HP Select Identity

Software Version: 4.20

Web Services Guide

Document Release Date: September 2007
Software Release Date: September 2007

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

Trademark Notices

UNIX® is a registered trademark of The Open Group.

The Select Identity product CD contains a `license` directory where you can find the license agreements for each of the third-party products used in this product.

Support

You can visit the HP software support web site at:

www.hp.com/go/hpsoftwaresupport

HP Software online support provides an efficient way to access interactive technical support tools. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels and HP Passport, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

1 Introduction

Provisioning and maintaining user accounts and system access in HP Select Identity is typically done via “out of the box” graphical user interface (GUI) features. When a system requires a customized method of provisioning, Select Identity web services provides a flexible substitute using an XML framework that uses the Service Provisioning Markup Language (SPML) developed by the Oasis Open organization’s Provisioning Services Technical Committee (PSTC) to facilitate data exchange among service provisioning systems.

Web services can be used to create request mechanisms that correspond to those provided in the Select Identity GUI, but which work in nonstandard settings, or with systems that require specialized configuration.

External systems can send Simple Object Access Protocol (SOAP) messages to Select Identity for user provisioning. For each request, web services, working on a Request/Response Paradigm, sends a response.

This guide discusses Select Identity web services within the overall context of Service Provider Markup Language (SPML). It provides details on how to develop properly formed requests and includes numerous example requests and fragments.

Audience

This guide is written for developers who are using web services to provision user management functionality. It is essential that you are familiar with the OASIS SPML 1.0 and 2.0 specifications and with XML in general.

Web services is a Provisioning Service Point (PSP), as defined by the OASIS SPML 1.0 and 2.0 specifications. Therefore, only element and attribute extensions used in web services are thoroughly documented in this guide. The OASIS SPML 1.0 and 2.0 specifications provide complete documentation of the standard elements and attributes, in addition to the concepts, frames of reference, and conventions established for SPML provisioning.

Refer to the SPML Overview for:

- More information about how Select Identity has implemented web services according to the SPML 1.0 and 2.0 standard.
- The URL of the OASIS SPML 1.0 and 2.0 specifications.

Functional Example

The following example illustrates how a web services request functions:

The Human Resources department at “Company X” relies on an enterprise resource planning (ERP) application to manage employees. When a new employee is hired, the HR department adds the employee to the system. However, the new hire will need email and network

accounts and access to the systems on which he will fulfill his job responsibilities. Select Identity web services can be used as a mechanism enabling the ERP application to send a request to provision the user. Then, Select Identity can create the necessary accounts and access privileges on Company X's systems according to the services defined for the user.

SPML Overview

Select Identity supports web services according to the OASIS SPML 1.0 and the 2.0 specifications, which define the concepts, operations, and XML schema for XML-based provisioning using a request and response paradigm.

The OASIS SPML 1.0 and 2.0 specifications, as the foundation for web services, are important reference documents if you are developing, configuring, or supporting web services on Select Identity. It is recommended that you download a copy of the specifications at the following URLs:

http://www.oasis-open.org/committees/download.php/3032/cs-pstc-spml-core-1.0.pdf

http://www.oasis-open.org/committees/download.php/17708/pstc-spml-2.0-os.zip

Workflow for SPML 2.0 is to discover the schema, make a request for a schema, and receive a response. Build the XML offline, and then validate the constructed XML using any validating editing tool.

Context SPML 1.0

Select Identity web services enforces all business processes on operations based on the context of each operation. Context is specified by the Requesting Authority (RA) that has initiated the request using the Select Identity application programming interface (API), and by the Provisioning Service Target (PST), which is the service on which the action will take place. When Select Identity receives a request (via an SPML-compliant message), the context of the request is determined as follows:

- Users or administrators can initiate requests such as user account creation, profile updates, password changes, addition of new service memberships, or stage completion in an approval process.
- Sensitive fields such as passwords can be encrypted using the security settings from the Select Identity configuration.

The following example illustrates how identifying information for the person originating the request is passed to the PST:

```
<operationalAttributes xmlns="">
   <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
      <value>sisa</value>
   </attr>
   <attr name='urn:trulogica:concero:2.0#password'>
      <value>abc123</value>
   </attr>
   <attr name="urn:trulogica:concero:2.0#keyFields">
      <value>UserName</value>
   </attr>
   <attr name="urn:trulogica:concero:2.0#serviceName">
      <value>CS1</value>
   </attr>
```

```
</operationalAttributes>
```

- The context of the PST is obtained from the following attribute in the request:

  `urn:trulogica:concero:2.0#serviceName`

- SPML 1.0 mandates use of the `requestID` attribute in requests and responses, so that each response can be matched with the corresponding request in asynchronous requests. Select Identity generates a unique internal ID for each request. It is not necessary for an externally-generated request ID to be unique, although it is good practice.

The `Keyfields` attribute, shown in the previous snippet, is used in reconciliation requests and is used to uniquely identify the user in Select Identity. The `ServiceName` attribute is optional; if it is not present, the request is a modify profile request, for modifying user profile attributes, as opposed to user service attributes.

Context SPML 2.0

Select Identity web services enforces all business processes on operations based upon the context of each operation. Context is specified by the RA that has initiated the request using the Select Identity API, and by the PST, which is the service on which the action will take place. When Select Identity receives a request (via an SPML-compliant message), the context of the request is determined as follows:

Users or administrators can initiate requests such as user account creation, profile updates, password changes, addition of new service memberships, or stage completion in an approval process.

The following snippet illustrates how identifying information for the person originating the request is passed to the PST:

```
<wsse:Security>
   <wsse:UsernameToken>
      <wsse:Username>sisa</wsse:Username>
      <wsse:Password Type=”wsse:PasswordText”>abc123</wsse:Password>
      <wsse:Nonce>WScqanjCEAC4moBE07sAQ__</wsse:Nonce>
      <wsu:Created>2003-07-16T01:24:32/wsu:Created>
   </wsse:UsernameToken>
</wsse:Security>
```

- The context of the PST is obtained from the `targetID` attribute in the request:

  `<addRequest targetID=”SpmlTest”>`

- SPML 2.0 mandates use of the `requestID` attribute in requests and responses, so that each response can be matched with the corresponding request in asynchronous requests. If an asynchronous request to Select Identity doesn't contain a `requestId`, Select Identity generates one and returns it in the response. It is not necessary for an externally-generated request ID to be unique, although it is good practice.

Encoding

All SPML files that contain requests sent to web services must be properly formed and valid XML, and must conform to the SOAP protocol.

Standard and Extended Requests for SPML 1.0

The SPML 1.0 standard allows operational elements to be used in their original form or an extended form. SPML 1.0 Structure and Elements on page 29 provides details of the elements that are supported in Select Identity in standard and extended form.

Extended requests are used to perform the following Select Identity operations:

- Terminate user
- Change password
- Reset password
- Enable user
- Disable user
- Enable user on service
- Disable user on service
- Transfer account

Core, Suspend, Password, and Asynchronous Capabilities in SPML 2.0

The SPML 2.0 standard separates operational elements into core capabilities that must be supported under SPML 2.0, and suspend, password, and asynchronous capabilities that may or may not be supported. SPML 2.0 Structure and Elements on page 45 provides details of the elements that are supported in Select Identity.

Core requests perform the following operations:

- Discover targets
- Add a user
- Delete a user
- Modify a user profile
- Lookup and display a user's service attributes

Suspend requests perform the following operations:

- Disable a user or user's service membership
- Enable a user or user's service membership
- Display the status (enable/disable) of a user

Password requests perform the following operations in Select Identity or in a service:

- Set a user’s password
- Reset a user’s password

Asynchronous requests perform the following operations:

- Cancel a pending asynchronous request
- Display the status (success/failure/pending) of a given asynchronous request

Individual and Batch Requests SPML 1.0

Many requests submitted via web services are for individual delegated or self-service operations, such as profile modifications, user service membership additions or deletions, user resource account additions or deletions, or service account attribute modifications.

Web services allows multiple requests to be processed as a group via batch request processing. Any user provisioning operation in Select Identity can be performed using this method. A single batch request consists of a series of requests that are all of the same type.

Batch processing uses the Hypertext Transport Protocol (HTTP) as its transfer mechanism. In general, this type of processing is more suitable for smaller numbers of requests (less than 1000). For large numbers of requests, Select Identity's file based reconciliation or bulk capabilities are more suitable.

The following web services batch request types are supported in Select Identity with SPML 1.0:

- Add
- Modify
- Delete

Batch operations are *not* available by SPML 2.0.

Primary and Secondary Account Clusters

Select Identity supports additional, "secondary," user identities associated with a single, "primary" user account. This grouping of a primary user ID and one or more secondary IDs is known as a user cluster. The first time a secondary account is created for a primary user ID, Select Identity creates a user account cluster.

Transfer accounts are *not* supported by SMPL 2.0.

Sample Requests

The Select Identity product CD contains a `samples` directory that contains several example web service requests. These examples can serve as a basis for developing and implementing requests on a Select Identity system.

SPML 2.0 Namespaces

The following SPML 2.0 namespaces are used in this release:

`urn:oasis:names:tc:SPML:2.0`

`urn:oasis:names:tc:SPML:2.0:suspend`

`urn:oasis:names:tc:SPML:2.0:password`

`urn:oasis:names:tc:SPML:2.0:async`

The following namespace defines Select Identity specific schema:

`urn:hp:si:SPML20:common`

2 Operational Summary

This section summarizes the operations supported in web services SPML, and provides structural information about typical requests. It also specifies which Select Identity operations are *not* supported in web services.

User Provisioning

Most user management operations available in Select Identity can be performed using web services. Both self-service and delegated requests are supported.

The following are the supported operations for Select Identity using SPML 1.0:

- Add a user (`addRequest`)
- Add a user on a composite service (`addRequest`)
- Enable a user account (`extendedRequest`)
- Disable a user account (`extendedRequest`)
- Enable or disable service membership (`extendedRequest`)
- Modify user attributes (except passwords and context) and entitlements (`modifyRequest`)
- Delete a user from a service (`deleteRequest`)
- Retrieve information about one or more users (`searchRequest`)
- Reset or change a user's password (`extendedRequest`)
- Terminate a user (`extendedRequest`)
- Batch requests of similar request types are supported for add, modify, and delete (terminate) only.

When retrieving a user (`searchRequest`), either a list of users or various details, including resource accounts with entitlements, can be retrieved for individual users. See Chapter 4, SPML 1.0 Structure and Elements, for more details on SPML 1.0.

