```xml
<xs:complexType name="TagType">
    <xs:attribute name="Ref" type="xs:anyURI" use="required"/>
</xs:complexType>
```

The value of the Ref attribute SHOULD be a tag space (a place that collates or defines tags), where the last component of the URL is the tag. For instance, http://technorati.com/tag/music is a URL for the tag "music."
2.5. **<ObjectRef> Element**

The `<ObjectRef>` element is used as a pointer to an `<Object>` through that `<Object>`’s `<ObjectID>` element.

The `<ObjectRef>` element allows an `<Object>` element to be included in another by reference rather than directly. For instance, the fact that a given user belongs to different groups can be represented by both those groups’ `<Object>` element containing an `<ObjectRef>` element that refers to that user’s `<Object>` element.

2.6. **<Token> Element**

The `<sec:Token>` element acts as a container for identity tokens, see [LibertySecMech] page 7.

The `<sec:Token>` element is used by the PS provider to return requested identity tokens to the WSC, either in a `<ResolveIdentifierResponse>` message or in a `<Notify>` message to a previous `<Subscription>`.
3. People Service

3.1. Overview

A People Service is an ID-WSF identity web service by which service consumers can query the list of entities (e.g. friends, co-workers, family, devices etc) with which a particular individual chooses to track an online relationship. These listed individuals may be organized into groups. Service consumers use the People Service to add members and/or groups, update information for particular members or groups, test group membership of a particular user, and obtain identity tokens for desired members.

3.2. Service Type

A People Service is identified by the service type URN:

urn:liberty:ps:2005-11

3.3. Request and Response Abstract Types

3.3.1. Complex Type RequestAbstractType

All PS request messages are of types that are derived from the abstract RequestAbstractType complex type. This type defines common attributes that are associated with all PS requests:

- **id [Required]**: An identifier for the request. It is of type xs:ID. It should be noted that XML processors, such as XML Signature verifiers, must be aware of the xs:ID type of these ID attributes in order resolve references to the elements they identify. If the W3C xml:id recommendation is finalized before this specification goes final, all ID attributes defined in this specification will be changed to xml:id. This change will allow XML processors to resolve references to elements defined in this specification without requiring specific knowledge about the schema defined in this specification.

- **TimeStamp [Required]**: The time instant of issue of the request. It is of type xs:dateTime and MUST be expressed in UTC form. Senders and receivers SHOULD NOT rely on other applications supporting time resolution finer than milliseconds. Implementations MUST NOT generate time instants that specify leap seconds.

The following schema fragment defines the XML the RequestAbstractType complex type:

```
<!-- Definition of RequestAbstractType -->
<xs:complexType name="RequestAbstractType" abstract="true">
  <xs:attribute name="id" type="xs:ID" use="required"/>
  <xs:attribute name="TimeStamp" type="xs:dateTime" use="required"/>
</xs:complexType>
```

3.3.2. Complex Type ResponseAbstractType

All PS response messages are of types that are derived from the abstract ResponseAbstractType complex type. This type defines common attributes and elements that are associated with all PS responses:
id [Required] An identifier for the request. It is of type xs:ID. It should be noted that XML processors, such as XML Signature verifiers, must be aware of the xs:ID type of these ID attributes in order resolve references to the elements they identify. If the W3C xml:id recommendation is finalized before this specification goes final, all ID attributes defined in this specification will be changed to xml:id. This change will allow XML processors to resolve references to elements defined in this specification without requiring specific knowledge about the schema defined in this specification.

TimeStamp [Required] The time instant of issue of the request. It is of type xs:dateTime and MUST be expressed in UTC form. Senders and receivers SHOULD NOT rely on other applications supporting time resolution finer than milliseconds. Implementations MUST NOT generate time instants that specify leap seconds.

<Status> [Required] The <Status> element is used to convey status codes and related information. The schema fragment is defined in the Liberty ID-WSF Utility schema. The local definition of status codes are described in Section 3.4.

The following schema fragment defines the XML the ResponseAbstractType complex type:

```xml
<xs:complexType name="ResponseAbstractType" abstract="true">
  <xs:sequence>
    <xs:element ref="Status"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="required"/>
  <xs:attribute name="TimeStamp" type="xs:dateTime" use="required"/>
</xs:complexType>
```

### 3.4. Status

All the response messages extended from ResponseAbstractType contain a <Status> element (see Section 3.3.2) to indicate whether or not the processing of the request message has succeeded. The <Status> element is included from the Liberty ID-WSF Utility Schema. A <Status> element MAY contain other <Status> elements providing more detailed information. A <Status> element has a code attribute, which contains the return status as a string. The local definition of these codes is specified in this document. This specification defines the following status codes to be used as values for the code attribute:

- CannotFindIDP
- CannotFindObject
- CannotResolveToken
- Failed
- InvalidNodeType
- InvalidObjectID
- ResolveIdentifierNotSupported
- ObjectIsCollection
- ObjectIsEntity
341  • OK
342  • NoResults
343  • PolicyDoesNotAllow
344  • Timeout
345  • UnexpectedError
346  • UnrecognizedFilter
347  • UnrecognizedNamespace
348  • UnspecifiedError

349  The <Status> element may contain other <Status> elements supplying more detailed return status information.
350  The code attribute of the top level <Status> element MUST contain either OK or Failed. The remainder of the values
351  above are used to indicate more detailed return status inside second level <Status> element(s).

352  OK    The value OK means that the processing of the request message has succeeded. A second level status
353    code MAY be used to indicate some special cases, but the processing of the request message has
354    succeeded.
355  Failed The value Failed means that the processing of the request message has failed. A second level status
356    code SHOULD be used to indicate the reason for the failure.

3.5. Identity Token Policy

358  For those messages that may result in an identity token being returned (either directly or not) to the WSC, that WSC
359  may wish to indicate its requirements of the identity token. For instance, the WSC may wish that the returned identity
360  token should carry a long-lived federated identifier for the user in question. Alternatively, should its immediate
361  requirements not justify the establishment of such a federated identifier (and the potential associated management
362  burden) it may desire only a short-lived and transient identifier.

363  The <TokenPolicy> element serves as a container for such WSC policy requirements. This specification defines the
364  use of the SAML <NameIDPolicy> element within the <TokenPolicy> element but the <TokenPolicy> element
365  is defined to be extensible to accommodate other aspects of token policy.

366  The schema declaration for the <TokenPolicy> element is shown below.

367 368  <xs:element name="TokenPolicy" type="TokenPolicyType"/>
369  <xs:complexType name="TokenPolicyType">
370     <xs:sequence>
371         <xs:element ref="samlp:NameIDPolicy"/>
372         <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
373     </xs:sequence>
374  </xs:complexType>
375

376  If no <TokenPolicy> element is present, or if there is no <NameIDPolicy> element within a <TokenPolicy>
377  element element, the default identity token policy is that the WSC desires a SAML assertion with a name identifier

379  If the WSC desires an alternative identity token, it MUST specify this accordingly.

3.6. Success & Failure
For those protocol messages that support multiple operations to be requested in a single message (e.g. removing multiple users from a targeted group in one step), all operations succeed or fail together.

### 3.7. Subscription and Notification

When present in a PS request message, a `<Subscription>` element indicates that the WSC wishes to be notified if and when the data associated with the targeted `Object` changes.

For each request message for which a `<Subscription>` element is allowed, this specification defines the `Object` for which changes are being subscribed to and the data that the PS provider will return in any `<Notify>` message.

The subscription & notification model is a constrained version of the more flexible model defined in [LibertyDST](#).

#### 3.7.1. `<Subscription>` Element

The schema declaration for the `<Subscription>` element is shown below. The semantics of the attributes are defined in [LibertyDST](#).

```xml
<xs:element name="Subscription">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="NotifyToRef" use="optional" type="xs:anyURI"/>
    <xs:attribute name="AdminNotifyToRef" use="optional" type="xs:anyURI"/>
    <xs:attribute name="Starts" use="optional" type="xs:dateTime"/>
    <xs:attribute name="Expires" use="optional" type="xs:dateTime"/>
    <xs:attribute name="id" use="optional" type="xs:ID"/>
    <xs:attribute name="SubscriptionID" use="required" type="IDType"/>
    <xs:attribute name="IncludeData" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="Yes"/>
          <xs:enumeration value="No"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

#### 3.7.2. Notify and NotifyResponse Messages

If and when the criteria specified by a `<Subscription>` element are met, the PS uses a `<Notify>` message to indicate this to the WSC.

The schema declarations for the `<Notify>` and `<NotifyResponse>` messages are shown below:

```xml
<xs:element name="Notify">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="RequestAbstractType">
        <xs:sequence>
          <xs:element ref="Notification"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```
3.7.3. <Notification> Element

The <Notification> element carries any changed data. The value of the SubscriptionID attribute on the <Notification> element MUST match that of the SubscriptionID attribute on the <Subscription> element for which the <Notification> is being sent.

If the WSC subscribed only to be notified of an event occurring, but did not request the actual changed data to be returned, the PS returns an empty <Notification> element.

The schema declaration for the <Notification> element is shown below. The semantics of the elements are defined in [LibertyDST].

```xml
<xs:element name="Notification">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Object" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="Token" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="Expires" use="optional" type="xs:dateTime"/>
    <xs:attribute name="id" use="optional" type="xs:ID"/>
    <xs:attribute name="SubscriptionID" use="required" type="IDType"/>
  </xs:complexType>
</xs:element>
```

3.8. Adding an Entity

A WSC indicates to the PS provider that it wishes a user Object to be created by sending an <AddEntityRequest> message. The Object being created MUST be a urn:liberty:ps:entity Object.

3.8.1. AddEntityRequest Message

A WSC uses the <AddEntityRequest> message to request that a specified <Object> be created.

The <AddEntityRequest> MUST NOT be used to add a new Object to an existing Object, nor to create two nested Objects.

The presence of a <Subscription> element indicates to the PS provider that the WSC desires that the PS provider return to it (when later possible) an identity token for the invited user within a <Notify> message - this possible after a federation has been established between the PS provider and the appropriate IDP. If no <Subscription> element is present, the WSC is indicating that the PS provider need not return an identity token through this mechanism.

A PS provider can also use the <AddEntityRequest> message to request that an Object be added to a PS list. Typically, this will happen to ensure bilateral PS lists, e.g. if a user is added to a friend’s PS, then the friend will be added to the user’s PS.

The <AddEntityRequest> message has the complex type AddEntityRequestType, which extends RequestAbstractType and adds the following elements:

- **<Object> [Required]** The <Object> element is used to convey the target user Object being added.
- **<SPRedirectURL> [Optional]** The <SPRedirectURL> element is used to convey the URL to which a PS provider will redirect the invited users after federating their IDP account to the PS provider.
<CreatePSObject> [Optional]  The <CreatePSObject> element is used as a directive with which a WSC indicates that it would like a PS provider create an Object for the inviting user at the PS provider of the invited user.

<Subscription> [Optional]  The <Subscription> element is used to indicate to the PS provider that the WSC desires that the PS provider return to it (when later possible) an identity token for the invited user.

<TokenPolicy> [Optional]  The <TokenPolicy> element is used as a container for the WSC’s policy requirements of an identity token.

The schema declaration for the <AddEntityRequest> message is shown below.

```xml
<AddEntityRequest>
  <Object NodeType="urn:liberty:ps:entity">
    <DisplayName>Alison</DisplayName>
  </Object>
  <SPRedirectURL>some SP URL</SPRedirectURL>
</AddEntityRequest>
```

The following is an example of an <AddEntityRequest> message used to create an Object for a user. The WSC is not requesting that an identity token for the newly created user Object be returned. If it desired this, it would include a <Subscription> element and, optionally, a <TokenPolicy> element indicating its requirements of the returned identity token.

```
<AddEntityRequest>
  <Object NodeType="urn:liberty:ps:entity">
    <DisplayName>Alison</DisplayName>
  </Object>
  <SPRedirectURL>some SP URL</SPRedirectURL>
</AddEntityRequest>
```

### 3.8.2. AddEntityResponse Message

A PS provider responds to an <AddEntityRequest> message with an <AddEntityResponse> message containing the newly created <Object> element.

The <AddEntityResponse> message has the complex type AddEntityResponseType, which extends ResponseAbstractType and adds the following elements:

<Object> [Optional]  The <Object> element is used to convey an Object data that has been created at the PS provider.

<PSRedirectURL> [Optional]  The <PSRedirectURL> element is used to convey the URL to which the PS provider desires the invited user be sent if and when they respond to the invitation that the SP will compose and deliver.
The schema declaration for the <AddEntityResponse> message is shown below.

```xml
<!-- Declaration of AddEntityResponse element -->
<xs:element name="AddEntityResponse" type="AddEntityResponseType"/>

<!-- Definition of AddEntityResponseType -->
<xs:complexType name="AddEntityResponseType">
  <xs:complexContent>
    <xs:extension base="ResponseAbstractType">
      <xs:sequence>
        <xs:element ref="Object" minOccurs="0"/>
        <xs:element ref="PSRedirectURL" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The following is an example of an <AddEntityResponse> to the <AddEntityRequest> message above. The PS provider is responding that the request that Alison be added was successful and returns the created <Object> element and <PSRedirectURL> element to which the PS provider desires the invited user be sent if and when they respond to the invitation that the SP will compose and deliver.

```xml
<AddEntityResponse>
  <Status code="OK"/>
  <Object NodeType="urn:liberty:ps:entity">
    <objectID>https://ps.com/kudfhgs</objectID>
    <DisplayName>Alison</DisplayName>
  </Object>
  <PSRedirectURL>some PS URL</PSRedirectURL>
</AddEntityResponse>
```

### 3.8.3. Processing Rules

The WSC:

- MUST include an <Object> element within the <AddEntityRequest> message.
- MUST include a NodeType attribute on the <Object> element with a value of urn:liberty:ps:entity.
- MUST include at least one <DisplayName> element for the invited user within the <Object> element. This element contains the friendly name that the user desires be used for the created urn:liberty:ps:entity Object.
- MAY include a <SPRedirectURL> element.
- MAY, if it desires that the PS provider create an Object for the inviting user at the PS provider of the invited user, include a <CreatePSObject> element.
- MAY, if it desires that an identity token be returned to it through a subsequent <Notification> message, include a <Subscription> element. The presence of a <Subscription> element indicates to the PS provider that the SP desires that the PS provider return to it (when later possible) an identity token for the invited user within a <Notification> message - this possible after a federation has been established between the PS provider and the appropriate IDP. If no <Subscription> element is present, the SP is indicating that the PS provider need not return an identity token through this mechanism.

In responding to an <AddEntityRequest> message, the PS provider:
• SHOULD include a <PSRedirectURL> element in the <AddEntityResponse> message returned to the calling SP.

• MUST be prepared for the invited user to, at some point in the future, visit the URL provided in the <PSRedirectURL> element. As it may be some time before the invited user does respond, the PS provider SHOULD store this url for a reasonable length of time.

• MUST, if and when the invited user does respond to the url specified by the <PSRedirectURL> element, endeavor to establish a federated identity for that user with the appropriate identity provider (see Section 4).

• SHOULD, if and when such a federated identifier is established, send an <sa:TokenMap> message to that IDP requesting an identity token (targeted at itself as the provider) for the user for which the federated identifier was established.

• SHOULD, if the <AddEntityRequest> message contained a <CreatePSObject> element, send either a <AddEntityRequest> or <AddKnownEntityRequest> message to the relevant People Service (that of the user for which a federation was just established).

The <AddKnownEntityRequest> message option allows the first PS provider to specify to the second PS provider an identity token for the user for whom the object is being created, thereby potentially removing the necessity of user interaction.

• SHOULD, if the <AddEntityRequest> message contained a <Subscription> element, send an <sa:TokenMap> message to that IDP requesting an identity token for the user for which the federated identifier was established but in the namespace of the requesting SP.

This <sa:TokenMap> message to the IDP MUST include any <sa:NameIDPolicy> present in the <AddEntityRequest>.

• SHOULD, if the <AddEntityRequest> message contained a <Subscription> element and the <sa:TokenMap> message to the IDP resulted in an identity token for the user being returned, forward on this identity token to the SP within a <Notification> message corresponding to the original <Subscription> element.

3.9. Adding a Known Entity

If a WSC knows an identifier for a user at some identity provider, it can provide this to the PS provider in an <AddKnownEntityRequest> message. This known identifier can act as a bootstrap for the establishment of the necessary federations. For instance, if the inviting user provides an email address for the invited user, this address may allow the identity provider for that user to be ascertained, thereby obviating the need to ask the user for this information.

A WSC indicates to the PS provider that it wishes a known user Object to be created by sending an <AddKnownEntityRequest> message. The Object being created MUST be a urn:liberty:ps:entity Object.

The <AddKnownEntityRequest> message carries the known identifier for the relevant user within.

As for the <AddEntityRequest> message, the presence of a <Subscription> element indicates to the PS provider that the WSC desires that the PS provider return to it (when later possible) an identity token for the invited user within a <Notification> message - this possible after a federation has been established between the PS provider and the appropriate IDP. If no <Subscription> element is present, the WSC is indicating that the PS provider need not return an identity token through this mechanism.

3.9.1. AddKnownEntityRequest Message

A WSC uses the <AddKnownEntityRequest> message to request that a specified <Object> be created for the known user.
The `<AddKnownEntityRequest>` MUST NOT be used to add a new `Object` to an existing `Object`, nor to create two nested `Objects`.

The `<AddKnownEntityRequest>` MUST include an appropriate identity token for the target `Object` being created. The `<sec:Token>` element will carry the known identifier for the user.

The `<AddKnownEntityRequest>` message has the complex type `AddKnownEntityRequestType`, which extends `RequestAbstractType` and adds the following elements:

- `<Object>` [Required] The `<Object>` element is used to convey the target `Object` being added.
- `<sec:Token>` [Required] The `<sec:Token>` element is used to convey an identity token for the target user `Object` being created.
- `<CreatePSObject>` [Optional] The `<CreatePSObject>` element is used as a directive with which a WSC indicates that it would like a PS provider create an `Object` for the inviting user at the PS provider of the invited user.
- `<Subscription>` [Optional] The `<Subscription>` element is used to indicate to the PS provider that the WSC desires that the PS provider return to it (when later possible) an identity token for the invited user.
- `<TokenPolicy>` [Optional] The `<TokenPolicy>` element is used as a container for WSC’s requirements to an identity token.

The schema declaration for the `<AddKnownEntityRequest>` message is shown below.

```xml
    <!-- Declaration of AddKnownEntityRequest element -->
    <xs:element name="AddKnownEntityRequest" type="AddKnownEntityRequestType"/>
    <!-- Definition of AddKnownEntityRequestType -->
    <xs:complexType name="AddKnownEntityRequestType">
        <xs:complexContent>
            <xs:extension base="RequestAbstractType">
                <xs:sequence>
                    <xs:element ref="Object"/>
                    <xs:element ref="Token"/>
                    <xs:element ref="CreatePSObject" minOccurs="0"/>
                    <xs:element ref="Subscription" minOccurs="0"/>
                    <xs:element ref="TokenPolicy" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

The following is an example of an `<AddKnownEntityRequest>` message used to create an `Object` for a user.

```xml
    <AddKnownEntityRequest>
        <Object NodeType="urn:liberty:ps:entity">
            <DisplayName>Bob</displayName>
        </Object>
        <sec:Token>
            <saml:Assertion>
                <saml:Subject>
                    <saml:NameID></saml:NameID>
                </saml:Subject>
            </saml:Assertion>
        </sec:Token>
    </AddKnownEntityRequest>
```
3.9.2. AddKnownEntityResponse Message

A PS provider responds to an <AddKnownEntityRequest> message with an <AddKnownEntityResponse> message, in which the PS provider MAY contain the newly created <Object> element.

The <AddKnownEntityResponse> message has the complex type AddKnownEntityResponseType, which extends ResponseAbstractType and adds the following elements:

- **<Object> [Optional]** The <Object> element is used to convey an Object data that has created at the PS provider.

- **<PSRedirectURL> [Optional]** The <PSRedirectURL> element is used to convey the URL to which the PS provider desires the invited user be sent if and when they respond to the invitation that the SP will compose and deliver.

The schema declaration for the <AddKnownEntityResponse> message is shown below.

```xml
<xs:element name="AddKnownEntityResponse" type="AddKnownEntityResponseType"/>
<xs:complexType name="AddKnownEntityResponseType">
  <xs:complexContent>
    <xs:extension base="ResponseAbstractType">
      <xs:sequence>
        <xs:element ref="Object" minOccurs="0"/>
        <xs:element ref="PSRedirectURL" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The following is an example of an <AddKnownEntityResponse> to the <AddKnownEntityRequest> message above. The PS provider is responding that the request that Bob be added was successful and returns the created <Object> element.

```xml
<AddKnownEntityResponse>
  <Status code="OK"/>
  <Object NodeType="urn:liberty:ps:entity">
    <ObjectID>https://ps.com/lafnervf</ObjectID>
    <DisplayName>Bob</DisplayName>
  </Object>
</AddKnownEntityResponse>
```

3.9.3. Processing Rules

The WSC:

- MUST include an <Object> element within the <AddKnownEntityRequest> message.
- MUST include a NodeType attribute on the <Object> element with a value of urn:liberty:ps:entity.
- MUST include a <Token> element within the <AddKnownEntityRequest> message.
• MUST, if a SAML `<Assertion>` is used as the identity token format, specify the known identifier in that assertion's `<Subject>` element.

• MUST include at least one `<DisplayName>` for the invited user within the `<Object>` element. This element contains the friendly name that the user desires be used for the created user `Object`.

• MAY, if it desires that the PS provider create an `Object` for the inviting user at the PS provider of the invited user, include a `<CreatePSObject>` element.

• MAY, if it desires that an identity token be returned to it through a subsequent `<Notification>` message, include a `<Subscription>` element. The presence of a `<Subscription>` element indicates to the PS provider that the SP desires that the PS provider return to it (when later possible) an identity token for the invited user within a `<Notification>` message - this possible after a federation has been established between the PS provider and the appropriate IDP. If no `<Subscription>` element is present, the SP is indicating that the PS provider need not return an identity token through this mechanism.

In responding to an `<AddKnownEntityRequest>` message, the PS provider:

• MAY include a `<PSRedirectURL>` element in the `<AddKnownEntityResponse>` message returned to the calling SP.

• MUST be prepared for the invited user to, at some point in the future, visit the URL provided in the `<PSRedirectURL>` element. As it may be some time before the invited user does respond, the PS provider SHOULD store this url for a reasonable length of time.

• MUST, if and when the invited user does respond to the URL specified by the `<PSRedirectURL>` element, endeavor to establish a federated identity for that user with the appropriate identity provider (see Section 4).

• SHOULD, if and when such a federated identifier is established, send a `<sa:TokenMap>` message to that IDP requesting an identity token (targeted at itself as the provider) for the user for which the federated identifier was established.

• SHOULD, if the `<AddKnownEntityRequest>` message contained a `<Subscription>` element, send a `<sa:TokenMap>` message to that IDP requesting an identity token for the user for which the federated identifier was established but in the namespace of the requesting SP. This `<sa:TokenMap>` message to the IDP MUST include any `<sa:NameIDPolicy>` present in the `<AddKnownEntityRequest>`.

• SHOULD, if the `<AddKnownEntityRequest>` message contained a `<Subscription>` element and the `<sa:TokenMap>` message to the IDP resulted in an identity token for the user being returned, forward on this identity token to the SP within a `<Notification>` message corresponding to the original `<Subscription>` element.

• SHOULD, if the `<AddKnownEntityRequest>` message contained a `<CreatePSObject>` element, send either an `<AddEntityRequest>` or `<AddKnownEntityRequest>` message to the relevant People Service (that of the user for which a federation was just established).

The `<AddKnownEntityRequest>` message option allows the first PS provider to specify to the second PS provider an identity token for the user for whom the object is being created, thereby potentially removing the necessity of user interaction.
3.10. Removing an Entity

A WSC indicates to the PS provider that it wishes a user Object to be completely removed from the PS resource by sending a `<RemoveEntityRequest>` message. The Object being removed MUST be a `urn:liberty:ps:entity Object`.

3.10.1. RemoveUser Message

A WSC uses the `<RemoveEntityRequest>` message to request that a user Object corresponding to the value of the specified `<TargetID>` element be removed.

The `<RemoveEntityRequest>` message is used to completely remove a user Object from the PS resource. To simply remove a child user Object element from some parent group Object, the `<RemoveEntityRequest>` message MUST NOT be used, but rather a `<RemoveFromCollectionRequest>` message with the parent Object's ObjectID specified in the `<TargetID>` element, MUST be used (see Section 3.14 for more details).

The `<RemoveEntityRequest>` message has the complex type `RemoveEntityRequestType`, which extends `RequestAbstractType` and adds the following element:

- `<TargetID>` [Required] The `<TargetID>` element is used to convey one or more ObjectID's of the target user Objects being removed.

The schema declaration for the `<RemoveEntityRequest>` message is shown below.