The following are the supported operations for Select Identity using SPML 2.0:

- Add a user in Select Identity (`addRequest`)
- Add a user in a service (`addRequest`)
- Enable a user account in Select Identity (`resumeRequest`)
- Disable a user account in Select Identity (`suspendRequest`)
- Enable or disable service membership (`suspendRequest`, `resumeRequest`)
- Modify user attributes (except passwords and context) and entitlements (`modifyRequest`)

- Delete a user from a service (`deleteRequest`)
- Retrieve a user's service information (`lookupRequest`)
- Retrieve a user's profile information (`lookupRequest`)
- Retrieve information about the status of a user in a service (`activeRequest`)
- Retrieve information about the status of a user in Select Identity (`activeRequest`)
- Reset or change a user's password (`setPasswordRequest`, `resetPasswordRequest`)
- Terminate a user (`deleteRequest`)

Operations that are Not Supported

Not all operations are supported through web services using SPML. The following operations are not supported in web services using SPML 1.0:

- Search operations for services, resources, or any object other than user accounts
- Batch requests of mixed request types
- Operations other than user account management (such as resource, service, context, or role creation and deletion)
- Terminating a primary user account *without* also terminating all secondary accounts (termination of the primary account automatically terminates all secondaries)
- Schema discovery operations

SPML 2.0 does not provide support for the following operations in Web Service:

- Batch operations
- Bulk operations
- Reconciliation operations
- Transfer accounts (multi-user accounts)

Anatomy of a Web Services Request with SPML 1.0

Web services requests are structured as explained in this section, by using a simple `addRequest` example (adding a user account) to illustrate the main sections of each request.

SOAP Envelope

Each SPML 1.0 request is contained within a SOAP envelope, typically sent over HTTP, as follows:

```
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
   <soap:Body>
```

Request Type and Identifier

The request type and accompanying identifier attribute follow the SOAP envelope declaration:

```
<addRequest requestID='12345' execution='urn:oasis:names:tc:
SPML:1:0#asynchronous'>
```

The request type and its identifier follow the SOAP envelope declaration, SPML 1.0:

```
<addRequest requestID='12345' execution='asynchronous'>
```

The identifier value is returned in the `requestResponse` and is therefore important for tracking and auditing requests. Request IDs are normally numeric.

Operational Attributes

Within the body of an SPML 1.0 request, the operational attributes are provided first. These attributes specify identifying information about the user originating the request, including the user name, password, and the services to which the request is targeted (`Default`, `Global`, `USA`, and `Texas` in the following snippet). Refer to SPML 1.0 Structure and Elements on page 29 for further information.

For an `addRequest`, the specified `serviceName` values are the services on which the user is being added. For a `modifyRequest`, these values are the services to which the attributes being modified or deleted belong. For a `deleteRequest`, the `serviceName` values are the services from which the user is being deleted.

The following example illustrates the operational attributes section of a web services request:

```
<operationalAttributes>
   <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
      <value>sisa</value>
   </attr>
   <attr name='urn:trulogica:concero:2.0#password'>
      <value>abc123</value>
   </attr>
   <attr name='urn:trulogica:concero:2.0#serviceName'>
      <value>Default</value>
      <value>Global</value>
      <value>USA</value>
      <value>Texas</value>
   </attr>
</operationalAttributes>
```

Security with SPML 1.0

Select Identity provides the same level of security for web services as for the browser GUI. Thus, Select Identity supports HTTP through Secure Sockets Layer (SSL) 3.0 or Transport Layer Security (TLS) 1.0.

To enable external authentication of requests, you must modify properties in the `TruAccess.properties` file. Refer to the *HP Select Identity Installation Guide* for details. Appendix B Configuring TruAccess.properties details all the properties in this file, in addition to Chapter 6, Configuring HP OpenView Select Identity which discusses Select Identity security framework and keystores.

Authentication and Authorization with SPML 1.0

Web services uses the Select Identity authentication mechanism. To authenticate requests, web services support basic user authentication via the Select Identity user name and password that must be included in the request in an `operationalAttributes>` element.

Before any operation is completed, the user name and password is verified. For authorization, web services allows or denies the request according the roles assigned to the user. The user can only perform operations according to the permissions set within the assigned roles.

External authentication can be achieved through use of a verification external call assigned to the `Password` attribute. The *HP Select Identity External Call Developer Guide* provides more detailed and comprehensive information for developers to use in coding custom external calls. Refer to the *HP Select Identity Administration Online Help*, which includes information about registering and using external calls.

Resource Identification For Reconciliation Requests

Web Service reconciliation requests contain a resource identifier to indicate the origin of the request.

An initial user creation request for a primary user must come from an authoritative resource. Once the user is created in Select Identity, resource specific entitlements can be added from non-authoritative resources. Secondary users may be added from non-authoritative resources.

For further information about reconciliation, refer to the *HP Select Identity Administration Guide* and the *HP Select Identity Administration Online Help*.

Request Attributes

The data to be used in the request follows the operational attributes as a series of individual `attributes`, such as the user's first and last names, email account, and password. Each attribute is bounded by the `<attr>` and `</attr>` tag.

All attributes, including user name, password, and entitlements, can be autogenerated by Select Identity.

The following example shows identity and profile information for a user named `AAllen`, together with entitlements for a resource named `LDAP`.

```
<attributes>
   <attr name='UserName'>
      <value>AAllen</value>
   </attr>
   <attr name='Password'>
      <value>abcd1234</value>
   </attr>
   <attr name='FirstName'>
      <value>Anna</value>
   </attr>
   <attr name='LastName'>
      <value>Allen</value>
   </attr>
   <attr name='Email'>
      <value>anna.allen@companyx.com</value>
   </attr>
   <attr name='urn:trulogica:concero:2.0#groups:LDAP'>
      <value>Group 1</value>
      <value>Group 2</value>
   </attr>
```

```
      <attr name='Company'>
         <value>Company X</value>
      </attr>
      <attr name="Department">
         <value>Sales</value>
      </attr>
      <attr name='City'>
         <value>Dallas</value>
      </attr>
      <attr name='State'>
         <value>Texas</value>
      </attr>
      <attr name='Country'>
         <value>USA</value>
      </attr>
      <attr name='BirthDate'>
         <value>08/09/1980</value>
      </attr>
      <attr name="Zip"/>
   </attributes>
```

Request Closure

At the end of the attributes, the request is closed, as are the request body and SOAP envelope, as follows:

```
      </addRequest>
   </soap:Body>
</soap:Envelope>
```

Responses

SPML uses a *request-response paradigm.* For every type of request there exists a correspondingly-named response. Responses provide confirmation if the originating request is successful. They also provide information about the cause of failure; this can be interpreted to correct the source of the problem. In addition, a `searchRequest` returns records that match the search criteria by enclosing them in a `searchResponse`.

Delegated and Self Service Requests

Delegated requests are performed for a user's account by a system administrator. In the case of a delegated request, the user account from which the request is made must have an appropriate level of administration privileges to allow the request.

Self-service requests are submitted by a user on his or her own behalf. Self-service requests are identical to delegated requests in every regard except for the access level and permissions granted to the user. Self-service accounts normally have a lower level of access and administration privileges than does an administrator or a user who makes delegated requests.

Reconciliation

You can set up web services to allow a user's attributes to be changed on a resource and then synchronized to propagate the change into the Select Identity repository.

Reconciliation, as it is called, enables a resource to push user information to Select Identity. The agent on the resource (provided by a two-way connector) tracks changes made on the resource and synchronizes them with the data in Select Identity.

All attributes changed in Select Identity are pushed out to resources according to the attribute mapping that has been defined. For detailed information about mapping attributes, see the *HP Select Identity Administration Online Help*, including the topics about the Attribute Mapper utility.

For additional information about reconciliation and authoritative versus non-authoritative resources, refer to the appropriate sections in the *HP Select Identity Administration Guide* and the *HP Select Identity Administration Online Help*.

Reconciliation Rules

Rules identify the services that are affected when a reconciliation operation is issued, based on the resource ID sent by the resource. Rules also specify the location of the reverse mapping that should be performed for add and modify operations, such as:

```
NT Domain Resource -> ntuser.properties.
```

Thus, reverse synchronization can be performed only for services with specified mapping for the incoming resource ID. If the mapping does not exist, the request is logged and ignored.

Reconciliation Operations in Web Services

The following reconciliation operations can be performed using web services, either individually or as a batch under SPML 1.0:

- Add a user, if the resource is authoritative (`<addRequest>`)
- Add attributes to an existing user, if the resource is non-authoritative (`<addRequest>` or `<modifyRequest>`)
- Modify user attributes and entitlements (`<modifyRequest>`)
- Delete a user (`<deleteRequest>`)
- Issue multiple add, modify, and/or delete requests (`<batchRequest>`)

Unlike batch requests, reconciliation requests can contain mixed request types; they are not limited to a single type of request.

Anatomy of a Web Services Request with SPML 2.0

Web services requests are structured as explained in this section, which uses a simple `addRequest` example (adding a user account) to illustrate the main sections of each request.

SOAP Envelope with SPML 2.0

Each SPML 2.0 request is contained within a SOAP envelope, typically sent over HTTPS, as follows:

```
<soap:Envelope xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://
docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:soap="http://
schemas.xmlsoap.org/soap/envelope/">

   <soap:Header>
      <wsse:Security>
      <!-- Enter admin username, password here -->
         <wsse:UsernameToken>
            <wsse:Username>spml20admin</wsse:Username>
            <wsse:Password Type="wsse:PasswordText">PASSWORD
            </wsse:Password>
            <wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
            <wsu:Created>2003-07-16T01:24:32Z</wsu:Created>
         </wsse:UsernameToken>
      </wsse:Security>

   <serviceHeader>
      <!-- indicates SPML 2.0 request -->
      <serviceId>spml20</serviceId>
   </serviceHeader>

   </soap:Header>

   <soap:Body>

      <spml2:addRequest xmlns:spml2="urn:oasis:names:tc:SPML:2:0"
      xmlns:hpsi="urn:hp:si:spml20:common" xmlns:xsi="http://www.w3.org/
      2001/XMLSchema-instance" targetID="Service:spml20 test"
      xmlns="urn:hp:si:spml20:spml20 test" hpsi:serviceCallId="2001">
         <spml2:psoID ID="jamessmith"></spml2:psoID>
         <spml2:data>
         <!-- Attributes within data element are based on
         DelagatedAddEvent from Business service XSD schema in the default
         namespace -->
            <CompanyDelegatedAddUserDefaultf020View>
            <Company>HP</Company>
            <State>TX</State>
            <spml20_Email>james.smith@hp.com</spml20_Email>
            <!-- Multi-Valued -->
            <spml20auth_ENTITLEMENTS>$UNIX1</spml20auth_ENTITLEMENTS>
            <spml20auth_ENTITLEMENTS>$UNIX2</spml20auth_ENTITLEMENTS>
            <spml20_FirstName>James</spml20_FirstName>
            <LastName>Smith</LastName>
            <Password>PASSWORD</Password>
             <!-- attribute name with special character; XML element name
            cannot contain special characters -->
            <spml20_Employee0020Id mappedSIattrName="spml20_Employee
            Id">12345</spml20_Employee0020Id>
            <UserName>jamessmith</UserName>
            <OVSIDateOfBirth>2002-10-10T00:00:00</OVSIDateOfBirth>
            </CompanyDelegatedAddUserDefaultf020View>
```

```
            </spml2:data>
         </spml2:addRequest>
      </soap:Body>
   </soap:Envelope>
```

Security with SPML 2.0

Select Identity provides the same level of security for web services as for the browser GUI. Thus, Select Identity supports HTTP through SSL 3.0 or TLS 1.0.

Select Identity does not support encryption or decryption of sensitive attributes sent to the SPML 2.0 SOAP port. All SPML 2.0 requests *are required* to be sent using HTTP or HTTPS.

Select Identity mandates using HTTP or HTTPS connections for sending SPML 2.0 requests. Encryption of Select Identity sensitive information is handled in SPML 2.0 entirely through HTTPS.

Namespace declarations for security elements are made as part of the SOAP Envelope element in SPML 2.0.

Authentication and Authorization with SPML 2.0

Web services uses the Select Identity authentication mechanism. To authenticate requests, web services support basic user authentication via the Select Identity user name and password that must be included in the request in an `wsse:Security>` element nested within the `soap:Header` element.

Before any operation is completed, the user name and password is verified. For authorization, web services allows or denies the request according to the roles assigned to the requestor identified in the `soap:Header`. The user can only perform operations according to the permissions set within the assigned roles. The following snippet shows an authentication.

```
<soap:Header>
   <wsse:Security>
      <wsse:UsernameToken>
         <wsse:Username>sysAdmin</wsse:Username>
         <wsse:Password Type=”wsse:PasswordText”>pw1234
         </wsse:Password>
         <wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
         <wsu:Created>2003-07-16T01:24:32Z</wsu:Created>
      </wsse:UsernameToken>
   </wsse:Security>
</soap:Header>
```

As shown in the previous snippet, the `UsernameToken` element contains the elements `Username` and `Password` for the authenticating user.

Service Header

The `serviceHeader` element is found within in the soap:Header element and defines the version of SPML that is used.

Body

Within the body of the SOAP envelope is the request where the attributes and sub-elements are provided. These sub-elements specify identifying information about the user who is being provisioned in Select Identity by the request, including the user name, password, and the entitlements to which the request is targeted.

For an `addRequest`, the specified service name value, found in the `targetID` attribute of the request, is the service on which the user is being added. For a `modifyRequest`, these values are the services to which the attributes being modified or deleted belong. For a `deleteRequest`, the service name values are the services from which the user is being deleted.

```
<soap:body>
   <spml2:addRequest ... targetID="Service:spml20 test"
   xmlns="urn:hp:si:spml20:spml20 test">
      <spml2:psoID ID="moduser1"></spml2:psoID>
      <spml2:data>
      ...
      </spml2:data>
   </spm2:addrequest>
</soap:body>
```

Request Type and Identifier

The request type and its identifier follow the SOAP body declaration:

```
<spml2:addRequest xmlns:spml2="urn:oasis:names:tc:SPML:2:0"
xmlns:hpsi="urn:hp:si:spml20:common"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
targetID="Service:spml20 test" xmlns="urn:hp:si:spml20:spml20
test">
```

The `requestID` value is returned in the request response and is therefore important for tracking and auditing requests. The `requestID` in SPML 2.0 requests is alphanumeric and must start with an alpha character. All asynchronous requests to Select Identity require a `requestID` for correlating responses. If a `requestID` is not provided in the asynchronous request, Select Identity generates one and returns it in the response.

Request Elements

The data used in the request are contained within the request element as individual elements, such as the user's first and last names, email account, and user name.

The following snippet shows a request to add a user in a service. The `psoID` element identifies the user being provisioned in Select Identity. The data element contains the user's service attribute values. Service attributes are specified based upon specific service schema. The data element contains the fields from the service XSD obtained using `listTargetsRequest`, which defines the elements and their format.

```
<spml2:psoID ID="jamessmith"></spml2:psoID>
   <spml2:data>
      <!-- Attributes within data element are based on DelagatedAddEvent
      from Business service XSD schema in the default namespace -->
      <CompanyDelegatedAddUserDefaultf020View>
         <Company>HP</Company>
         <State>TX</State>
```

```
            <spml20_Email>james.smith@hp.com</spml20_Email>
            <!-- Multi-Valued -->
            <spml20auth_ENTITLEMENTS>$UNIX1</spml20auth_ENTITLEMENTS>
            <spml20auth_ENTITLEMENTS>$UNIX2</spml20auth_ENTITLEMENTS>
            <spml20_FirstName>James</spml20_FirstName>
            <LastName>Smith</LastName>
            <Password>PASSWORD</Password>
            <!-- attribute name with special character; XML element name
            cannot contain special characters -->
            <spml20_Employee0020Id mappedSIattrName="spml20_Employee
            Id">12345</spml20_Employee0020Id>
            <UserName>jamessmith</UserName>
            <OVSIDateOfBirth>2002-10-10T00:00:00</OVSIDateOfBirth>
        </CompanyDelegatedAddUserDefaultf020View>
    </spml2:data>
```

Request Closure

At the end of the attributes, the request is closed, as are the request body and SOAP envelope, as follows:

```
        </addRequest>
    </soap:Body>
</soap:Envelope>
```

Responses

SPML uses a *request-response paradigm.* For every type of request there exists a correspondingly-named response. Responses provide confirmation if the originating request is successful. They also provide information about the cause of failure; this can be interpreted to correct the source of the problem.

Delegated and Self-service Requests

Delegated requests are performed for a user's account by a system administrator. In the case of a delegated request, the user account from which the request is made must have an appropriate level of administration privileges to allow the request.

Self-service requests are submitted by a user on his or her own behalf. Self-service requests are identical to delegated requests in every regard except for the access level and permissions granted to the user. Self-service accounts normally have a lower level of access and administration privileges than does an administrator or a user who makes delegated requests.

Web Services and Multiple Resource IDs

This section describes web services user management operations in the context of the Select Identity multiple resource ID feature.

In Select Identity, a user is normally identified by a single user name. However, in some cases, it is desirable for a user to have several user names. For example, one might be the person's general-purpose user name and another might be a task and/or resource-specific user name. Multiple user IDs are organized so that one is the "primary" user name and all others are "secondary" user names. The person who owns these user names and accounts logs into Select Identity with the primary ID.

Delegated User Management and Multiple Resource IDs

The following operations are supported for multiple resource IDs in self-service and delegated modes:

- Add secondary user to service and associate with a primary user

  When a secondary user account is created, the `operationalAttributes>` must include the primary user key for validation. The primary user key is not needed for other operations, as it is available to Select Identity when retrieved from the database.
- Modify secondary user in a service
- Delete secondary user from service
- Enable secondary user in a service
- Disable secondary user in a service
- Enable secondary user
- Disable secondary user
- Terminate secondary user
- Transfer accounts from one primary account to another
- Reset or change user password

Multiple Resource IDs

Web services provides the capability to:

- Link multiple resource accounts to one user account in a cluster consisting of one primary user account and one or more secondary accounts during reconciliation add or modify requests.
- Synchronize Select Identity user attributes across multiple accounts within a cluster based on the user's auto synchronization control flag settings in the reconciliation data file.

Password Management and Multiple Resource IDs

Use web services to change the password for supplied user accounts in a similar fashion to self-service password changes. The user account for login *must* be the primary account, and only this account password is verified. The password change can be either for the primary account or for a specified secondary account.

The `changePassword` operation is designed for self-service (and for primary users to change a secondary user's password). The `resetPassword` operation is only used in delegated password reset operations.

Password change requests under SPML 1.0 are performed using the `extendedRequest` element.

Password change requests under SPML 2.0 are performed using the password capability namespace `urn:oasis:names:tc:SPML:2:0:password`. The password capability elements include the `requestID>` attribute and the `psoID>` element with the `targetID` attribute, as in the following snippet:

```
<soap:Body>
   <spmlpwd:resetPasswordRequest
   xmlns:spml="urn:oasis:names:tc:SPML:2:0"
   xmlns:spmlpwd="urn:oasis:names:tc:SPML:2:0:password"
   xmlns:hp="urn:hp:si:spml20:common"
   xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
   requestID="R1004" executionMode="asynchronous"
   hp:serviceCallId="2001">
      <spmlpwd:psoID ID="resetPassword1"
      targetID="Attribute:Password"></spmlpwd:psoID>
   </spmlpwd:resetPasswordRequest>
</soap:Body>
```

3 Issuing Requests

To issue web services user provisioning and data synchronization requests to Select Identity, an external system must send a SOAP message containing an SPML request.

URL for Sending Requests

All web services requests in SPML 1.0 must be sent to the following URL:

http://*select_identity_host:port*/lmz/webservice/

All web services requests in SPML 2.0 must be sent to the following URL:

http://*localhost:7001*/axis2/rest/ProcessDocumentWebService/processRequest

The section of the URL in italic type corresponds to the usual address or hostname where you access Select Identity. Refer to the *HP Select Identity Administration Guide* for further information.

4 SPML 1.0 Structure and Elements

This section provides detailed discussion of the supported request elements and their attributes, illustrated with a snippet of SPML.

The HP Select Identity product CD contains a `samples` directory, where you will find a set of web services SPML request examples. These are provided so that you can use them as a basis for developing custom requests.

<addRequest>

Select Identity supports several variations of the `addRequest` element. In any `addRequest`, if the user name already exists in Select Identity, then other specified attributes, such as service memberships, are updated. An `addRequest` only adds a user account if the specified account does not already exist in Select Identity. Since a `modifyRequest` only allows modification of profile attributes, the `addRequest` is used whenever the objective is to add a user account to a given service.

The following is a partial example of an `addRequest`. The `<operationalAttributes>` provide login credentials for the administrator or self-service user originating the request. If you add one or more services to the `serviceName` operational attribute (using a value that corresponds to one of the services in your Select Identity system), the `addRequest` subscribes the user to that service as well. The `attributes` element provides information about the user. These attributes are service form specific, because a given form may not have all service attributes; an attribute cannot be accessed if it is not in the service form for the request type. The `requestID` identifies the provisioning transaction in the `requestResponse`.

```
<addRequest ... requestID=12345>
```

General Use Attributes

The following attributes are used in all instances of an `addRequest` and in reconciliation operations.

```
<operationalAttributes>
  <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
    <value>sysAdmin</value>
  </attr>
  <attr name='urn:trulogica:concero:2.0#password'>
    <value>pw1234</value>
  </attr>
</operationalAttributes>
```

The `UserIDAndOrDomainName` and the `password` elements are the username and password of the administrator authorized to add a user to the specified service, or the user name and password of a self-service user. Select Identity authenticates this user against the `serviceName` attribute.