```xml
<!-- Declaration of RemoveEntityRequest element -->
<xs:element name="RemoveEntityRequest" type="RemoveEntityRequestType"/>
<!-- Definition of RemoveEntityRequestType -->
<xs:complexType name="RemoveEntityRequestType">
    <xs:complexContent>
        <xs:extension base="RequestAbstractType">
            <xs:sequence>
                <xs:element ref="TargetID" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The following is an example of a `<RemoveEntityRequest>` message used to remove an Object for a user.

```xml
<RemoveEntityRequest>
    <TargetID>https://ps.com/lafnervf</TargetID>
</RemoveEntityRequest>
```

3.10.2. RemoveEntityResponse Message

A PS provider responds to a `<RemoveEntityRequest>` message with a `<RemoveEntityResponse>` element. A PS provider removes the specified Object and responds a status of this process based on the processing rules described in section Section 3.10.3.

The `<RemoveEntityResponse>` message has the type of `ResponseAbstractType`.

The schema declaration for the `<RemoveEntityResponse>` message is shown below.

```xml
<!-- Declaration of RemoveEntityResponse element -->
<xs:element name="RemoveEntityResponse" type="ResponseAbstractType"/>
```
The following is an example of a `<RemoveEntityResponse>` to the `<RemoveEntityRequest>` message above. The PS provider is responding that the request that a specified `Object` be removed was successful.

```xml
<RemoveEntityResponse>
  <Status code="OK"/>
</RemoveEntityResponse>
```

### 3.10.3. Processing Rules

The WSC:

- MUST ensure that the targeted `<Object>` has a `NodeType` attribute with a value of `urn:liberty:ps:entity`.

The PS provider:

- MUST, if the specified user object is not a direct member of the targeted group object, respond `Failed` as the code attribute of the top level `<Status>` element, and the code attribute of the second level `<Status>` element MUST be set with the following status code:

  - `NotDirectChild`

- MAY cancel any existing federated identifier with the relevant IDP for that user being removed.

### 3.11. Adding a Collection

A WSC indicates to the PS provider that it wishes a group `Object` to be created by sending an `<AddCollectionRequest>` message. The `Object` being created MUST be a `urn:liberty:ps:collection Object`.

#### 3.11.1. AddCollectionRequest Message

A WSC uses the `<AddCollectionRequest>` message to request that a specified group `<Object>` be created.

The `<AddCollectionRequest>` MUST NOT be used to add a new group `Object` to an existing group `Object`, nor to create two nested `Objects`.

The `<AddCollectionRequest>` message has the complex type `AddCollectionRequestType`, which extends `RequestAbstractType` and adds the following element:

- `<Object> [Required]` The `<Object>` element is used to convey the target group `Object` being added.

- `<Subscription> [Optional]` The `<Subscription>` element is used to indicate to the PS provider that the WSC desires that the PS provider send a notification if and when the group object being added changes.
The schema declaration for the `<AddCollectionRequest>` message is shown below.

```xml
<!-- Declaration of AddCollectionRequest element -->
<xs:element name="AddCollectionRequest" type="AddCollectionRequestType"/>
<!-- Definition of AddCollectionRequestType -->
<xs:complexType name="AddCollectionRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="Object"/>
        <xs:element ref="Subscription" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The following is an example of an `<AddCollectionRequest>` message used to create an `Object` for a group.

```xml
<AddCollectionRequest>
  <Object NodeType="urn:liberty:ps:collection">
    <DisplayName>Soccer Team</DisplayName>
  </Object>
</AddCollectionRequest>
```

### 3.11.2. AddCollectionResponse Message

A PS provider responds to an `<AddCollectionRequest>` message with an `<AddCollectionResponse>` message containing the newly created `<Object>` element.

The `<AddCollectionResponse>` message has the complex type `AddCollectionResponseType`, which extends `ResponseAbstractType` and adds the following element:

- **Object** [Optional]: The `<Object>` element is used to convey an `Object` data that has created at the PS provider.

The schema declaration for the `<AddCollectionResponse>` message is shown below.

```xml
<!-- Declaration of AddCollectionResponse element -->
<xs:element name="AddCollectionResponse" type="AddCollectionResponseType"/>
<!-- Definition of AddCollectionResponseType -->
<xs:complexType name="AddCollectionResponseType">
  <xs:complexContent>
    <xs:extension base="ResponseAbstractType">
      <xs:sequence>
        <xs:element ref="Object" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The following is an example of an `<AddCollectionResponse>` to the `<AddCollectionRequest>` message above.

The PS provider is responding that the request that the Soccer Team be added was successful and returns the created `<Object>` element.

```xml
<AddCollectionResponse>
  <Status code="OK"/>
  <Object NodeType="urn:liberty:ps:collection">
```
3.11.3. Processing Rules

The WSC:

• MUST include an <Object> element within the <AddCollectionRequest> message.
• MUST include a NodeType attribute on the <Object> element with a value of urn:liberty:ps:collection.
• MUST include at least one <DisplayName> element for a group within the <Object>. This element contains the friendly name that the user desires be used for the created group Object.

In responding to an <AddCollectionRequest> message, the PS provider:

• MUST return an <Object> element within the <AddCollectionResponse> message with the same <DisplayName> as specified on the <AddCollectionRequest>.
• MUST include an <ObjectID> element for the newly created group Object.

3.12. Removing a Collection

A WSC indicates to the PS provider that it wishes a group Object to be removed by sending a <RemoveCollectionRequest> message. The Object being removed MUST be a urn:liberty:ps:collection Object.

3.12.1. RemoveCollectionRequest Message

A WSC uses the <RemoveCollectionRequest> message to request that a group Object corresponding to the value of the specified <TargetID> be removed.

The <RemoveCollectionRequest> message is used to completely remove a group Object from the PS resource. To simply remove a child group <Object> element from some parent group <Object>, the <RemoveCollectionRequest> message MUST NOT be used, but rather a <RemoveFromCollectionRequest> with the parent Object’s ObjectID specified in the TargetID element. MUST be used (see Section 3.14 for more details).

The <RemoveCollectionRequest> message has the complex type RemoveCollectionRequestType, which extends RequestAbstractType and adds the following element:

<TargetID> [Required] The <TargetID> element specifies the ObjectID of the targeted group Objects being removed.
The schema declaration for the `<RemoveCollectionRequest>` message is shown below.

```xml
<xs:element name="RemoveCollectionRequest" type="RemoveCollectionRequestType"/>
```

The following is an example of a `<RemoveCollectionRequest>` message used to remove an Object for a group.

```xml
<RemoveCollectionRequest>
  <TargetID>https://ps.com/roqlsfof</TargetID>
</RemoveCollectionRequest>
```

### 3.12.2. RemoveCollectionResponse Message

A PS provider responds to a `<RemoveCollectionRequest>` message with a `<RemoveCollectionResponse>` element. A PS provider removes the specified `<Object>` and responds a status of this process based on the processing rules described in Section 3.10.3.

The `<RemoveCollectionResponse>` message has the type of `ResponseAbstractType`.

The schema declaration for the `<RemoveCollectionResponse>` message is shown below.

```xml
<xs:element name="RemoveCollectionResponse" type="ResponseAbstractType"/>
```

The following is an example of a `<RemoveCollectionResponse>` to the `<RemoveCollectionRequest>` message above. The PS provider is responding that the request that a specified Object be removed was successful.

```xml
<RemoveCollectionResponse>
  <Status code="OK"/>
</RemoveCollectionResponse>
```

### 3.12.3. Processing Rules

**The WSC:**
- MUST include a `NodeType` attribute on the `<Object>` element with a value of `urn:liberty:ps:collection`.

**The PS provider:**
- MUST, if the specified user object is not a direct member of the targeted group object, respond `Failed` as the code attribute of the top level `<Status>` element, and the code attribute of the second level `<Status>` element MUST be set with the following status code:
3.13. Adding to a Collection

A WSC uses the <AddToCollectionRequest> message to request that child Object elements be added to an existing group Object. Both user and group Objects can be added to a parent group Object with the <AddToCollectionRequest> message.

3.13.1. AddToCollectionRequest Message

The target parent group Object to which child Objects are to be added is indicated by the value of the <TargetID> element within the <AddToCollectionRequest> element. The child Object’s being added are specified by the values of the <ObjectID> elements within the <AddToCollectionRequest> element.

The <AddToCollectionRequest> message has the complex type AddToCollectionRequestType, which extends RequestAbstractType and adds the following elements:

- **<TargetID> [Required]** The <TargetID> element is used to convey the ObjectID of the target group Object to which specified Objects are added.
- **<ObjectID> [Required]** The <ObjectID> element is used to convey ObjectID's of the Objects that would be added to the target group Object.
- **<Subscription> [Optional]** The <Subscription> element is used to indicate to the PS provider that the WSC desires that the PS provider send a notification if and when the membership of the targeted group changes.

The schema declaration for the <AddToCollectionRequest> message is shown below.

```
<!-- Declaration of AddToCollectionRequest element -->
<xsd:element name="AddToCollectionRequest" type="AddToCollectionRequestType"/>
<!-- Definition of AddToCollectionRequestType -->
<xsd:complexType name="AddToCollectionRequestType">
  <xsd:complexContent>
    <xsd:extension base="RequestAbstractType">
      <xsd:sequence>
        <xsd:element ref="TargetID"/>
        <xsd:element ref="ObjectID" minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element ref="Subscription" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The following is an example of an <AddToCollectionRequest> message used to add three Objects to the target group Object.

```
<AddToCollectionRequest>
  <TargetID>https://ps.com/roql sfof</TargetID>
  <ObjectID>https://ps.com/qaf3eflo</ObjectID>
  <ObjectID>https://ps.com/bzpfnrns</ObjectID>
  <ObjectID>https://ps.com/nsdf lfss</ObjectID>
</AddToCollectionRequest>
```
3.13.2. AddToCollectionResponse Message

A PS provider responds to an `<AddToCollectionRequest>` message with an `<AddToCollectionResponse>` message. The PS provider makes the indicated modification to the specified target `<Object>` and responds with the status.

The `<AddToCollectionResponse>` message has the type of `ResponseAbstractType`

The schema declaration for the `<AddToCollectionResponse>` message is shown below.

```
<!-- Declaration of AddToCollectionResponse element -->
<xs:element name="AddToCollectionResponse" type="ResponseAbstractType"/>
```

The following is an example of an `<AddToCollectionResponse>` to the `<AddToCollectionRequest>` message above. The PS provider is responding that the request that the three Objects be added to the target Object was successful.