You can provision a user onto more than one service by placing multiple values in the `serviceName` operational attribute.

The following are name-value pairs of the attributes to add for the user, where `attr_name` is the name of the attribute and `value` is the value. You must provide an `attr` element for each attribute required by the resource(s) for a newly created user. Do not specify attributes that are autogenerated.

```
<attributes>
   <attr name='UserName'>
      <value>wsuser1</value>
   </attr>
   <attr name='Password'>
      <value>abcd1234</value>
   </attr>
   <attr name='FirstName'>
      <value>Anna</value>
   </attr>
   <attr name='LastName'>
      <value>Allen</value>
   </attr>
   <attr name='Email'>
      <value>user@companyx.com</value>
   </attr>
   <attr name='urn:trulogica:concero:2.0#groups:LDAP177'>
      <value>West Oregon</value>
      <value>West Washington</value>
   </attr>
   <attr name='Company'>
      <value>HP</value>
   </attr>
</attributes>
```

To add a user with a system-generated username and password provide the `FirstName` and `LastName` values. The absence of the `taUserName` in the request triggers the auto-generation of a user name and password by Select Identity.

```
   <attr name='FirstName'>
      <value>Anna</value>
   </attr>
   <attr name='LastName'>
      <value>Allen</value>
   </attr>
```

Reconciliation Attributes

Use these attributes in add requests issued during reconciliation by an authoritative or non-authoritative resource.

The `reverseSync` attribute indicates that the operation is a reverse synchronization (data is pushed from the resource to Select Identity). This value must always be *true*. Use the `reverseSync` attribute for reverse synchronization or in add requests issued during reconciliation by an authoritative or non-authoritative resource.

```
<operationalAttributes>
   <attr name='urn:trulogica:concero:2.0#reverseSync'>
      <value>true</value>
   </attr>
```

The `resouceID` attribute is the name of the resource issuing the request, as in the case of synchronization of resource data. Use this attribute for reverse synchronization or in add requests issued during reconciliation by an authoritative or non-authoritative resource.

```
<operationalAttributes>
  <attr name='urn:trulogica:concero:2.0#resourceId'>
    <value>resource_ID</value>
  </attr>
```

The `keyFields` attribute is used to find the user in the Select Identity database during reconciliation. Use the `keyFields` attribute in add requests issued during reconciliation by an authoritative or non-authoritative resource

```
<operationalAttributes>
  <attr name='urn:trulogica:concero:2.0#keyFields'>
    <value>field</value>
  </attr>
```

The `taUserName` attribute is the name or ID of the user in Select Identity. Use this attribute for all reconciliation requests (standalone `addRequest` and `addRequest` in a `batchRequest`)

```
<operationalAttributes>
  <attr name='urn:trulogica:concero:2.0#taUserName'>
    <value>ID</value>
  </attr>
```

The `taResourceKey` attribute is the name or ID of the user on the resource. Specify this attribute for all reconciliation requests (standalone `addRequest` and `addRequest` in a `batchRequest`).

```
<operationalAttributes>
  <attr name='urn:trulogica:concero:2.0#taResourceKey'>
    <value>key</value>
  </attr>
```

Each attribute name is defined when a service is created in Select Identity. If performing reverse synchronization, the attribute names are defined by the resource, and Select Identity determines this based upon resource mapping.

To provide entitlements, include the attribute name `urn:trulogica:concero:2.0#groups` and provide a value element for each entitlement to grant. If performing reverse synchronization, you do not need to specify the resource.

Attributes for Other Uses

When provisioning administrative user accounts, you can include attributes that enable the user to be provisioned with a specific administrative role and context.

A service can be supported by multiple resources. The services to which a resource is assigned filter out un-needed attributes.

When provisioning administrative user accounts, you can include attributes that enable the user to be provisioned with a specific administrative role and context.

The `SIAdminRole`, `SIService`, and `RoleServiceContext` attributes contain the assigned combination of the role name, the service on which the role exists, and the context.

These attributes can be used to assign any administrator role to a user ID. The following example would assign the `Concero Sys Admin` role on services named `Global` and `Default`. (This combination would only be provisioned onto an administrative user, since the level of permission would be inappropriate to a non-administrative account.)

The `RoleServiceContext` is set to all contexts (using an asterisk as a wildcard for the context), for the assigned role on these services.

```
<attr name='SIAdminRole'>
  <value>Concero Sys Admin</value>
```

```
</attr>
<attr name='SIService'>
   <value>Global</value>
   <value>Default</value>
</attr>
<attr name='RoleServiceContext'>
   <value>Concero Sys Admin|Global|*</value>
   <value>Concero Sys Admin|Default|*</value>
 </attr>
```

The following list demonstrates how the `SIAdminRole`, `SIService`, and `RoleServiceContext` attributes can be assign to a user account any combination of roles, services, and contexts. For example:

- **Specified service, specified context:** `Admin Role 1|Service1|Company|HP`
- **All services, any Company context:** `Admin Role 1|*|Company|*`
- **Specified service, all contexts:** `Admin Role 1|Service1|*`
- **All services, specified context:** `Admin Role 1|*|Company|HP`
- **All services, all contexts:** `Admin Role 1|*|*`

<deleteRequest>

Delete operations using the `deleteRequest` to remove service subscriptions from user accounts that are otherwise left intact. Delete operations do not necessarily delete attributes or attribute values, although they do normally delete entitlement values for a given resource that a service provided.

User account removal must be performed using an `extendedRequest`, which terminates the account. Requests to disable user accounts service memberships are performed using the `extendedRequest`.

The following is a partial example of a `deleteRequest`. The `operationalAttributes` provide login credentials for the administrator or self-service user originating the request, and the service from which the user is to be deleted. The `attributes` element specifies information about the user. These attributes are service form specific.

The `requestID` identifies the provisioning transaction in the `requestResponse`.

```
<deleteRequest... requestID=12345>
```

General Use Attributes

The attributes `UserIDAndOrDomainName` and `password` are the user name and password of the administrator authorized to delete the user, or that of a self-service user, followed by the name of the service on which the request is to be carried out.

```
<operationalAttributes>
   <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
      <value>admin_username</value>
   </attr>
   <attr name='urn:trulogica:concero:2.0#password'>
      <value>123passwd</value>
   </attr>
   <attr name='urn:trulogica:concero:2.0#serviceName'>
```

```
      <value>service_name</value>
   </attr>
```

The following element defines the Select Identity ID of the user to delete.

```
<identifier identifierType='urn:oasis:names:tc:SPML:
1:0#UserIDAndOrDomainName'>
   <id>user</id>
</identifier>
```

Reconciliation Attributes

The `reverseSync` attribute indicates that the operation is a reverse synchronization (data is pushed from the resource to Select Identity). This value must always be *true*. Specify this attribute for reverse synchronization or in delete requests issued during reconciliation by an authoritative or non-authoritative resource

```
<operationalAttributes>
   <attr name='urn:trulogica:concero:2.0#reverseSync'>
      <value>true</value>
   </attr>
```

The key field to update in the Select Identity database during reconciliation. Multiple `keyFields` can be included. Specify this in delete requests issued during reconciliation by an authoritative or non-authoritative resource.

```
<operationalAttributes>
   <attr name='urn:trulogica:concero:2.0#keyFields'>
      <value>field</value>
   </attr>
```

<batchRequest>

The operational attributes of each request in a batch, by default, are taken from those specified at the beginning of the request. Alternatively, operational attributes can be specified for some or all of the requests in the batch.

To add accounts using a `batchRequest`, regardless of whether they are primary, secondary, or normal, you must have Add User permission on the service to which the request is targeted. A `batchRequest` from a user who only has Add Account permission fails.

The following is a partial example of a `batchRequest`.The `requestID` identifies the provisioning transaction in the `requestResponse`.

```
<batchRequest ... requestID=value>
```

General Use Attributes

The user name and password of the administrator authorized to perform the requests. Select Identity authenticates this administrative user against the service.

```
<operationalAttributes>
   <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
      <value>admin_username</value>
</attr>
```

```
<attr name='urn:trulogica:concero:2.0#password'>
   <value>admin_passwd</value>
</attr>
```

The format for each request to process. There may be one or more `addRequest`, `modifyRequest`, or `deleteRequest` elements in any order, and all of the same type.

```
<addRequest requestID=value>
   <operationalAttributes />
      <attributes>
         <attr name="username">
            <value>user1</value>
         </attr>
         ...
      </attributes>
</addRequest>
```

Reconciliation Attributes

The name of the resource issuing the request, during synchronization of resource data.

```
<operationalAttributes>
   <attr name='urn:trulogica:concero:2.0#resourceId'>
      <value>resource_ID</value>
   </attr>
```

The attribute to use to identify the user in Select Identity and the resource.

```
<operationalAttributes>
   <attr name='urn:trulogica:concero:2.0#keyFields'>
      <value>field</value>
   </attr>
```

Indicates that the operation is a reverse synchronization (data is pushed from the resource to Select Identity). This value must always be *true*.

```
<operationalAttributes>
   <attr name='urn:trulogica:concero:2.0#reverseSync'>
      <value>true</value>
   </attr>
```

<extendedRequest>

The `extendedRequest` element allows operations that are not explicitly part of the SPML 1.0 specification. Select Identity web services supports several operations using the `extendedRequest`.

The following examples illustrate operations performed using the `extendedRequest` element. Some examples of typical requests follow the summary of attributes and values.