```
<AddToCollectionResponse>
  <Status code="OK"/>
</AddToCollectionResponse>
```

3.13.3. Processing Rules

The WSC:

• MUST specify, as a value of the `<TargetID>`, an `ObjectId` of the `Object` that has `urn:liberty:ps:collection` as a value of `NodeType` attribute.

The PS provider:

• MUST, if the `Object` specified by the value of the `<TargetID>` element has `urn:liberty:ps:entity` as the value of its `NodeType` attribute, respond `Failed` as the code attribute of the top level `<Status>` element, and the code attribute of the second level `<Status>` element MUST be set with the following status code:

  • `ObjectIsEntity`

3.14. Removing from a Collection

A WSC uses the `<RemoveFromGroup>` message to request the removal of a child `Object` element from a parent group `Object`. Both user and group `Objects` can be removed from a parent group `Object` with the `<RemoveFromGroup>` message.

3.14.1. RemoveFromCollectionRequest Message

The target parent group `Object` from which a child `Object` is to be removed is indicated by the value of the `<TargetID>` element within the `<RemoveFromCollectionRequest>` message. The child `Object` being removed are specified by the value(s) of the `<ObjectId>` elements within the `<RemoveFromCollectionRequest>` element.

The `<RemoveFromCollectionRequest>` message has the complex type `RemoveFromCollectionRequestType`, which extends `RequestAbstractType` and adds the following elements:

• `<TargetID>` [Required] The `<TargetID>` element is used to convey the `ObjectId` of the target group `Object` from which specified `Objects` are removed.
<ObjectID> [Required] The <ObjectID> element is used to convey ObjectID's of the Objects that would be removed from the target group Object.

<Subscription> [Optional] The <Subscription> element is used to indicate to the PS provider that the WSC desires that the PS provider send a notification if and when the membership of the targeted group changes.

The schema declaration for the <RemoveFromCollectionRequest> message is shown below.

<!-- Declaration of RemoveFromCollectionRequest element -->
<xs:element name="RemoveFromCollectionRequest" type="RemoveFromCollectionRequestType"/>
<!-- Definition of RemoveFromCollectionRequestType -->
<xs:complexType name="RemoveFromCollectionRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="TargetID"/>
        <xs:element ref="ObjectID" minOccurs="1" maxOccurs="unbounded"/>
        <xs:element ref="Subscription" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

The following is an example of a <RemoveFromCollectionRequest> message used to remove three Objects from the target group Object.

<RemoveFromCollectionRequest>
  <TargetID>https://ps.com/roqlsfof</TargetID>
  <ObjectID>https://ps.com/qaf3eflo</ObjectID>
  <ObjectID>https://ps.com/bzpfnrns</ObjectID>
  <ObjectID>https://ps.com/nsdflfss</ObjectID>
</RemoveFromCollectionRequest>

3.14.2. RemoveFromCollectionResponse Message

A PS provider responds to a <RemoveFromCollectionRequest> message with a <RemoveFromCollectionResponse> message. The PS provider makes the indicated modification to the specified target Object and responds with the status.

The <RemoveFromCollectionResponse> message has the type of ResponseAbstractType

The schema declaration for the <RemoveFromCollectionResponse> message is shown below.

<!-- Declaration of RemoveFromCollectionResponse element -->
<xs:element name="RemoveFromCollectionResponse" type="ResponseAbstractType"/>

The following is an example of a <RemoveFromCollectionResponse> to the <RemoveFromCollectionRequest> message above. The PS provider is responding that the request that the three objects be removed from the target Object was successful.

<RemoveFromCollectionResponse>
  <Status code="OK"/>
</RemoveFromCollectionResponse>
3.14.3. Processing Rules

The WSC:

- MUST specify, as a value of the <TargetID> element, an ObjectID of the Object that has urn:liberty:ps:collection as a value of NodeType attribute.

The PS provider:

- MUST, if the Object specified by the value of the <TargetID> element has urn:liberty:ps:entity as the value of its NodeType attribute, respond Failed as the code attribute of the top level <Status> element, and the code attribute of the second level <Status> element MUST be set with the following status code:

- ObjectIsEntity

3.15. Listing Members

A WSC uses the <ListMembersRequest> message to navigate the Object structure of the users (urn:liberty:ps:entity Objects) and groups (urn:liberty:ps:collection Objects) that comprise the PS resource.

A WSC can control how Objects are returned to it by specifying its preferences with the Structured attribute.

If a WSC does not specify a <TargetID> element in the <ListMembersRequest> message, this is equivalent to asking for all top-level Objects.

3.15.1. ListMembersRequest Message

The WSC indicates to the PS provider the Object of interest by specifying that <Object> element’s ObjectID in the <TargetID> element. If no <TargetID> element is specified in the <ListMembersRequest> message, the PS provider MUST return all the top-level Objects (i.e., Objects that do not have any parent Object.)

The <ListMembersRequest> message has the complex type ListMembersRequestType, which extends RequestAbstractType and adds the following attributes and elements:

- Structured [Optional] A flag with which a WSC indicates whether it wants to navigate the Object tree structure incrementally or to have the complete Object tree returned. See Section 3.15.1.1 for more detail.

- Count [Optional] The Count attribute specifies how many child <Object> elements should be returned in a <ListMembersResponse> message. See Section 3.15.1.2 for more detail.

- Offset [Optional] The Offset attribute specifies from which element to continue, when requesting for more data. See Section 3.15.1.2 for more detail.

- <TargetID> [Optional] The <TargetID> element is used to convey an ObjectID of the target group Object whose child Objects are to be listed.

- <Subscription> [Optional] The <Subscription> element is used to indicate to the PS provider that the WSC desires that the PS provider send a notification if and when the membership of the targeted group changes.
The schema declaration for the `<ListMembersRequest>` message is shown below.

```xml
<xs:element name="ListMembersRequest" type="ListMembersRequestType"/>
<xs:complexType name="ListMembersRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="TargetID" minOccurs="0"/>
        <xs:element ref="Subscription" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="Structured" type="xs:boolean" default="true" use="optional"/>
      <xs:attribute name="Count" type="xs:nonNegativeInteger" use="optional"/>
      <xs:attribute name="Offset" type="xs:nonNegativeInteger" default="0" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

3.15.1.1. Structured Attribute

WSC’s may choose to navigate the tree structure for a given Object incrementally (i.e. by successive requests to probe deeper) or to have the complete tree returned. The Structured attribute allows the WSC to indicate this preference to the PS provider.

If the value of the Structured attribute is set as false, then the PS provider MUST respond with an unstructured view of the Object tree and return only the descendant user <Object> elements of the specified Object. Any group hierarchy (if present) for the specified Object is ignored and only the descendant user Objects are returned.

If the value of the Structured attribute is set as true, then the PS provider MUST respond with all the direct child <Object> elements of the specified Object, either group or user Objects.

The default value for the Structured attribute is true.

3.15.1.2. Count and Offset Attributes

When the specified Object has multiple child Objects, the WSC may want to be responded the child <Object> elements in smaller sets (i.e., a smaller number of elements at a time). This is achieved by using the Count and Offset attributes of the <ListMembersRequest> element.

The Count attribute defines how many child <Object> elements should be returned in a <ListMembersResponse> message. The Count attribute only pertains to the direct child <Object> elements of the Object specified in the <ListMembersRequest> message.

The Offset attribute specifies from which element to continue, when requesting for more data. The default value is zero, which refers to the first child <Object> element.

3.15.2. ListMembersResponse Message

A PS provider responds to a <ListMembersRequest> message with a <ListMembersResponse> message containing the appropriate set of <Object> elements.

The <ListMembersResponse> message has the complex type ListMembersResponseType, which extends ResponseAbstractType and adds the following element:

<Object> [Optional] The <Object> element is used to convey data of zero or more Objects to be listed.
The schema declaration for the `<ListMembersResponse>` message is shown below.

```xml
<!-- Declaration of ListMembersResponse element -->
<xsd:element name="ListMembersResponse" type="ListMembersResponseType"/>
<!-- Definition of ListMembersResponseType -->
<xsd:complexType name="ListMembersResponseType">
  <xsd:complexContent>
    <xsd:extension base="ResponseAbstractType">
      <xsd:sequence>
        <xsd:element ref="Object" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

### 3.15.3. Examples

All the examples described in this subsection assume that a PS provider has the following virtual XML document for data.

```xml
<Object NodeType="urn:liberty:ps:entity">
  <ObjectID>https://ps.com/lsdjfojd</ObjectID>
  <DisplayName>Mary</DisplayName>
</Object>
<Object NodeType="urn:liberty:ps:entity">
  <ObjectID>https://ps.com/sijfsfsf</ObjectID>
  <DisplayName>Bob</DisplayName>
</Object>
<Object NodeType="urn:liberty:ps:entity">
  <ObjectID>https://ps.com/reremvls</ObjectID>
  <DisplayName>Nick</DisplayName>
</Object>
<Object NodeType="urn:liberty:ps:entity">
  <ObjectID>https://ps.com/soijfsfd</ObjectID>
  <DisplayName>JoJo</DisplayName>
</Object>
<Object NodeType="urn:liberty:ps:entity">
  <ObjectID>https://ps.com/sdosafms</ObjectID>
  <DisplayName>Taro</DisplayName>
</Object>
<Object NodeType="urn:liberty:ps:entity">
  <ObjectID>https://ps.com/lgsdfsfd</ObjectID>
  <DisplayName>Hanako</DisplayName>
</Object>
<Object NodeType="urn:liberty:ps:collection">
  <ObjectID>https://ps.com/eiruvoie</ObjectID>
  <DisplayName>Soccer Team</DisplayName>
  <ObjectRef Ref="https://ps.com/lsdjfojd"/>
  <ObjectRef Ref="https://ps.com/sijfsfsf"/>
  <ObjectRef Ref="https://ps.com/reremvls"/>
  <ObjectRef Ref="https://ps.com/soijfsfd"/>
</Object>
<Object NodeType="urn:liberty:ps:collection">
  <ObjectID>https://ps.com/zxlidfdf</ObjectID>
  <DisplayName>Starting Members</DisplayName>
  <ObjectRef Ref="https://ps.com/lsdjfojd"/>
  <ObjectRef Ref="https://ps.com/sijfsfsf"/>
</Object>
<Object NodeType="urn:liberty:ps:collection">
  <ObjectID>https://ps.com/merflas</ObjectID>
  <DisplayName>Family</DisplayName>
  <ObjectRef Ref="https://ps.com/sdosafms"/>
</Object>
```

Liberty Alliance Project

30
The following is an example of a `ListMembersRequest` message by which a WSC requests the list of child elements that belong to the "Soccer Team" group `Object`. As the WSC specifies `Structured="false"`, it is indicating that it desires to have a 'flat' view of that group's `Object` tree returned.

```xml
<ListMembersRequest Structured="false">
  <TargetID>https://ps.com/eiruvoe</TargetID>
</ListMembersRequest>
```

The following is an example `ListMembersResponse` message as might be returned to the `ListMembersRequest` above.

```xml
<ListMembersResponse>
  <Status code="OK"/>
  <Object NodeType="urn:liberty:ps:entity">
    <ObjectID>https://ps.com/lsdjfojd</ObjectID>
    <DisplayName>Mary</DisplayName>
  </Object>
  <Object NodeType="urn:liberty:ps:entity">
    <ObjectID>https://ps.com/sijfsfsf</ObjectID>
    <DisplayName>Bob</DisplayName>
  </Object>
  <Object NodeType="urn:liberty:ps:entity">
    <ObjectID>https://ps.com/reremvl s</ObjectID>
    <DisplayName>Nick</DisplayName>
  </Object>
  <Object NodeType="urn:liberty:ps:entity">
    <ObjectID>https://ps.com/soijfsfd</ObjectID>
    <DisplayName>JoJo</DisplayName>
  </Object>
</ListMembersResponse>
```

As the WSC indicated it desired a non-structured view, the PS expands the group `Object` called "Starting Members" and returns its two child user `Objects` instead of the group `Object` itself.

Alternatively, the following is an example `ListMembersResponse` message as might be returned to a `ListMembersRequest` message in which the WSC indicated it desired a structured view for the returned data by specifying `Structured="true"`.

```xml
<ListMembersResponse>
  <Status code="OK"/>
  <Object NodeType="urn:liberty:ps:collection">
    <ObjectID>https://ps.com/nmerflas</ObjectID>
    <DisplayName>Starting Members</DisplayName>
  </Object>
  <Object NodeType="urn:liberty:ps:entity">
    <ObjectID>https://ps.com/reremvl s</ObjectID>
    <DisplayName>Nick</DisplayName>
  </Object>
  <Object NodeType="urn:liberty:ps:entity">
    <ObjectID>https://ps.com/soijfsfd</ObjectID>
    <DisplayName>JoJo</DisplayName>
  </Object>
</ListMembersResponse>
```
3.15.4. Processing Rules

- All the <Object> elements, that a PS provider responds within the <ListMembersResponse> message, MUST NOT have any child <Object> element and/or <ObjectRef> element. If an <Object> being responded has some child <Object> element and/or <ObjectRef> element in the virtual XML document at the PS provider, the PS provider MUST remove them from the <ListMembersResponse>.

- When a PS provider receives a <ListMembersRequest> message with a <TargetID> element specified, and if the value of the Structured attribute is set as false, then the PS provider MUST respond return only the descendant user <Object> elements of the specified Object. Any group hierarchy (if present) for the specified Object is ignored and only the descendant user Objects are returned.

- When a PS provider receives a <ListMembersRequest> message with no <TargetID> element specified, and if the value of the Structured attribute is set as false, then the PS provider MUST respond return only all descendant user <Object> elements.

- When a PS provider receives a <ListMembersRequest> message with a <TargetID> element specified, and if the value of the Structured attribute is set as true, then the PS provider MUST respond with all the direct child <Object> elements of the specified Object, either group or user Objects.

- When a PS provider receives a <ListMembersRequest> message with no <TargetID> element specified, and if the value of the Structured attribute is set as true, then the PS provider MUST respond with all the <Object> elements that do not have any parent Objects (i.e., all the top-level Objects in the virtual XML document at the PS provider), either group or user Objects.

- If a PS provider receives a <ListMembersRequest> message on which the value of <TargetID> matches that of an <ObjectID> element of a given Object, and if the value of the NodeType attribute of the Object is urn:liberty:ps:entity, then the PS provider MUST respond Failed as the code attribute of the top level <Status> element, and the code attribute of the second level <Status> element MUST be set with the following status code:

  • ObjectIsEntity

3.16. Retrieving Info

A WSC uses the <GetObjectInfoRequest> message to retrieve information for a particular Object. An Object’s information includes all the child elements and attributes of the <Object> element, except for either <Object> or <ObjectRef> elements.

Note that if a WSC wants to get child members’s Objects of the particular Object, the WSC MUST use <ListMembersRequest> message (see Section 3.15).

3.16.1. GetObjectInfoRequest Message

The WSC indicates to the PS provider the Object of interest by specifying that <Object> element’s ObjectID in the <TargetID> element.

The <GetObjectInfoRequest> message has the complex type GetObjectInfoRequestType, which extends RequestAbstractType and adds the following element:

- <TargetID> [Required] The <TargetID> element is used to convey an ObjectID of the target Object of which information is requested.

- <Subscription> [Optional] The <Subscription> element is used to indicate to the PS provider that the WSC desires that the PS provider send a notification if and when the information (but not membership) of the targeted Object changes.
The schema declaration for the `<GetObjectInfoRequest>` message is shown below.

```xml
<!-- Declaration of GetObjectInfoRequest element -->
<xs:element name="GetObjectInfoRequest" type="GetObjectInfoRequestType"/>
<!-- Definition of GetObjectInfoRequestType -->
<xs:complexType name="GetObjectInfoRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="TargetID"/>
        <xs:element ref="Subscription" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The following is an example of a `<GetObjectInfoRequest>` message.

```xml
<GetObjectInfoRequest>
  <TargetID>https://ps.com/eiruvoie</TargetID>
</GetObjectInfoRequest>
```

### 3.16.2. GetObjectInfoResponse Message

A PS provider responds to a `<GetObjectInfoRequest>` message with a `<GetObjectInfoResponse>` message, on which the specified `Object`’s information is conveyed, except for child `Object`’s information.

The `<GetObjectInfoResponse>` message has the complex type `GetObjectInfoResponseType`, which extends `ResponseAbstractType` and adds the following elements:

- `<Object>` [Required] The `<Object>` element is used to convey the information of the requested `Object`.

The schema declaration for the `<GetObjectInfoResponse>` message is shown below.

```xml
<!-- Declaration of GetObjectInfoResponse element -->
<xs:element name="GetObjectInfoResponse" type="GetObjectInfoResponseType"/>
<!-- Definition of GetObjectInfoResponseType -->
<xs:complexType name="GetObjectInfoResponseType">
  <xs:complexContent>
    <xs:extension base="ResponseAbstractType">
      <xs:sequence>
        <xs:element ref="Object"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The following is an example of a `<GetObjectInfoResponse>` to the `<GetObjectInfoRequest>` message above.

```xml
<GetObjectInfoResponse>
  <Status code="OK"/>
  <Object NodeType="urn:liberty:ps:collection">
    <ObjectID>https://ps.com/eiruvoie</ObjectID>
    <DisplayName>Soccer Team</DisplayName>
  </Object>
</GetObjectInfoResponse>
```
3.16.3. Processing Rules

A PS provider:

- MUST NOT respond with any child <Object> and/or <ObjectRef> elements of the <Object> specified with the ObjectID in the <TargetID> element on the <GetObjectInfoRequest> message by a WSC.

3.17. Updating Info

A WSC may wish to update or modify the information for an Object, e.g. to change a DisplayName etc.

A WSC uses the <SetObjectInfoRequest> message to update the information for a particular Object. Updateable information does not include <Object> element, <ObjectRef> element, NodeType attribute, CreatedDateTime attribute, and ModifiedDateTime attribute.

Note that if a WSC wants to insert a child Object to the particular Object, the WSC MUST use <AddToCollectionRequest> message (see Section 3.13). Also note that if a WSC wants to remove a child Object from the particular Object, the WSC MUST use <RemoveFromCollectionRequest> message (see Section 3.14).

3.17.1. SetObjectInfoRequest Message

The WSC specifies <Object> elements of the target Object for updating. These <Object> elements MUST NOT have child <Object> and/or <ObjectRef> elements. Also, these <Object> elements MUST NOT have CreatedDateTime and ModifiedDateTime attributes.

The <SetObjectInfoRequest> message has the complex type SetObjectInfoRequestType, which extends RequestAbstractType and adds the following element:

(Object) [Required] The <Object> element is used to convey the updated data of the Object to be updated.

<Subscription> [Optional] The <Subscription> element is used to indicate to the PS provider that the WSC desires that the PS provider send a notification if and when the information of the targeted Object changes.

The schema declaration for the <SetObjectInfoRequest> message is shown below.

The following is an example of a <SetObjectInfoRequest> message in which the WSC is changing the value of the <DisplayName> element for the specified <Object>. The previously existing value for this element would be overwritten.

<SetObjectInfoRequest>
  <Object NodeType="urn:liberty:ps:collection"/>
</SetObjectInfoRequest>
3.17.2. SetObjectInfoResponse Message

A PS provider responds to a <SetObjectInfoRequest> message with a <SetObjectInfoResponse> message.

The <SetObjectInfoResponse> message has the type of ResponseAbstractType.

The schema declaration for the <SetObjectInfoResponse> message is shown below.

```xml
<xs:element name="SetObjectInfoResponse" type="ResponseAbstractType"/>
```

The following is an example of a <SetObjectInfoResponse> to the <SetObjectInfoRequest> message above.

```xml
<SetObjectInfoResponse>
  <Status code="OK"/>
</SetObjectInfoResponse>
```

3.17.3. Processing Rules

A PS provider:

- MUST, if it can not find the target Object specified with the ObjectID, respond Failed as the code attribute of the top level <Status> element, and the code attribute of the second level <Status> element MUST be set with the following status code:

  • CannotFindObject

- MUST, if it finds that the value of the specified NodeType attribute is different from the value of the NodeType attribute of the target Object specified with the ObjectID, respond Failed as the code attribute of the top level <Status> element, and the code attribute of the second level <Status> element MUST be set with the following status code:

  • InvalidNodeType

- MUST, if it finds that a WSC specifies the <Object> element, <ObjectRef> element, CreatedDateTime attribute, or ModifiedDateTime, ignore these elements and/or attributes.
3.18. Querying Objects

A WSC may wish to have returned to it a list of Objects that meet some criteria. The <QueryObjectsRequest> message allows the WSC to indicate this of the PS provider. The criteria to be met are specified in the <Filter> element in the <QueryObjectsRequest> element.

Note: The <ListMembers> message can be considered a specialization of the <QueryObjectsRequest> message, in which the criteria to be met is that the returned Objects are members of the specified group Object. The <QueryObjectsRequest> message allows a WSC to pose more generalized queries, e.g. to which groups does a specific user belong?

3.18.1. QueryObjectsRequest Message

The WSC specifies criteria of its interest in the <Filter> element.

The <QueryObjectsRequest> message has the complex type QueryObjectsRequestType, which extends RequestAbstractType and adds the following elements:

- <Filter> [Required] The <Filter> element is used to convey criteria for matching Object elements that a WSC is interested in.

- <Subscription> [Optional] The <Subscription> element is used to indicate to the PS provider that the WSC desires that the PS provider send a notification if and when the set of objects that satisfy the specified filter changes.

The schema declaration for the <QueryObjectsRequest> message is shown below.

```xml
<xs:element name="QueryObjectsRequest" type="QueryObjectsRequestType"/>
<xs:complexType name="QueryObjectsRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element name="Filter" type="xs:string"/>
        <xs:element ref="Subscription" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The value of the <Filter> element SHOULD be specified with an XPath expression.

For instance, if the WSC wants to get all the <Object> elements with a NodeType attribute with a value of urn:liberty:ps:collection (i.e., the WSC wants to get all the group Objects), the WSC can specify the value of the <Filter> element as "//Object[@NodeType='urn:liberty:ps:collection']."

The following is an example of a <QueryObjectsRequest> message with which the WSC requests to get all the <Object> elements that have urn:liberty:ps:entity as the value of NodeType element.

```xml
<QueryObjectsRequest>
  <Filter>//Object[@NodeType='urn:liberty:ps:entity']/Filter
</QueryObjectsRequest>
```

3.18.2. QueryObjectsResponse Message
A PS provider responds to a `<QueryObjectsRequest>` message with a `<QueryObjectsResponse>` message. The `<QueryObjectsResponse>` contains the `<Object>` elements that meet the criteria specified in the `<Filter>` element of the `<QueryObjectsRequest>` message.

The `<QueryObjectsResponse>` message has the complex type `QueryObjectsResponseType`, which extends `ResponseAbstractType` and adds the following element:

- `<Object> [Optional]` The `<Object>` element is used to convey data of zero or more `Objects` that the requested criteria are met to.

The schema declaration for the `<QueryObjectsResponse>` message is shown below.

```xml
<xs:element name="QueryObjectsResponse" type="QueryObjectsResponseType"/>
<xs:complexType name="QueryObjectsResponseType">
  <xs:complexContent>
    <xs:extension base="ResponseAbstractType">
      <xs:sequence>
        <xs:element ref="Object" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The following is an example of a `<QueryObjectsResponse>` to the `<QueryObjectsRequest>` message above.

```xml
<QueryObjectsResponse>
  <Status code="OK"/>
  <Object NodeType="urn:liberty:ps:entity">
    <ObjectID>https://ps.com/lsdjfojd</ObjectID>
    <DisplayName>Mary</DisplayName>
  </Object>
  <Object NodeType="urn:liberty:ps:entity">
    <ObjectID>https://ps.com/sijfsfsf</ObjectID>
    <DisplayName>Bob</DisplayName>
  </Object>
  <Object NodeType="urn:liberty:ps:entity">
    <ObjectID>https://ps.com/reremvls</ObjectID>
    <DisplayName>Nick</DisplayName>
  </Object>
  <Object NodeType="urn:liberty:ps:entity">
    <ObjectID>https://ps.com/soijfsfd</ObjectID>
    <DisplayName>JoJo</DisplayName>
  </Object>
  <Object NodeType="urn:liberty:ps:entity">
    <ObjectID>https://ps.com/sdosafms</ObjectID>
    <DisplayName>Taro</DisplayName>
  </Object>
  <Object NodeType="urn:liberty:ps:entity">
    <ObjectID>https://ps.com/lgsdfsfd</ObjectID>
    <DisplayName>Hanako</DisplayName>
  </Object>
</QueryObjectsResponse>
```

### 3.18.3. Processing Rules
• If a WSC specifies the value of the <Filter> element through an XPath expression, the value SHOULD begin with the expression "//Object".

• All the <Object> elements that are responded in the <QueryObjectsResponse> message from a PS provider MUST NOT contain any child <Object> and/or <ObjectRef> elements.

• If a PS provider cannot find any Objects that meets the criteria that a WSC specifies in the request, the PS provider MUST respond OK as the code attribute of the top level <Status> element, and the code attribute of the second level <Status> element MUST be set with the following status code:

• NoResults

3.19. Testing Membership

A WSC may wish to pose a question of the Object tree structure and have the results of that question returned rather than the Object tree itself (as the <ListMembersRequest> or <QueryObjectsRequest> messages support). For instance, the WSC might wish to ask whether a specified individual (or more generally any Object) is a member of a particular specified group Object. This scenario will be common when the owning user has defined some access control policy in terms of membership in some group (e.g. 'allow members of my soccer team to view these photos'). If and when some other user tries to access the resource in question, the WSC will need to determine if they are entitled (e.g. whether or not they are on the soccer team).

The <TestMembershipRequest> allows a WSC to pose the question 'Is user X a member of group Y?'

3.19.1. TestMembershipRequest Message

The target parent group Object for which the membership of another Object is being tested is indicated by the value of the <TargetID> element within the <TestMembershipRequest> message. The Object for which membership is being tested is specified by the <Token> element within the <TestMembershipRequest> message.

The <TestMembershipRequest> message has the complex type TestMembershipRequestType, which extends RequestAbstractType and adds the following elements:

<TargetID> [Required] The <TargetID> element is used to convey the ObjectID of the target group Object for which the membership of another user is being tested.

<Token> [Required] The <Token> element is used to convey an identity token of a user for which membership is being tested.

<Subscription> [Optional] The <Subscription> element is used to indicate to the PS provider that the WSC desires that the PS provider send a notification if and when results of the test changes.
The schema declaration for the `<TestMembershipRequest>` message is shown below.

```xml
<!-- Declaration of TestMembershipRequest element -->
<xs:element name="TestMembershipRequest" type="TestMembershipRequestType"/>
<!-- Definition of TestMembershipRequestType -->
<xs:complexType name="TestMembershipRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="TargetID"/>
        <xs:element ref="Token"/>
        <xs:element ref="Subscription" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The following is an example of a `<TestMembershipRequest>` message.

```xml
<TestMembershipRequest>
  <TargetID>https://ps.com/eiruvoie</TargetID>
  <Token/>
</TestMembershipRequest>
```

### 3.19.2. TestMembershipResponse Message

The PS returns the result of the specified membership test in a `<Result>` element within a `<TestMembershipResponse>` message.

The `<TestMembershipResponse>` message has the complex type `TestMembershipResponseType`, which extends `ResponseAbstractType` and adds the following elements:

- `<Result>` [Required] The `<Result>` element is used to convey the result of the specified membership test. This element has a type of `ResultType`, which is derived from `xs:boolean`.