The `requestID` identifies the provisioning transaction in the `requestResponse`.

```
<extendedRequest ... requestID=value>
```

The user name and password of the administrator authorized to add a user to the service, or that of a self-service user. Select Identity authenticates this user against the service.

```
<operationalAttributes>
   <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
      <value>admin_username</value>
   </attr>
```

```
<attr name='urn:trulogica:concero:2.0#password'>
   <value>admin_passwd</value>
</attr>
```

This attribute names the service where the user will be enabled or disabled in an `enableMembership` or `disableMembership` operation.

```
<attr name='urn:trulogica:concero:2.0#serviceName'>
   <value>service_name</value>
</attr>
```

This attribute defines the user name to modify (may be the same as that in the `<operationalAttributes>` if this is a self-service request).

```
<identifier identifierType='urn:oasis:names:tc:SPML:
1:0#UserIDAndOrDomainName'>
   <id>UserID</id>
</identifier>
```

This attribute identifies the provider of the service (Select Identity).

```
<providerIdentifier providerIDType='urn:oasis:names:tc:SPML:1:0#URN'>
   <providerID>urn:trulogica:concero:2.0</providerID>
</providerIdentifier>
```

This attribute identifies the operation to perform. The following values are valid for the `<operationIdentifier>`, depending upon the request:

```
<operationIdentifier operationIdType='urn:oasis:names:tc:SPML:1:0#URN'>
   <operationID>urn:trulogica:concero:2.0#operation</operationID>
</operationIdentifier>
```

- `enableMembership` – Enable a user's membership on the specified service:

```
<operationIdentifier operationIdType='urn:oasis:names:tc:SPML:1:0#URN'>
   <operationID>urn:trulogica:concero:2.0#enableMembership</operationID>
</operationIdentifier>
```

- `disableMembership` – Disable a user's membership on the specified service:

```
<operationIdentifier operationIdType='urn:oasis:names:tc:SPML:1:0#URN'>
   <operationID>urn:trulogica:concero:2.0#disableMembership</operationID>
</operationIdentifier>
```

- `changePassword` – Change a user's password (for self-service use only):

```
<operationIdentifier operationIdType='urn:oasis:names:tc:SPML:1:0#URN'>
   <operationID>urn:trulogica:concero:2.0#changePassword</operationID>
</operationIdentifier>
```

- `resetPassword` – Reset a user's password (for delegated use only):

```
<operationIdentifier operationIdType='urn:oasis:names:tc:SPML:1:0#URN'>
   <operationID>urn:trulogica:concero:2.0#resetPassword</operationID>
</operationIdentifier>
```

- `enable` – Enable a user (and any secondary users, if this is a primary user):

```
<operationIdentifier operationIdType='urn:oasis:names:tc:SPML:1:0#URN'>
   <operationID>urn:trulogica:concero:2.0#enable</operationID>
</operationIdentifier>
```

- `disable` – Disable a user (and any secondary users, if this is a primary user):

```
<operationIdentifier operationIdType='urn:oasis:names:tc:SPML:1:0#URN'>
   <operationID>urn:trulogica:concero:2.0#disable</operationID>
</operationIdentifier>
```

- `terminate` – Terminate a user:

```
<operationIdentifier operationIdType='urn:oasis:names:tc:SPML:1:0#URN'>
   <operationID>urn:trulogica:concero:2.0#terminate</operationID>
```

```
</operationIdentifier>
```

The attributes for the request follow the `operationIdentifier` element. This attribute defines a new password for the user specified by the `identifier` element.

```
<attributes>
   <attr name='urn:trulogica:concero:2.0#changePassword'>
      <value>passwd</value>
   </attr>
```

This attribute identifies the resource on which to change the user's password.

```
<attributes>
   <attr name='urn:trulogica:concero:2.0#resourceId'>
      <value>resource</value>
   </attr>
```

A user can have mulitple password attributes for multiple resources, Password1 for resource1, Password2 for resouce2, Password for SI, and so forth. If the Password attribute is mapped to multiple resources, then changing the user passoword changes the password for all mapped resources. The specific password attribute you are changing must be sent in order to change that password. In the following example of an administrator changing a user password, the user Password attribute changes the user's login password and the password on *only* the two specified mapped resources:

```
<attributes>
   <attr name='urn:trulogica:concero:2.0#rcPassword'>
      <value>attr:pwd</value>
      <value>new:lmn</value>
      <value>res:lpa75</value>
      <value>res:lpn75</value>
   </attr>
</attributes>
```

The four `<value>` items are:

- Password attribure (`attr:pwd`)
- New password, (`new:lmn`)
- Two resources on which the password change is to take effect (`lpa75` and `lpn75`).

The following is an example for a self-service change password (old password value is passed):

```
<attr name='urn:trulogica:concero:2.0#rcPassword'>
   <value>attr:pwd</value>
   <value>new:xyz1234</value>
   <value>old:abcd1234</value>
   <value>res:lpa75</value>
   <value>res:lpn75</value>
</attr>
```

Enabling Service Membership

The following is an example `extendedRequest` for enabling service membership:

```
<extendedRequest requestID='1769'
execution='urn:oasis:names:tc:SPML:1:0#asynchronous'>
   <operationalAttributes>
      <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
         <value>sisa</value>
      </attr>
```

```
            <attr name='urn:trulogica:concero:2.0#password'>
               <value>abc123</value>
            </attr>
            <attr name='urn:trulogica:concero:2.0#serviceName'>
               <value>hLDAP177</value>
            </attr>
         </operationalAttributes>
         <identifier type='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
            <id>wsuser1</id>
         </identifier>
         <providerIdentifier providerIDType='urn:oasis:names:tc:SPML:1:0#URN'>
         </providerIdentifier>
         <operationIdentifier
         operationIDType='urn:oasis:names:tc:SPML:1:0#URN'>
            <operationID>urn:trulogica:concero:2.0#enableMembership</
            operationID>
         </operationIdentifier>
         <attributes/>
      </extendedRequest>
```

Disabling Service Membership

The following is an example `extendedRequest` for disabling service membership:

```
<extendedRequest requestID='1769'
execution='urn:oasis:names:tc:SPML:1:0#asynchronous'>
   <operationalAttributes>
      <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
         <value>sisa</value>
      </attr>
      <attr name='urn:trulogica:concero:2.0#password'>
         <value>abc123</value>
      </attr>
      <attr name='urn:trulogica:concero:2.0#serviceName'>
         <value>hLDAP177</value>
      </attr>
   </operationalAttributes>
   <identifier type='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
      <id>wsuser1</id>
   </identifier>
   <providerIdentifier providerIDType='urn:oasis:names:tc:SPML:1:0#URN'>
      <providerID>urn:trulogica:concero:2.0</providerID>
   </providerIdentifier>
   <operationIdentifier
   operationIDType='urn:oasis:names:tc:SPML:1:0#URN'>
      <operationID>urn:trulogica:concero:2.0#disableMembership</
      operationID>
   </operationIdentifier>
   <attributes/>
</extendedRequest>
```

Resetting a User's Password

The following is an example `extendedRequest` for resetting a user password:

```
<extendedRequest requestID='1769'
execution='urn:oasis:names:tc:SPML:1:0#asynchronous'>
   <operationalAttributes>
      <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
         <value>sisa</value>
      </attr>
      <attr name='urn:trulogica:concero:2.0#password'>
         <value>abc123</value>
      </attr>
    </operationalAttributes>
   <identifier type='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
      <id>dk60@test.trulogica.com</id>
   </identifier>
   <providerIdentifier providerIDType='urn:oasis:names:tc:SPML:1:0#URN'>
      <providerID>urn:trulogica:concero:2.0</providerID>
   </providerIdentifier>
   <operationIdentifier
   operationIDType='urn:oasis:names:tc:SPML:1:0#URN'>
      <operationID>urn:trulogica:concero:2.0#resetPassword</operationID>
   </operationIdentifier>
   <attributes>
      <!--
          Default Password for all unspecified resources
      -->
      <attr name='urn:trulogica:concero:2.0#rcPassword'>
         <value>new:abcd1234</value>
      </attr>
   </attributes>
</extendedRequest>
```

Enabling a User Account

The following is an example of an `extendedRequest` for enabling a user account:

```
<extendedRequest requestID=value execution='urn:oasis:names:tc:SPML:
1:0#asynchronous'>
   <operationalAttributes>
      <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
         <value>admin_username</value>
      </attr>
      <attr name='urn:trulogica:concero:2.0#password'>
         <value>admin_passwd</value>
      </attr>
   </operationalAttributes>

   <identifier type='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
      <id>id</id>
   </identifier>
   <providerIdentifier providerIDType='urn:oasis:names:tc:SPML:1:0#URN'>
      <providerID>urn:trulogica:concero:2.0</providerID>
   </providerIdentifier>
   <operationIdentifier operationIDType='urn:oasis:names:tc:SPML:
   1:0#URN'>
      <operationID>urn:trulogica:concero:2.0#enable</operationID>
   </operationIdentifier>
   <attributes/>
</extendedRequest>
```

Disabling a User Account

The following is an example of an `extendedRequest` for disabling a user account:

```
<extendedRequest requestID='1769'
execution='urn:oasis:names:tc:SPML:1:0#asynchronous'>
   <operationalAttributes>
      <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
         <value>sisa</value>
      </attr>
      <attr name='urn:trulogica:concero:2.0#password'>
         <value>abc123</value>
      </attr>
   </operationalAttributes>
   <identifier type='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
      <id>wsuser1</id>
   </identifier>
   <providerIdentifier providerIDType='urn:oasis:names:tc:SPML:1:0#URN'>
      <providerID>urn:trulogica:concero:2.0</providerID>
   </providerIdentifier>
   <operationIdentifier
   operationIDType='urn:oasis:names:tc:SPML:1:0#URN'>
       <operationID>urn:trulogica:concero:2.0#disable</operationID>
   </operationIdentifier>
   <attributes/>
</extendedRequest>
```

Terminating a User Account

The following is an example of an `extendedRequest` for terminating a user account:

```
<extendedRequest requestID='1769'
execution='urn:oasis:names:tc:SPML:1:0#asynchronous'>
   <operationalAttributes>
      <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
         <value>sisa</value>
      </attr>
      <attr name='urn:trulogica:concero:2.0#password'>
         <value>abc123</value>
      </attr>
      <attr name='urn:trulogica:concero:2.0#serviceName'>
         <value>hLDAP177</value>
      </attr>
   </operationalAttributes>
   <identifier type='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
      <id>wsuser1</id>
   </identifier>
   <providerIdentifier providerIDType='urn:oasis:names:tc:SPML:1:0#URN'>
      <providerID>urn:trulogica:concero:2.0</providerID>
   </providerIdentifier>
   <operationIdentifier
   operationIDType='urn:oasis:names:tc:SPML:1:0#URN'>
      <operationID>urn:trulogica:concero:2.0#terminate</operationID>
   </operationIdentifier>
</extendedRequest>
```

<modifyRequest>

This request type is used to modify an existing user account.

You cannot modify a user's password using the `modifyRequest` element; use the `extendedRequest` element with the `passwordReset` operational identifier for this purpose. If you modify a `Password` attribute, this causes the appropriate password request to be generated internally.

The `modifyRequest` element cannot be used to add or remove service subscriptions. These operations are also performed using `addRequest` and `deleteRequest`.