The schema declarations for the `<TestMembershipResponse>` message and the `<Result>` element are shown below.

```xml
<!-- Declaration of ResultType -->
<xs:complexType name="ResultType">
  <xs:complexContent>
    <xs:extension base="xs:boolean"/>
  </xs:complexContent>
</xs:complexType>
<!-- Declaration of TestMembershipResponse element -->
<xs:element name="TestMembershipResponse" type="TestResponseType"/>
<!-- Definition of TestMembershipResponseType -->
<xs:complexType name="TestMembershipResponseType">
  <xs:complexContent>
    <xs:extension base="ResponseAbstractType">
      <xs:sequence>
        <xs:element name="Result" type="ResultType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
The following is an example of a `<TestMembershipResponse>` message.

```
<TestMembershipResponse>
  <Status code="OK"/>
  <Result>true</Result>
</TestMembershipResponse>
```

### 3.19.3. Processing Rules

If the `<TargetID>` element specifies the `ObjectID` of an `Object` with a `NodeType` attribute of "urn:liberty:ps:entity", the PS provider MUST respond `Failed` as the code attribute of the top level `<Status>` element, and the code attribute of the second level `<Status>` element MUST be set with the following status code:

- ObjectIsEntity

### 3.20. Resolving Objects

Once a WSC has an `ObjectID` for a user, it will often desire to communicate with other providers about that user. To do so, the WSC will first need an identity token for that user. The process of converting an object identifier into an identity token is referred to as resolving the identity token.

#### 3.20.1. ResolveIdentifierRequest Message

The WSC can use the `<ResolveIdentifierRequest>` message to ask the PS provider to resolve the specified `ObjectID` in the `<TargetID>` element, into an appropriate identity token.

The WSC MAY specify its requirements of the identity token through the `<TokenPolicy>`

The `<ResolveIdentifierRequest>` message has the complex type `ResolveIdentifierRequestType`, which extends `RequestAbstractType` and adds the following elements:

- `<TargetID>` [Required] The `<TargetID>` element is used to convey the `ObjectID` of the target group `Object` for which the WSC is requesting an identity token be resolved.

- `<TokenPolicy>` [Optional] The `<TokenPolicy>` element is used to convey any requirements the WSC may have of the resolved identity token.
The schema declaration for the `<ResolveIdentifierRequest>` message is shown below.

```
<!-- Declaration of ResolveIdentifierRequest element -->
<xs:element name="ResolveIdentifierRequest" type="ResolveIdentifierRequestType"/>
<!-- Definition of ResolveIdentifierRequestType -->
<xs:complexType name="ResolveIdentifierRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="TargetID"/>
        <xs:element ref="TokenPolicy" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The following is an example of a `<ResolveIdentifierRequest>` message in which the WSC is requesting that an identity token for the `Object` identified by the specified `ObjectID` be returned. The WSC is indicating that it desires a transient identifier for the identity token.

```
<ResolveIdentifierRequest>
  <TargetID>https://ps.com/lgsdfsfd</TargetID>
  <TokenPolicy>
    <samlp:NameIDPolicy Format="urn:oasis:names:tc:SAML:2.0:nameid-format: transient"/>
  </TokenPolicy>
</ResolveIdentifierRequest>
```

### 3.20.2. ResolveIdentifierResponse Message

A PS provider responds to a `<ResolveIdentifierRequest>` message with a `< ResolveIdentifierResponse>` element. The PS provider returns the identity token corresponding to the `Object` specified by the `ObjectID` in the `TargetID` element on the `<ResolveIdentifierRequest>` message.

The schema declaration for the `<ResolveIdentifierResponse>` message is shown below.

```
<!-- Declaration of ResolveIdentifierResponse element -->
<xs:element name="ResolveIdentifierResponse" type="ResolveIdentifierResponseType"/>
<!-- Definition of ResolveIdentifierResponseType -->
<xs:complexType name="ResolveIdentifierResponseType">
  <xs:complexContent>
    <xs:extension base="ResponseAbstractType">
      <xs:sequence>
        <xs:element ref="Token"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The following is an example of a `<ResolveIdentifierResponse>` to the `<ResolveIdentifierRequest>` message above.

```
<ResolveIdentifierResponse>
  <Status code="OK"/>
  <Token>
    <saml:Assertion/>
  </saml:Assertion>
</ResolveIdentifierResponse>
```
3.20.3. Processing Rules

Upon receiving a `<ResolveIdentifierRequest>` from a SP carrying a `<TargetID>` element that corresponds to an existing `Object`, the PS provider MUST endeavor to return to the SP an appropriate identity token corresponding to that `Object`. To do so, the PS provider MAY itself send a `<sa:TokenMap>` message to the relevant identity provider for the `Object` in question, specifying the requesting WSC as the target namespace.

If a PS provider can not resolve an appropriate identity token with the `ObjectID` specified by a SP, the PS provider MUST respond Failed as the code attribute of the top level `<Status>` element, and the code attribute of the second level `<Status>` element MUST be set with the following status code:

- `CannotResolveToken`
4. Interaction with Users

Before a user can be added to another user’s PS resource, appropriate federations may need to be established between various providers. Before this can happen, the user must first be prompted to visit one such provider in order to kick-off the process. The mechanism by which this prompt occurs is referred to here as an invitation.

Except for special cases where a user can be added to a user’s PS resource without the participation of that user (as would be possible if the PS resource owning user happened to know an identifier for the other user at an IDP), such an invitation is necessary. Supporting such user interactions is another aspect of the PS role.

4.1. Model (Informative)

The invitation model is as follows:

1. A user (either visiting one of their SP’s or their PS provider, decides that they wish to add some friend/contact/family member to their PS resource.

2. An invitation (consisting of some human readable descriptive text explaining the context as well as some mechanism by which interaction can be kicked off) is created by the provider on behalf of the PS resource owning user (the inviting user).

3. The invitation is delivered to the relevant user (the invited user).

4. The invited user examines the invitation and decides whether or not they wish to accept. If no, they take no further steps. If so, they proceed with the indicated mechanism to interact with the relevant providers (being given appropriate information and consent mechanisms at each step).

5. The invited user is added to the inviting user’s PS resource.

6. The invited user can be directed to the SP to access the resource in question.

4.2. Processing Rules

When the invited user responds to the invitation by clicking on the embedded URL, the SP MUST, after appropriately informing and ‘consenting’ the invited user, direct the user agent to the address previously returned within the <PSRedirectURL> element within the <AddEntityResponse> message. Once the invited user is at the PS provider, the PS provider:

1. MUST determine the invited user’s IDP

2. MUST attempt to federate the invited user with that IDP

3. MUST obtain an identity token from the IDP for the invited user targeted at the SP.

4. MUST, unless the SP did not include a <Subscription> in its <AddEntityRequest> message, forward on the identity token just received from the IDP in a <Notify> message, specifying the SubscriptionID of the previous <Subscription> element.

5. MUST redirect the invited user’s agent to the address previously specified by the <SPRedirectURL> element in the original <AddEntityRequest> message.

Once the invited user has been redirected to the <SPRedirectURL> address, the SP MAY choose to send a <samlp:AuthnRequest > message to the IDP asking for a <saml:AuthStatement> attesting to that user’s authentication status there. In its Response, the IDP MUST use the same subject identifier for this <saml:Assertion> as previously delivered to the SP through the PS provider and the <Notify> message.
5. Sequence Examples

Following are detailed sequence examples for:

- setting access control against a PS group
- checking group membership for access control
- performing a collective operation against group members

5.1. Policy definition

The following demonstrates examples of the messages exchanged when a user defines access control for some SP resource in terms of group membership:

1. Alice visits SPa and indicates that she wishes to allow a group of hers to view some resource there.

2. SPa discovers Alice’s PS.

3. SPa queries Alice’s PS for top-level Objects.

   <ListMembersRequest Structured="true"/>

4. PS responds with the Objects.

   <ListMembersResponse>
     <Status code="OK"/>
     <Object Node Type="urn:liberty:ps:entity">
       <ObjectID>https://ps.com/sdfhugusfsf</ObjectID>
       <DisplayName>Bob</DisplayName>
     </Object>
     <Object Node Type="urn:liberty:ps:entity">
       <ObjectID>https://ps.com/itndojoj</ObjectID>
       <DisplayName>Mary</DisplayName>
     </Object>
     <Object Node Type="urn:liberty:ps:collection">
       <ObjectID>https://ps.com/sijfsfsf</ObjectID>
       <DisplayName>Work Friends</DisplayName>
     </Object>
     <Object Node Type="urn:liberty:ps:collection">
       <ObjectID>https://ps.com/lstdjojof</ObjectID>
       <DisplayName>Soccer Team</DisplayName>
     </Object>
   </ListMembersResponse>

5. SPs displays the list to Alice.

6. Alice specifies that members of the group called ’Work Friends’ should be able to access the resource in question.

7. SPa defines appropriate permissions against the ’Work Friends’ group’s ObjectID of ’https://ps.com/sijfsfsf’. If and when somebody tries to access Alice’s resource in question, at that point SPa will need to determine if that individual is a member of the group Object with this ObjectID. See the following example in Section 5.2 for the sequences of messages.

   Rather than defining permissions against the ObjectID, the service provider could have chosen to obtain identity tokens (using a sequence of <ListMembersRequest> and <ResolveIdentifierRequest> messages) for all current members of the ’Work Friends’ group and then define access control rules directly against the relevant identifiers. This may not be appropriate if the membership of the group in question is expected to change.
5.2. AccessControl

The following is an example of the use-case in which an SP uses group membership information for controlling access to resources that it holds. In the use-case, Alice has defined access rules to some resources at SPa/WSCa based on membership in a group she maintains at PSa. Bob is a friend of Alice. When Bob appears at the SPa and tries to access the resource in question, the SPa must determine if Bob is a member of the group.

1. Bob shows up at SPa and tries to access the resource in question.
2. SPa asks 'Who are you?'.
3. Bob says 'Ask IDPb'.
4. SPa redirects Bob to IDPb with AuthnRequest.

<
<samlp:AuthnRequest
ID="NTT7630E00861279F0ADC63E241D0926D0B"
Version="2.0" IssueInstant="...">
  <saml:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
    https://spa.com
  </saml:Issuer>
  <samlp:NameIDPolicy Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"/>
</samlp:AuthnRequest>

5. IDPb authenticates Bob.
6. IDPb sends a Response to SPa with an AuthnStatement carrying a name identifier for Bob.

<
<samlp:Response
ID="NTT3F633E3F712BAC4B0804714431D46D7B"
InResponseTo="NTT7630E00861279F0ADC63E241D0926D0B"
Version="2.0" IssueInstant="...">
  <saml:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
    https://idpb.com
  </saml:Issuer>
  <samlp:Status>
  </samlp:Status>
  <saml:Assertion
Version="2.0" IssueInstant="..."
ID="NTT02062BBDE3E97EF0749828BCB8C15DFB"
Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
    <saml:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
      https://idpb.com
    </saml:Issuer>
    <saml:Subject
NameQualifier="https://idpb.com"
Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persitent">
      e0b735bf9d1f3959241d3584733d704c
    </saml:Subject>
    <saml:AuthnStatement
AuthnInstant="...
SessionIndex="...">
      <saml:AuthnContext>AuthnContext goes here</saml:AuthnContext>
    </saml:AuthnStatement>
  </saml:Assertion>
</samlp:Response>
7. SPa sends IDPb an `<sa:TokenRequest>`, providing the previous name identifier for Bob and specifying PSa as the target namespace.

```
<sa:TokenRequest>
  <saml:NameID
    NameQualifier="https://idpb.com"
    Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">
    e0b735bf9d1f3959241d3584733d704c
  </saml:NameID>
  <TokenPolicy SPNameQualifier="https://psa.com"/>
</sa:TokenRequest>
```

8. IDPb returns an appropriate mapped (and encrypted) identifier for Bob that PSa will recognize.

```
<sa:TokenResponse>
  <sec:Token>
    <saml:Assertion>
      identity token for Bob in PS’s namespace
    </saml:Assertion>
  </sec:Token>
</sa:TokenResponse>
```

9. As Alice has defined her access control rules in terms of a group maintained at PSa, SPa knows how to invoke PSa. SPa sends the PS provider query to PSa questioning Bob’s membership in the group in question.

```
<TestMembershipRequest>
  <TargetID>https://ps.com/nmerflas</TargetID>
  <Token>
    <saml:Assertion>
      identity token for Bob in PS’s namespace
    </saml:Assertion>
  </Token>
</TestMembershipRequest>
```

10. PSa extracts the identity token, might decrypt the encrypted identifier in the identity token, looks up the specified group, and finds Bob’s entry.

```
<TestMembershipResponse>
  <Status code="OK"/>
  <TestResult>true</TestResult>
</TestMembershipResponse>
```

11. PSa returns ‘true’ to SPa.

```
<TestMembershipResponse>
  <Status code="OK"/>
  <TestResult>true</TestResult>
</TestMembershipResponse>
```

12. Confident that Bob is a member of the group against which Alice defined privileges, SPa grants Bob access to the resource in question.

5.3. Group Operation

The following demonstrates the sequence of steps and messages involved when a user desires that some operation (e.g. send a party invited) to members of a particular group in her PS list.

1. Alice signs on to SPa.
2. Alice requests that SPa sends a party invited to all members in a group.

3. SPa/WSCa finds PSa, via DSa.

4. SPa/WSCa queries the list of available groups. Alice picks the relevant group. SPa/WSCa requests from PSa a list of members of the specified group.

   <ListMembersRequest>
   <TargetID>https://ps.com/nmerflas</TargetID>
   </ListMembersRequest>

5. PSa responds a list of members to SPa/WSCa.

   <ListMembersResponse>
   <Status code="OK"/>
   <Object NodeType="urn:liberty:ps:leaf">
   <ObjectID="https://ps.com/sijfsfsf"/>
   <DisplayName>Bob</DisplayName>
   </Object>
   <Object NodeType="urn:liberty:ps:leaf">
   <ObjectID="https://ps.com/lsdjfojd"/>
   <DisplayName>Mary</DisplayName>
   </Object>
   </ListMembersResponse>

6. SPa/WSCa sends a ResolveIdentifierRequest with appropriate TargetID element to PSa to request identity tokens for Bob & Mary.

   <ResolveIdentifierRequest>
   <TargetID>https://ps.com/sijfsfsf</TargetID>
   </ResolveIdentifierRequest>

7. PSa sends sa:TokenMap including the existing identity token between PSa and IDPb, to IDPb and specifies SPa/WSCa as the target provider.

   <sa:TokenMap>
   <sec:Token>
   <saml:Assertion>
   existing identity token for between PSa and IDPb
   </saml:Assertion>
   </sec:Token>
   <samlp:NameIDPolicy SPNameQualifier="https://spa.com"/>
   </sa:TokenMap>

8. IDPb sends an sa:TokenResponse message with an identity token that includes Bob’s federated (and maybe encrypted) name identifier between SPa/WSCa and IDPb.

   <sa:TokenResponse>
   <sa:Status>OK</Status>
   <sec:Token>
   <saml:Assertion>
   an identity token for Bob in SPa/WSCa’s namespace goes here
   </saml:Assertion>
   </sec:Token>
   </sa:TokenResponse>
9. PSa forwards on the identity token to SPa/WSCa in its ResolveIdentifierResponse message to the original ResolveIdentifierRequest message from SPa/WSCa.

<ResolveIdentifierResponse>
<Status code="OK"/>
<Token>
<saml:Assertion>
an identity token for Bob in SPa/WSCa’s namespace goes here
</saml:Assertion>
<Token>
</ResolveIdentifierResponse>

10. Once SPa/WSCa has the identity token, it is able to use the embedded extracts Endpoint Reference of Bob’s Discovery Service to discover Bob’s relevant WSPs, e.g. a Personal Profile service so as to get Bob’s email in order to send the party invited.
6. Security Considerations

A discussion of security considerations unique to the People Service and the user interaction model.
7. XML Schema for ID-WSF People Service

The formal XML schema for the ID-WSF People Service follows:

```xml
<xs:schema targetNamespace="urn:liberty:ps:2005-11"
            xmlns="urn:liberty:ps:2005-11"
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:sa="urn:liberty:sa:2005-11"
            xmlns:disco="urn:liberty:disco:2005-11"
            xmlns:sec="urn:liberty:sec:2005-11"
            xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
            xmlns:wsa="http://www.w3.org/2005/08/addressing"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified">
    <xs:include schemaLocation="liberty-idwsf-utility-v2.0.xsd"/>
    <xs:import namespace="urn:liberty:sa:2005-11"
                 schemaLocation="liberty-idwsf-authn-svc-v2.0.xsd"/>
    <xs:import namespace="urn:liberty:disco:2005-11"
                 schemaLocation="liberty-idwsf-disco-svc-v2.0.xsd"/>
    <xs:import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
                 schemaLocation="saml-schema-protocol-2.0.xsd"/>
    <xs:import namespace="http://www.w3.org/2005/08/addressing"
                 schemaLocation="ws-addr-1.0.xsd"/>
    <xs:import namespace="urn:liberty:sec:2005-11"
                 schemaLocation="liberty-idwsf-security-mechanisms-v2.0.xsd"/>
    <xs:annotation>
      <xs:documentation>
        The source code in this XSD file was excerpted verbatim from:
        Liberty ID-WSF People Service Specification
        Version 1.0-08 Draft
        Sept 19 2005
        Copyright (c) 2005 Liberty Alliance participants, see
        http://www.projectliberty.org/specs/idwsf_copyrights.html
      </xs:documentation>
    </xs:annotation>
    <!-- Definition of LocalizedDisplayNameType -->
    <xs:complexType name="LocalizedDisplayNameType">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="Locale" type="xs:language" use="required"/>
          <xs:attribute name="IsDefault" type="xs:boolean" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
    <!-- Definition of TagType -->
    <xs:complexType name="TagType">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="Ref" type="xs:anyURI" use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
    <!-- Declaration of ObjectID element -->
    <xs:element name="ObjectID" type="ObjectIDType"/>
    <!-- Declaration of TargetID element -->
    <xs:element name="TargetID" type="ObjectIDType"/>
</xs:schema>
```
<xs:complexType name="ObjectIDType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI"/>
  </xs:simpleContent>
</xs:complexType>

<!-- Declaration of Object element -->
<x:s:element name="Object" type="ObjectType"/>

<!-- Definition of ObjectType -->
<x:complexType name="ObjectType">
  <xs:sequence>
    <xs:element ref="ObjectID" minOccurs="0"/>
    <xs:element name="DisplayName" type="LocalizedDisplayNameType" minOccurs="1" maxOccurs="unbounded"/>
    <xs:element name="Tag" type="TagType" minOccurs="0"/>
    <xs:element ref="Object" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="ObjectRef" type="ObjectIDType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="NodeType" type="xs:anyURI" use="required"/>
  <xs:attribute name="CreatedDateTime" type="xs:dateTime" use="optional"/>
  <xs:attribute name="ModifiedDateTime" type="xs:dateTime" use="optional"/>
</xs:complexType>

<!-- Declaration of SPRedirectURL-->
<x:s:element name="SPRedirectURL" type="SPRedirectURLType"/>

<!-- Definition of SPRedirectURLType-->
<x:complexType name="SPRedirectURLType">
  <xs:annotation>
    <xs:documentation>
      When sending a AddEntityRequest to a PS provider, the SP may insert a SPRedirectURL. It will be to this URL that the invited principals will be sent after federating their IDP account to the PS provider.
    </xs:documentation>
  </xs:annotation>
  <xs:simpleContent>
    <xs:extension base="xs:anyURI"/>
  </xs:simpleContent>
</xs:complexType>

<!-- Declaration of PSRedirectURL-->
<x:s:element name="PSRedirectURL" type="PSRedirectURLType"/>

<!-- Definition of PSRedirectURLType-->
<x:complexType name="PSRedirectURLType">
  <xs:annotation>
    <xs:documentation>
      A PS provider may insert a PSRedirectURL in its AddEntityResponse. It will be to this URL that the invited principal will be sent after responding to the invitation.
    </xs:documentation>
  </xs:annotation>
  <xs:simpleContent>
    <!--<xs:restriction base="xs:anyURI"/>-->
    <xs:extension base="xs:anyURI"/>
  </xs:simpleContent>
</xs:complexType>

<!-- Declaration of CreatePSObject element -->
<xs:element name="CreatePSObject"/>

<!-- Declaration of TokenPolicy element -->
<xs:element name="TokenPolicy" type="TokenPolicyType"/>

<!-- Definition of TokenPolicyType -->
<xs:complexType name="TokenPolicyType">
  <xs:sequence>
    <xs:element ref="samlp:NameIDPolicy"/>
    <xs:any namespace="##other" processContents="lax"
      minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- Definition of RequestAbstractType -->
<xs:complexType name="RequestAbstractType" abstract="true">
  <xs:attribute name="id" type="xs:ID" use="required"/>
  <xs:attribute name="TimeStamp" type="xs:dateTime" use="required"/>
</xs:complexType>

<!-- Definition of ResponseAbstractType -->
<xs:complexType name="ResponseAbstractType" abstract="true">
  <xs:sequence>
    <xs:element ref="Status"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="required"/>
  <xs:attribute name="TimeStamp" type="xs:dateTime" use="required"/>
</xs:complexType>

<!-- Declaration of AddEntityRequest element -->
<xs:element name="AddEntityRequest" type="AddEntityRequestType"/>

<!-- Definition of AddEntityRequestType -->
<xs:complexType name="AddEntityRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="Object" minOccurs="0"/>
        <xs:element ref="SPRedirectURL" minOccurs="0"/>
        <xs:element ref="CreatePSObject" minOccurs="0"/>
        <xs:element ref="Subscription" minOccurs="0"/>
        <xs:element ref="TokenPolicy" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Declaration of AddKnownEntityRequest element -->
<xs:element name="AddKnownEntityRequest" type="AddKnownEntityRequestType"/>

<!-- Definition of AddKnownEntityRequestType -->
<xs:complexType name="AddKnownEntityRequestType">
  <xs:complexContent>
    <xs:extension base="ResponseAbstractType">
      <xs:sequence>
        <xs:element ref="Object" minOccurs="0"/>
        <xs:element ref="PSRedirectURL" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexContent>
  <xs:extension base="RequestAbstractType">
    <xs:sequence>
      <xs:element ref="Object"/>
      <xs:element ref="sec:Token"/>
      <xs:element ref="CreatePSObject" minOccurs="0"/>
      <xs:element ref="Subscription" minOccurs="0"/>
      <xs:element ref="TokenPolicy" minOccurs="0"/>
    </xs:sequence>
  </xs:extension>
</xs:complexContent>
<xs:complexType>
  <!-- Declaration of AddKnownEntityResponse element -->
  <xs:element name="AddKnownEntityResponse" type="AddKnownEntityResponseType"/>
  <!-- Definition of AddKnownEntityResponseType -->
  <xs:complexType name="AddKnownEntityResponseType">
    <xs:complexContent>
      <xs:extension base="ResponseAbstractType">
        <xs:sequence>
          <xs:element ref="Object" minOccurs="0"/>
          <xs:element ref="PSRedirectURL" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:complexType>
<xs:complexType>
  <!-- Declaration of AddCollectionRequest element -->
  <xs:element name="AddCollectionRequest" type="AddCollectionRequestType"/>
  <!-- Definition of AddCollectionRequestType -->
  <xs:complexType name="AddCollectionRequestType">
    <xs:complexContent>
      <xs:extension base="RequestAbstractType">
        <xs:sequence>
          <xs:element ref="Object"/>
          <xs:element ref="Subscription" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:complexType>
<xs:complexType>
  <!-- Declaration of AddCollectionResponse element -->
  <xs:element name="AddCollectionResponse" type="AddCollectionResponseType"/>
  <!-- Definition of AddCollectionResponseType -->
  <xs:complexType name="AddCollectionResponseType">
    <xs:complexContent>
      <xs:extension base="ResponseAbstractType">
        <xs:sequence>
          <xs:element ref="Object" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:complexType>
<xs:complexType>
  <!-- Declaration of AddToCollectionRequest element -->
  <xs:element name="AddToCollectionRequest" type="AddToCollectionRequestType"/>
  <!-- Definition of AddToCollectionRequestType -->
  <xs:complexType name="AddToCollectionRequestType">
    <xs:complexContent>
      <xs:extension base="RequestAbstractType">
        <xs:sequence>
          <xs:element ref="TargetID"/>
          <xs:element ref="ObjectID" minOccurs="1" maxOccurs="unbounded"/>
          <xs:element ref="Subscription" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:complexType>
<!-- Declaration of AddToCollectionResponse element -->
<xs:element name="AddToCollectionResponse" type="ResponseAbstractType"/>

<!-- Declaration of RemoveEntityRequest element -->
<xs:element name="RemoveEntityRequest" type="RemoveEntityRequestType"/>

<!-- Definition of RemoveEntityRequestType -->
<xs:complexType name="RemoveEntityRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="TargetID" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Declaration of RemoveEntityResponse element -->
<xs:element name="RemoveEntityResponse" type="ResponseAbstractType"/>

<!-- Declaration of RemoveCollectionRequest element -->
<xs:element name="RemoveCollectionRequest" type="RemoveCollectionRequestType"/>

<!-- Definition of RemoveCollectionRequestType -->
<xs:complexType name="RemoveCollectionRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="TargetID"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Declaration of RemoveCollectionResponse element -->
<xs:element name="RemoveCollectionResponse" type="ResponseAbstractType"/>

<!-- Declaration of RemoveFromCollectionRequest element -->
<xs:element name="RemoveFromCollectionRequest" type="RemoveFromCollectionRequestType"/>

<!-- Definition of RemoveFromCollectionRequestType -->
<xs:complexType name="RemoveFromCollectionRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="TargetID"/>
        <xs:element ref="ObjectID" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="Subscription" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Declaration of RemoveFromCollectionResponse element -->
<xs:element name="RemoveFromCollectionResponse" type="ResponseAbstractType"/>

<!-- Declaration of ListMembersRequest element -->
<xs:element name="ListMembersRequest" type="ListMembersRequestType"/>

<!-- Definition of ListMembersRequestType -->
<xs:complexType name="ListMembersRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="TargetID" minOccurs="0"/>
        <xs:element ref="Subscription" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="Structured" type="xs:boolean" default="true" use="optional"/>
      <xs:attribute name="Count" type="xs:nonNegativeInteger" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:attribute name="Offset" type="xs:nonNegativeInteger" default="0" use="optional"/>
</xs:extension>
</xs:complexType>

<!-- Declaration of ListMembersResponse element -->
<xs:element name="ListMembersResponse" type="ListMembersResponseType"/>

<!-- Definition of ListMembersResponseType -->
<xs:complexType name="ListMembersResponseType">
  <xs:complexContent>
    <xs:extension base="ResponseAbstractType">
      <xs:sequence>
        <xs:element ref="Object" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Declaration of QueryObjectsRequest element -->
<xs:element name="QueryObjectsRequest" type="QueryObjectsRequestType"/>

<!-- Definition of QueryObjectsRequestType -->
<xs:complexType name="QueryObjectsRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element name="Filter" type="xs:string"/>
        <xs:element ref="Subscription" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Declaration of QueryObjectsResponse element -->
<xs:element name="QueryObjectsResponse" type="QueryObjectsResponseType"/>

<!-- Definition of QueryObjectsResponseType -->
<xs:complexType name="QueryObjectsResponseType">
  <xs:complexContent>
    <xs:extension base="ResponseAbstractType">
      <xs:sequence>
        <xs:element ref="Object" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Declaration of GetObjectInfoRequest element -->
<xs:element name="GetObjectInfoRequest" type="GetObjectInfoRequestType"/>

<!-- Definition of GetObjectInfoRequestType -->
<xs:complexType name="GetObjectInfoRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="TargetID"/>
        <xs:element ref="Subscription" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Declaration of GetObjectInfoResponse element -->
<xs:element name="GetObjectInfoResponse" type="GetObjectInfoResponseType"/>

<!-- Definition of GetObjectInfoResponseType -->
<xs:complexType name="GetObjectInfoResponseType">
  <xs:complexContent>
    <xs:extension base="ResponseAbstractType">
      <xs:sequence>
        <xs:element ref="Object"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- Declaration of SetObjectInfoRequest element -->
<xs:element name="SetObjectInfoRequest" type="SetObjectInfoRequestType"/>

<!-- Definition of SetObjectInfoRequestType -->
<xs:complexType name="SetObjectInfoRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="Object" maxOccurs="unbounded"/>
        <xs:element ref="Subscription" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Declaration of SetObjectInfoResponse element -->
<xs:element name="SetObjectInfoResponse" type="ResponseAbstractType"/>

<!-- Declaration of TestMembershipRequest element -->
<xs:element name="TestMembershipRequest" type="TestMembershipRequestType"/>

<!-- Definition of TestMembershipRequestType -->
<xs:complexType name="TestMembershipRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="TargetID"/>
        <xs:element ref="ObjectID" maxOccurs="unbounded"/>
        <xs:element ref="Subscription" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Declaration of ResultType -->
<xs:complexType name="ResultType">
  <xs:simpleContent>
    <xs:extension base="xs:boolean"/>
  </xs:simpleContent>
</xs:complexType>

<!-- Declaration of TestMembershipResponse element -->
<xs:element name="TestMembershipResponse" type="TestMembershipResponseType"/>

<!-- Definition of TestMembershipResponseType -->
<xs:complexType name="TestMembershipResponseType">
  <xs:complexContent>
    <xs:extension base="ResponseAbstractType">
      <xs:sequence>
        <xs:element name="Result" type="ResultType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Declaration of ResolveIdentifierRequest element -->
<xs:element name="ResolveIdentifierRequest" type="ResolveIdentifierRequestType"/>

<!-- Definition of ResolveIdentifierRequestType -->
<xs:complexType name="ResolveIdentifierRequestType">
  <xs:complexContent>
    <xs:extension base="RequestAbstractType">
      <xs:sequence>
        <xs:element ref="TargetID"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexContent>
  <xs:extension base="ResponseAbstractType">
    <xs:sequence>
      <xs:element ref="sec:Token"/>
    </xs:sequence>
  </xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:element name="Subscription">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      <xs:attribute name="NotifyToRef" use="optional" type="xs:anyURI"/>
      <xs:attribute name="AdminNotifyToRef" use="optional" type="xs:anyURI"/>
      <xs:attribute name="Starts" use="optional" type="xs:dateTime"/>
      <xs:attribute name="Expires" use="optional" type="xs:dateTime"/>
      <xs:attribute name="id" use="optional" type="xs:ID"/>
      <xs:attribute name="SubscriptionID" use="required" type="IDType"/>
      <xs:attribute name="IncludeData" use="optional">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="Yes"/>
            <xs:enumeration value="No"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Notification">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Object" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="sec:Token" minOccurs="0" maxOccurs="unbounded"/>
      <xs:attribute name="Expires" use="optional" type="xs:dateTime"/>
      <xs:attribute name="id" use="optional" type="xs:ID"/>
      <xs:attribute name="SubscriptionID" use="required" type="IDType"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Notify">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="RequestAbstractType">
        <xs:sequence>
          <xs:element ref="Notification"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

<xs:element name="NotifyResponse" type="ResponseAbstractType"/>
</xs:schema>
8. Abstract WSDL