If you use the `Replace` operation in a `modifyRequest` to modify an entitlements attribute, ensure that the user's current entitlements are specified as part of the request. If you do not specify the existing entitlements, they are removed from the user. The `requestID` identifies the provisioning transaction in the `requestResponse`.

```
<modifyRequest ... requestID=value>
```

General Use Attributes

The following attributes define the user name and password of the administrator authorized to add a user to the service, or the user if this is a self-service request. Select Identity authenticates this user against the service. Specify this element for all modify requests

```
<operationalAttributes>
   <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
      <value>sisa</value>
   </attr>
   <attr name='urn:trulogica:concero:2.0#password'>
      <value>abc123</value>
   </attr>
```

The name of the service where the user to be modified is a member. Specify this attribute for user provisioning (from Select Identity to the resource) in modify requests only. Omit this attribute from modify profile requests.

```
<operationalAttributes>
   <attr name='urn:trulogica:concero:2.0#serviceName'>
      <value>hLDAP177</value>
   </attr>
```

The following attribute defines the Select Identity ID of the user to modify. If this is the same as the user specified in the `<operationalAttributes>`, the request is self-service.

```
<identifier type='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
   <id>wsuser1</id>
</identifier>
```

The following are name-value pairs of the attribute to modify and the operation to perform, where `attr_name` is the name of the attribute and `value` is the value. Each attribute name is defined when an administrator creates a service in Select Identity.

```
<modification name=attr_name operation=replace>
  <value>value</value>
</modification>
```

Available operations for `modifyRequest` depend on whether the target attribute is single- or multi-value, and are as follows:

- `add`: If the target is a single-value attribute and is empty, this adds the specified value. If the target is a multi-value attribute, it adds the value to the existing list of values. If the target attribute is single-value and already populated, this overwrites the existing value.
- `delete`: If the target is a single-value attribute and is populated, this deletes the specified value (subject to an attribute named `AllowDeletionSetting`). If the target is single-value and empty, the operation does nothing. If the target is a multi-value attribute, the delete operation deletes all values by default.
- `replace`: Replaces the value of an attribute with the value specified. It overwrites all values of a multi-value field with the specified value. If the attribute is single-value and empty, this operation adds the value to the attribute.

To modify entitlements, specify the resource name using the following format, and provide a value element for each attribute to grant. Specify `add` or `delete` (instead of `replace`) as the operation. If performing reverse synchronization, you do not need to specify the `resource`:

```
<modification name='urn:trulogica:concero:2.0#groups:resourceName'
operation=add>
```

Reconciliation Attributes

The following attribute indicates that the operation is a reverse synchronization (data is pushed from the resource to Select Identity). This value must always be *true*. Include this attribute in reverse synchronization or in modify requests issued during reconciliation by an authoritative or non-authoritative resource.

The following attribute defines the resource issuing the request, as in the case during synchronization of resource data. Include this attribute for reverse synchronization or in modify requests issued during reconciliation by an authoritative or non-authoritative resource.

```
<operationalAttributes>
   <attr name='urn:trulogica:concero:2.0#resourceId'>
      <value>resource_ID</value>
   </attr>
```

The keyFields attribute is used to find the user in the Select Identity database during reconciliation. Include the `keyFields` attribute in modify requests issued during reconciliation by an authoritative or non-authoritative resource.

```
<operationalAttributes>
   <attr name='urn:trulogica:concero:2.0#keyFields'>
      <value>field</value>
   </attr>
```

If performing reverse synchronization, the attribute names are defined by the resource, and Select Identity determines this based on attribute mapping.

```
<operationalAttributes>
   <attr name='urn:trulogica:concero:2.0#reverseSync'>
      <value>true</value>
   </attr>
```

<searchRequest>

The following is a partial example of a `searchRequest`. Use this element to perform user account searches on various criteria. You can only search for user accounts; searches for other objects such as services or resources, are not currently supported.

Only non-sensitive attributes are retrieved. Sensitive (encrypted) attributes are not retrieved.

Web services search operations are case-sensitive.

The behavior of this request type can be configured along with the user search functionality in the Select Identity web interface. This is done by adding, removing, or changing columns in the TAUser table.

The TAUser table columns are controlled by a setting in the `TruAccess.properties` file, which is documented in an appendix to the *HP Select Identity Installation Guide*.

If you do not include the `searchBase` or a `filter` attribute, a `searchRequest` returns all users. Therefore, use the TruAccess properties to set the maximum possible number of search results that an operation returns. You can also use the available `filter` attributes to set search criteria. The following are the default search criteria attributes:

- UserName
- Email
- FirstName
- LastName
- Status

The `Status` attribute contains the user state status value. For more information about possible values, refer to the *HP Select Identity Installation Guide*. An example `<filter>` attribute containing all of the possible values follows, with the textual value name in a comment line:

```
<filter>
   <equalityMatch name="urn:oasis:names:tc:SPML:1:0#status">
      <value>101</value>
         <!- Created ->
      <value>-10</value>
         <!- Disabled ->
      <value>-101</value>
         <!- Terminated ->
   </equalityMatch>
</filter>
```

The number of records retrieved by a `<searchRequest>` can be limited on a per-request basis by adding `maxResultSize` in an `equalityMatch` attribute, as follows:

```
<filter>
   <equalityMatch name='urn:trulogica:concero:2.0#maxResultSize'>
      <value>10</value>
   </equalityMatch>
</filter>
```

The `requestID` identifies the provisioning transaction in the `requestResponse`.

```
searchRequest ... requestID=12345>
```

The user name and password of the administrator authorized to search for a user on the service. Select Identity authenticates this administrative user against the service.

```
<operationalAttributes>
   <attr name='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
      <value>username</value>
   </attr>
   <attr name='urn:trulogica:concero:2.0#password'>
      <value>admin_passwd</value>
   </attr>
```

This optional attribute specifies a search operation type, which retrieves permissions for the specified user.

```
<attr name='urn:hp:selectidentity:4.0#getPermissions' >
   <dsml:value>true</dsml:value>
</attr>
```

This optional attribute specifies the user whose attributes you want to retrieve from Select Identity.

```
<searchBase type='urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName'>
   <id>user_ID</id>
</searchBase>
```

The following attribute specifies a search filter, which in this case retrieves matches on a specific resource (`resourceName`). You can also use `serviceName`.

```
<filter>
   <equalityMatch name='urn:trulogica:concero:2.0#resourceId'>
      <value>resourceName</value>
   </equalityMatch>
</filter>
```

Search Results

Search results are returned within a `searchResponse` element.

Self-Service Requests

Web services requests can be used for self-service operations. Self-service requests are submitted by a user on his or her own behalf. Self-service requests are identical to delegated requests in every regard except for the access level and permissions granted to the user. Self-service accounts normally have a lower level of access and administration privileges than does an administrator or a user who makes delegated requests.

Users who originate their own `addRequest` for secondary user IDs must have the *Add Account* permission in their role on the service where the account is to be added.

5 SPML 2.0 Structure and Elements

This section provides detailed discussion of the supported SPML 2.0 request elements and their attributes, illustrated with snippets of SPML. You should familiarize yourself with the SPML 2.0 specification before reading on. An understanding of the SPML 2.0 schema will help your understanding of this implementation of SPML 2.0 in Select Identity.

The Select Identity product CD contains a `samples` directory, where you will find a set of web services SPML request examples. These are provided so that you can use them as a basis for developing custom requests.

Select Identity's support for SPML 2.0 includes the following Web Service user management operations for a delegated user:

- Add a user to a service (`addRequest`) / asynchronous operation mode.
- Modify a user in a service, modify a user profile attributes (`modifyRequest`) / asynchronous operation mode.
- Delete a user from a service, terminate a user from Select Identity (`deleteRequest`) / asynchronous operation mode.
- Lookup a user's service attributes, lookup a user's profile attributes (`lookupRequest`) / synchronous operation mode.
- Disable a user from a service, disable a user in Select Identity (`suspendRequest`) / asynchronous operation mode.
- Enable a user in a service, enable a user in Select Identity (`resumeRequest`) / asynchronous operation mode.
- Query a user's status in a specified service, query a user's status in Select Identity (`activeRequest`) / synchronous operation mode.
- Query all targets available for provisioning (`listTargetsRequest`) / synchronous operation mode.
- Change a user's password to a pre-defined value (`setPasswordRequest`) / asynchronous operation mode.
- Reset a user's password (`resetPasswordRequest`) / asynchronous operation mode.
- Cancel an asynchronous request if it is still pending (`cancelRequest`) / synchronous operation mode.
- Query the status of an asynchronous request (`statusRequest`) / synchronous operation mode.

Select Identity's support for SPML 2.0 includes the following Web Service user management operations for a self-service user:

- Add self to a specified service (`addRequest`) / asynchronous operation mode.
- Modify self in a specified service, modify profile attributes for self (`modifyRequest`) / asynchronous operation mode.
- Delete self from a specified service (`deleteRequest`) / asynchronous operation mode.

- Lookup a user's service attributes, lookup a user's profile attributes (`lookupRequest`) / synchronous operation mode.
- Reset the requestor's password (`setPasswordRequest`) / asynchronous operation mode.
- Cancel an asynchronous request if it is still pending (`cancelRequest`) / synchronous operation mode.
- Queries the status of an asynchronous Request) (`statusRequest`) / synchronous operation mode.

Existing SPML 1.0 Web Service functionality will continue to work as-is. SPML 2.0 uses a different end-point URL than that of 1.0. Refer to WSDL on page 62 for information about the Web Servicing Description Language, how to access it, the endpoint URL address location, and how to retrieve the referenced XSDs.

Provisioning Targets

Based upon the provisioning target, different provisioning actions take place. The targets are defined as follows:

- Service – used for service specific operations like adding user to a service. Format "Service:<*servicename*>"
- User – used for user specific operations like modifying profile attributes.
- Attribute – used for password set/reset operation on Select Identity password attribute
- Resource – used for password set/reset operation on Select Identity resource

Targets are identified in a request in the `targetID` attribute.

Web Service clients use SOAP based SPML/XML requests for remote provisioning and administration of user accounts in Select Identity. In a typical web service scenario, the web service client interrogates the service provider services by sending a schema request operation. Select Identity responds to the request by returning a schema description for all of the services that Select Identity exposes for provisioning. Based on the schema description, the Web Service client creates a user provisioning request, optionally validates requests locally, and sends the request to Select Identity for provisioning. The provisioning target varies in relation to the context of the request.

Target Discovery

The `listTargetsRequest` element is used to interrogate all installed Select Identity targets for their identifier and schema.

listTargetsRequest without a Target Identifier

If the `targetID` attribute is not present in the request, the response is as follows:

- For a service, all `targetID`s of all business and admin services in Select Identity based on the requestor role display. Each service is identified using a unique target identifier in the format "Service:<*servicename*>". Composite services *are not* displayed.

- For a user, the schema for user profile attributes in Select Identity under one target identifier displays. The `targetID` is "Identity:User".

This user target name is fixed.

- For attributes, the schema for each and every password attribute defined in Select Identity displays. Each password attribute schema will be displayed using a unique target identifier. The `targetID` format is “Attribute:<*password attribute name*>”.
- For resources, all `targetIDs` display using the format “Resource:<*resource name in* Select Identity>”.

The “Attribute:<*password attribute name*>” target does not display schema.

listTargetsRequest with a Target Identifier

When the `targetID` is provided, as in the case of user and attribute targets, the schema of specified target is returned.

Refer to <ListTargetsRequest> on page 50 for an example.

Service Target

The service target is called “Service:<*servicename*>.” The services are used to provision a user in a specified Select Identity service. An SPML 2.0 request can support only one target per request. To provision more than one service per user, you must use separate requests. Only business and administrator services schema are displayed in the `listTargetsResponse`. Composite services are not displayed in the `listTargetsResponse` operation; composite services are not supported.

Supported SPML 2.0 core operations are:

- Add user to a specified service target (`addRequest`) / asynchronous operation mode. Refer to <addRequest> on page 51 an example.
- Modify user in a specified service target (`modifyRequest`) / asynchronous operation mode. Refer to <modifyRequest> on page 54 for an example.
- Delete user from a specified service target (`deleteRequest`) / asynchronous operation mode. Refer to <deleteRequest> on page 55 for an example.
- Lookup user's service attributes (`lookupRequest`) / synchronous operation mode. Refer to <lookupRequest> on page 56 for an example.

Supported SPML 2.0 suspend operations are:

- Disable user on a specified service target (`suspendRequest`) / asynchronous operation mode. Refer to <suspendRequest> on page 56 for an example.
- Enable user on a specified service target (`resumeRequest`) / asynchronous operation mode. Refer to <resumeRequest> on page 57 for an example.
- Check the user's service status (`activeRequest`) / synchronous operation mode. Refer to <activeRequest> on page 57 for an example.

The available service names in an installation can be discovered using the `listTargetsRequest` without a `targetID`. Services use the core capabilities of add, delete, modify, lookup, suspend, resume, active, and status. A user provisioning operation takes place in the service and in the underlying resource(s) using service as the provisioning target.

A typical use case scenario for SPML 2.0 add/modify (asynchronous) requests on a service target is as follows:

1. The Web Service client sends `listTargetsRequest` without a `targetID` to Select Identity.
2. Select Identity sends a `listTargetsResponse` with targetIDs of all services.
3. The Web Service client sends a `listTargetsRequest` with a `targetID` of the form "Service:<*servicename*>" to interrogate the Select Identity service schema.
4. Select Identity sends a `listTargetsResponse` with a schema for the specific service.
5. The Web Service client creates an `addRequest` or `modifyRequest` XML using SPML 2.0 schema. This `addRequest` or `modifyRequest` contains a `data` element whose XML is configured based upon the Select Identity service schema returned in the `listTargetsResponse`.
6. The Web Service client validates the `addRequest` or `modifyRequest` XML file using a third-party validating XML editor.
7. The Web Service client sends the `addRequest` XML to Select Identity.
8. Select Identity validates and creates an internal request for processing. If the input request does not contain a valid `requestID`, Select Identity generates a new `requestID`.
9. Since Select Identity treats `addRequest` as an asynchronous operation, and responds with the status of the `addRequest` (failure or pending) in an `addResponse`. The addResponse contains the Select Identity request status information like `RequestorName`, Select Identity `requestID`, Select Identity status, and so forth. The request status information is identical to what one sees on a Select Identity Request Status page.

The internal Select Identity `requestID` in the request status is different from the `requestID` of the `addResponse`.

10. The Web Service client sends a status request to query the status of this asynchronous operation. In the `statusRequest`'s `asyncRequestId`, the Web Service client uses the `requestID` returned in the `addResponse`.
11. Select Identity responds with the status of the operation as success, failure, or pending.

In a typical use case scenario for SPML 2.0 Delete/Suspend/Resume (asynchronous) requests on a service target, the flow of operations is similar to the previous add and modify scenario. The only difference in the flow of operations is that these requests do not contain a data element; for example, there is no need to configure XML based upon the service schema.

User Target

A user target is called with “Identity:User,” and it is a fixed ID in the system. User operations are service independent and can lookup or modify user profiles, or terminate a user in Select Identity.

The schema for "Identity:User" is displayed as part of `listTargetsResponse`.
The user target is fixed for all user management operations (modify profile, enable or disable user in Select Identity, and so forth).

Supported SPML 2.0 core operations are:

- Modify user profile attributes in Select Identity (`modifyRequest`) / asynchronous operation mode.

- Create a Select Identity request to terminate a user, or terminate a user in Select Identity (`deleteRequest`) / asynchronous operation mode.
- User profile attributes displayed. Lookup specified user info in Select Identity (`lookupRequest`) / synchronous operation mode.

Supported SPML 2.0 suspend operations are:

- Create a Select Identity request to disable a user (`suspendRequest`) / asynchronous operation mode.
- Create a Select Identity request to enable use (`resumeRequest`) / asynchronous operation mode.
- Display the user status (enabled/disabled) in Select Identity (`activeRequest`) / synchronous operation mode.

Attribute Target

Attribute target is used to support the SPML 2.0 password capability operations, `setPasswordRequest` and `resetPasswordRequest`. Currently, Select Identity does not support expire and validate password features within the SPML 2.0 password capability.

The sequence of workflow is as follows:

1. The administrator sends a `lookupRequest` with `psoID` set to `username` and `targetID` set to “Identity:User.”
2. SI responds with user information which contains the following:
   - ServiceNames
   - ResourceNames
   - Mapped SI Password attribute
3. The administrator sends a `listTargetsRequest`. The `listTargetsResponse` from Select Identity contains a list of targets that support password functions. These targets are in the form "Attribute:<*attributename*>” and/or "Resource:<*resourcename*>.”
4. The administrator looks for the appropriate target in the `listTargestResponse` that contains the matching Select Identity attribute name as returned in the lookup response for the user. For example, if the lookup response contains "Password" as the mapped Select Identity attribute, the corresponding target name will be "Attribute:Password."

An attribute target is called with “Attribute:<*attributename*>.” Attribute targets set or reset passwords for a user on *all* resources mapped to the Select Identity password attribute. By default, the Select Identity `Password` attribute contains the definition of a user's login password. Any password set/reset request on the `Password` attribute results in changing a user's Select Identity login password.

Supported SPML 2.0 password operations are:

- Sets the password of specified user in Select Identity and on all resource(s) mapped to this password attribute (`setPasswordRequest`). The password is set to a new value from the `setPasswordRequest`. / Asynchronous operation mode. Refer to <setPasswordRequest> on page 57 for an example.

An administrator can set a user's password to a pre-assigned value in the `setPasswordRequest`.

- Generates a new password for the specified user (`resetPasswordRequest`). All resource passwords mapped to the given Select Identity Attribute are reset. The password attribute in `targetID` should be associated with the password generation function and the "Generate on Reset" property should be enabled in Select Identity. / Asynchronous operation mode. Refer to <resetPasswordRequest> on page 58 for an example.

Only an administrator can reset a password.

The password can be successfully reset *only* when the Password attribute is associated with the password generation function and the "Generate on Reset" property is enabled in Select Identity. This rule is presently *not* enforced when resetting a password using a web service request.

Resource Target

A resource target is used to set/reset the password of a user on a specific resource and is called with “Resource:<*resourcename*>.”

Supported SPML 2.0 password operations are:

- The user’s password is changed on the specified resource only (`setPasswordRequest`). If the resource password attribute is mapped to the Select Identity login password, then the user's login password in Select Identity will also be set. / Asynchronous operation mode.
- Generates a new password of a specified user in Select Identity (`resetPasswordRequest`). The user's password is reset on the specified resource only. If the resource password attribute is mapped to the Select Identity login password, then the user's login password in Select Identity will also be reset. / Asynchronous operation mode.

Targetless Operation

The following two requests do not use a target. Both require a `requestID` to identify the request to act upon. Neither of these request types target a service, user, attribute, or a resource.

- Checks the status of an asynchronous request (`statusRequest`) / synchronous operation mode.
- Cancels an asynchronous request (`cancelRequest`) / synchronous operation mode.

Core Capability Operations Examples

The following operations are required for compliance with SPML 2.0.

<ListTargetsRequest>

The `listTargets` request retrieves a set of targets that are available for provisioning. When the service name is used in the `targetID` attribute, the request returns the schema for that service, which determines the set of capabilities that are supported.

If the `targetID` attribute is left empty, then the response returns *only* the names of all available services for that installation.

```
<listTargetsRequest xmlns="urn:oasis:names:tc:SPML:2:0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:SPML:2:0 file:/C:/SPML2.0/validate/
pstc_spmlv2_core.xsd" />
```

The `hpsi:targetID` attribute provides the service name of an available service. The response in that case is the schema for that service.

```
<spml2:listTargetsRequest xmlns:spml2="urn:oasis:names:tc:SPML:2:0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:hpsi="urn:hp:si:spml20:common" hpsi:targetID="Service:spml20 test" >
```

The following `listTargetsResponse` example returns the schema for Attribute:Password.

```
<listTargetsResponse requestID="" status="success"
xmlns="urn:oasis:names:tc:SPML:2:0">
   <target profile="urn:oasis:names:tc:SPML:2.0:profiles:XSD"
   targetID="Attribute:Password">
      <schema>
         <xs:schema targetNamespace="urn:hp:si:spml20:Attribute:Password"
         xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns="urn:hp:si:spml20:Attribute:Password">
            <xs:element name="Password">
               <xs:complexType>
                 <xs:simpleContent>
                    <xs:extension base="PasswordType">
                       <xs:attribute name="AutoGenerateOnReset"
                       type="xs:boolean" fixed="true"/>
                    </xs:extension>
                 </xs:simpleContent>
               </xs:complexType>
            </xs:element>
            <xs:simpleType name="PasswordType">
               <xs:restriction base="xs:string">
                  <xs:minLength value="1"/>
                  <xs:maxLength value="64"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:schema>
      </schema>
      <capabilities>
         <capability namespaceURI="urn:oasis:names:tc:SPML:2.0:password">
         </capabilities>
   </target>
   <target profile="urn:oasis:names:tc:SPML:2.0:profiles:XSD"
   targetID="Attribute:wstwoway">
      <schema> ...
```

<addRequest>

Select Identity supports several variations of the `addRequest` element. In any `addRequest`, if the user name already exists in Select Identity, then other specified attributes, such as service memberships, are not updated. An `addRequest` only adds a user account if the specified account does not already exist in Select Identity. Since a `modifyRequest` only allows modification of profile attributes, the `addRequest` is used whenever the objective is to add a user account to a given service.