```xml
<definitions name="id-wsf-ps:2005-11:wsdl:interface"
  xmlns="http://schemas.xmlsoap.org/wsdl/
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ps="urn:liberty:ps:2005-11">
  <xsd:documentation>
    XML Schema from Liberty People Service Specification.
    *** NOTICE ***
    Copyright (c) 2004,2005 Liberty Alliance participants, see
    http://www.projectliberty.org/specs/idwsf_2_0_copyrights.php
  </xsd:documentation>
  <types>
    <xsd:schema>
      <xsd:import
        namespace="urn:liberty:ps:2005-11"
        schemaLocation="liberty-idwsf-ps.xsd"/>
    </xsd:schema>
  </types>
  <!-- Messages -->
  <!-- Adding a User -->
  <message name="AddEntityRequest">
    <part name="body" element="ps:AddEntityRequest"/>
  </message>
  <message name="AddEntityResponse">
    <part name="body" element="ps:AddEntityResponse"/>
  </message>
  <!-- Adding a Known User -->
  <message name="AddKnownEntityRequest">
    <part name="body" element="ps:AddKnownEntityRequest"/>
  </message>
  <message name="AddKnownEntityResponse">
    <part name="body" element="ps:AddKnownEntityResponse"/>
  </message>
  <!-- Removing a User -->
  <message name="RemoveEntityRequest">
    <part name="body" element="ps:RemoveEntityRequest"/>
  </message>
  <message name="RemoveEntityResponse">
    <part name="body" element="ps:RemoveEntityResponse"/>
  </message>
  <!-- Adding a Group -->
  <message name="AddCollectionRequest">
    <part name="body" element="ps:AddCollectionRequest"/>
  </message>
</definitions>
```
<message name="AddCollectionResponse">
  <part name="body" element="ps:AddCollectionResponse"/>
</message>

<!-- Removing a Group -->
<message name="RemoveCollectionRequest">
  <part name="body" element="ps:RemoveCollectionRequest"/>
</message>

<message name="RemoveCollectionResponse">
  <part name="body" element="ps:RemoveCollectionResponse"/>
</message>

<!-- Adding to a Group -->
<message name="AddToCollectionRequest">
  <part name="body" element="ps:AddToCollectionRequest"/>
</message>

<message name="AddToCollectionResponse">
  <part name="body" element="ps:AddToCollectionResponse"/>
</message>

<!-- Removing From a Group -->
<message name="RemoveFromCollectionRequest">
  <part name="body" element="ps:RemoveFromCollectionRequest"/>
</message>

<message name="RemoveFromCollectionResponse">
  <part name="body" element="ps:RemoveFromCollectionResponse"/>
</message>

<!-- Listing Members -->
<message name="ListMembersRequest">
  <part name="body" element="ps:ListMembersRequest"/>
</message>

<message name="ListMembersResponse">
  <part name="body" element="ps:ListMembersResponse"/>
</message>

<!-- Retrieving Object Info -->
<message name="GetObjectInfoRequest">
  <part name="body" element="ps:GetObjectInfoRequest"/>
</message>

<message name="GetObjectInfoResponse">
  <part name="body" element="ps:GetObjectInfoResponse"/>
</message>

<!-- Updating Object Info -->
<message name="SetObjectInfoRequest">
  <part name="body" element="ps:SetObjectInfoRequest"/>
</message>

<message name="SetObjectInfoResponse">
  <part name="body" element="ps:SetObjectInfoResponse"/>
</message>

<!-- Querying Objects -->
<!-- Querying Objects -->
<message name="QueryObjectsRequest">
  <part name="body" element="ps:QueryObjectsRequest"/>
</message>

<message name="QueryObjectsResponse">
  <part name="body" element="ps:QueryObjectsResponse"/>
</message>

<!-- Testing Membership -->
<message name="TestMembershipRequest">
  <part name="body" element="ps:TestMembershipRequest"/>
</message>

<message name="TestMembershipResponse">
  <part name="body" element="ps:TestMembershipResponse"/>
</message>

<!-- Resolving Identifiers -->
<message name="ResolveIdentifierRequest">
  <part name="body" element="ps:ResolveIdentifierRequest"/>
</message>

<message name="ResolveIdentifierResponse">
  <part name="body" element="ps:ResolveIdentifierResponse"/>
</message>

<!-- Port Type -->
<portType name="PeopleServiceSPort">
  <operation name="AddEntity">
    <input message="typens:AddEntityRequest"/>
    <output message="typens:AddEntityResponse"/>
  </operation>

  <operation name="AddKnownEntity">
    <input message="typens:AddKnownEntityRequest"/>
    <output message="typens:AddUserKnownResponse"/>
  </operation>

  <operation name="RemoveEntity">
    <input message="typens:RemoveEntityRequest"/>
    <output message="typens:RemoveEntityResponse"/>
  </operation>

  <operation name="AddCollection">
    <input message="typens:AddCollectionRequest"/>
    <output message="typens:AddCollectionResponse"/>
  </operation>

  <operation name="RemoveCollection">
    <input message="typens:RemoveCollectionRequest"/>
    <output message="typens:RemoveCollectionResponse"/>
  </operation>

  <operation name="AddToCollection">
    <input message="typens:AddToCollectionRequest"/>
    <output message="typens:AddToCollectionResponse"/>
  </operation>

  <operation name="RemoveFromCollection">
    <input message="typens:RemoveFromCollectionRequest"/>
    <output message="typens:RemoveFromCollectionResponse"/>
  </operation>
</portType>
<operation name="ListMembersOfCollection">
   <input message="typens:ListMembersRequest"/>
   <output message="typens:ListMembersResponse"/>
</operation>

<operation name="GetObjectInfo">
   <input message="typens:GetObjectInfoRequest"/>
   <output message="typens:GetObjectInfoResponse"/>
</operation>

<operation name="SetObjectInfo">
   <input message="typens:SetObjectInfoRequest"/>
   <output message="typens:SetObjectInfoResponse"/>
</operation>

<operation name="QueryObjects">
   <input message="typens:QueryObjectsRequest"/>
   <output message="typens:QueryObjectsResponse"/>
</operation>

<operation name="TestMembership">
   <input message="typens:TestMembershipRequest"/>
   <output message="typens:TestMembershipResponse"/>
</operation>

<operation name="ResolveIdentifier">
   <input message="typens:ResolveIdentifierRequest"/>
   <output message="typens:ResolveIdentifierResponse"/>
</operation>

</portType>
</definitions>
References

Normative