The following is a partial example of an `addRequest`. The `wsse:Security` element provides login credentials for the administrator or self-service user originating the request and is located in the `soap:Header` element. Adding one or more services with a single `addRequest` requires a separate `targetID` element and attributes for each service in your Select Identity system. In this example the `RootCtxDelegatedAddUserDefaultf020View` element provides information about the user's service attribute values. These attributes are service form specific, because a given form may not have all service attributes. Refer to <ListTargetsRequest> on page 50.

An attribute cannot be accessed if it is not in the service form for the request type, not updateable, or not visible.

```
<spml2:addRequest xmlns:spml2="urn:oasis:names:tc:SPML:2:0"
xmlns:hpsi="urn:hp:si:spml20:common"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:hp:si:spml20:SPML20AdminService"
targetID="Service:SPML20AdminService" >
```

The ID of the provisioning service target is specified in the `targetID.` The `requestID` element is optional, and if not provided in the request, then the provider generates a `requestID`.

You can only provision a single user onto one service in any request.

```
    <spml2:psoID ID="spml20admin"></spml2:psoID>
```

If the request is from a delegated user, the `userid` in the `soap:Header` and the `psoID` identity in the request are different. If the request is self-service, then the `userid` in the `soap:Header` and the `psoID` in the request are the same.

The Select Identity ID of the user to add is within the `psoID` element. The `ID` attribute provides the user name. The service to which the user is being added is in the `targetID` attribute.

```
<RootCtxDelegatedAddUserDefaultf020View>
   <Company>HP</Company>
   <State>TX</State>
   <Email>admin1.spml20@si.hp.com</Email>
   <FirstName>admin1</FirstName>
   <LastName>spml20</LastName>
      <Password>PASSWORD</Password>
   <UserName>spml20admin</UserName>
   <SIAdminRole>SPML 20 Admin</SIAdminRole>
   <SIService>*</SIService>
   <RoleServiceContext>SPML 20 Admin|*|*</RoleServiceContext>
</RootCtxDelegatedAddUserDefaultf020View>
```

The `RootCtxDelegatedAddUserDefaultf020View` element contains the values to add for the user. You must provide an element for each attribute required by the service(s) for a newly created user. *Do not* specify elements that are autogenerated. Additionally, the `RootCtxDelegatedAddUserDefaultf020View` element is in context with the request and specifies the form that is used. Its content is parsed as the context (`RootCtx`), the event (`DelgatedAddUser`), and the form (`Defaultf020View`).

Any special characters in service or form names are replaced with unicoded values as in the name "Defaultf020View". The unicode value "f020" is substituted for a whitespace.

The values used are determined by the schema, and may be discovered using the `listTargetsRequest` for the user target.

A service can be supported by multiple resources. The services to which a resource is assigned filter out un-needed attributes.

When provisioning administrative user accounts, you can include attributes that enable the user to be provisioned with a specific administrative role and context.

The `SIAdminRole`, `SIService`, and `RoleServiceContext` elements contain the assigned combination of the role name, the service on which the role exists, and the context.

These elements can be used to assign any administrator role to a user ID. The following example would assign the `SPML 20 Admin` role on all services. (The following combination would only be provisioned onto an administrative user, since it confers a level of permission inappropriate to a non-administrative account.)

The `RoleServiceContext` is set to all companies and all contexts (using an asterisk as a wildcard for the context), for the assigned role on this service.

```
<SIAdminRole>SPML 20 Admin</SIAdminRole>
<SIService>*</SIService>
<RoleServiceContext>SPML 20 Admin|*|*</RoleServiceContext>
```

The previous snippet shows the role of `SPML 20 Admin`, specified for all services, in all contexts. The following examples show various ways the rules may specify roles, service, and context.

- **Specified service, specified context:**

```
<SIAdminRole>SPML 20 Admin</SIAdminRole>
<SIService>Service1</SIService>
<RoleServiceContext>SPML 20 Admin|HP|Company</RoleServiceContext>
```

- **All services, any Company context:**

```
<SIAdminRole>SPML 20 Admin</SIAdminRole>
<SIService>*</SIService>
<RoleServiceContext>SPML 20 Admin|*|Company</RoleServiceContext>
```

- **Specified service, all contexts:**

```
<SIAdminRole>SPML 20 Admin</SIAdminRole>
<SIService>Service1</SIService>
<RoleServiceContext>SPML 20 Admin|*|*</RoleServiceContext>
```

- **All services, specified context:**

```
<SIAdminRole>SPML 20 Admin</SIAdminRole>
<SIService>*</SIService>
<RoleServiceContext>SPML 20 Admin|HP|Company</RoleServiceContext>
```

- **All services, all contexts:**

```
<SIAdminRole>SPML 20 Admin</SIAdminRole>
<SIService>*</SIService>
<RoleServiceContext>SPML 20 Admin|*|*</RoleServiceContext>
```

You may check the status for an `addRequest` using the `statusRequest`. Refer to <statusRequest> on page 60.

<modifyRequest>

This request type is used to modify an existing user in a service using a service target, or to modify a user profile attributes using the “Identity:User” target.

You cannot modify a user’s password using the `modifyRequest` element; use the `setPasswordRequest` or `resetPasswordRequest` element for this purpose. If you modify a `Password` attribute, this causes the appropriate password request to be generated internally.

The `modifyRequest` element cannot be used to add or remove service subscriptions. You must use `addRequest` and `deleteRequest` to perform for those operations.

If you use the `Replace` operation in a `modifyRequest` to modify an entitlement attribute, ensure that the user’s current entitlements are specified as part of the request. If you do not specify the existing entitlements, they are removed from the user’s account.

The `requestID` element is optional in the `modifyRequest`. If not provided in the request, then the provider generates a `requestID`. The following is an example of the `modifyRequest`

```
<spml2:modifyRequest xmlns:spml2="urn:oasis:names:tc:SPML:2:0"
xmlns:hpsi="urn:hp:si:spml20:common" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xmlns="urn:hp:si:spml20:spml20 test">
```

The user name and password of the administrator authorized to add a user to the service, or the user if this is a self-service request is in the `wsse:Security` element of the `soap:Header`. Select Identity authenticates this user against the service.

```
<spml2:psoID ID="moduser1" targetID="Service:spml20 test"/>
```

The user is identified in the `ID` attribute of the `psoID` element. The service target is identified in the `targetID` attribute.

```
<spml2:modification modificationMode="replace">
   <spml2:data>
      <!-- Attributes within data element are based on
      DelagatedModifyEvent from Business service XSD schema in the
      default namespace -->
      <CompanyDelegatedModifyUserDefaultf020ViewElem>
         <FirstName>Manual UserRow7</FirstName>
      </CompanyDelegatedModifyUserDefaultf020ViewElem>
   </spml2:data>
</spml2:modification>
```

The `data` element contains the values for modification, and the context and form that is used. The attributes are based on the `DelagatedModifyEvent` from the Business service XSD schema in the default namespace (`CompanyDelegatedModifyUserDefaultf020ViewElem`).

In a self-service operation the administrator identified in the `wsse:Security` element `username` and the user in the `psoID` element are the same.

```
<spml2:psoID ID="moduser1" targetID="Identity:User"/>
```

The user is identified in the ID attribute of the `psoID` element. The user target is identified in the `targetID`. The accompanying attributes within the `data` element are based on the `ModifyUser` element from the user target XSD schema in the default namespace. The context and form changes based upon the target.

```
<spml2:modification modificationMode="replace">
   <spml2:data>
      <ModifyUser>
```

```
            <FirstName>DelegatedModifyProfile</FirstName>
         </ModifyUser>
      </spml2:data>
   </spml2:modification>
```

Available operations for the `modifyRequest` depend on whether the target attribute is single- or multi-value, and are as follows:

- `add`: If the target is a single-value attribute and is empty, this adds the specified value. If the target is a multi-value attribute, then the value is added to the existing list of values. If the target attribute is single-value and already populated, this overwrites the existing value.
- `delete:` If the target is a single-value attribute and is populated, this deletes the specified value (subject to an attribute named `AllowDeletionSetting`). If the target is single-value and empty, the operation does nothing. If the target is a multi-value attribute, then all values are deleted by default.
- `replace:` Replaces the value of an attribute with the value specified. the operation overwrites all values of a multi-value field with the specified value. If the attribute is single-value and empty, this operation adds the value to the attribute.

<deleteRequest>

Delete operations using the `deleteRequest` remove user accounts.

The `deleteRequest` removes a user from the service target. All capabilities assigned are removed simultaneously. The `requestID` of the provisioning transaction identifies the transaction for asynchronous transactions and is optional.

The user name and password of the administrator authorized to delete the user, or that of a self-service user, followed by the name of the service on which the request is to be carried out is in the `Security` element in the `soap:Header` element of the message.

The following snippet shows the delete request with the Select Identity ID of the user to delete within the `psoID` element. The `ID` attribute provides the user name. The service from which to delete the user is in the `targetID` attribute.

```
<spml2:deleteRequest xmlns:spml2="urn:oasis:names:tc:SPML:2:0"
xmlns:hpsi="urn:hp:si:spml20:common" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance" xmlns="urn:hp:si:spml20:spml20 test"
requestID="D1101">
   <spml2:psoID ID="deluser1" targetID="Service:spml20 test"/>
</spml2:deleteRequest>
```

If the user is being deleted in Select Identity the target would be “Identity:User” as shown in the following snippet.

```
<spml2:deleteRequest xmlns:spml2="urn:oasis:names:tc:SPML:2:0"
xmlns:hpsi="urn:hp:si:spml20:common" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance" xmlns="urn:hp:si:spml20:spml20 test"
requestID="D1101">
   <spml2:psoID ID="deluser1" targetID="Identity:User"/>
</spml2:deleteRequest>
```

<lookupRequest>

The following is a partial example of a `lookupRequest`. Use this element to look up a user's service attributes using a service target, or lookup a user profile attributes using the "Identity:User" target. This operation should be done synchronously to avoid other operations interfering with the data, since the desired information should reflect the current state of a target object.

```
<spml:lookupRequest xmlns:spml="urn:oasis:names:tc:SPML:2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
schemaLocation="urn:oasis:names"tc"SPML:2.0 file:/C:/SPML2.0/sischema/
SampleSchema.xsd" executionMode="synchronous" returnData="data"
requestID="1004">
```

The `lookupRequest` contains the `executionMode` attribute that specifies synchronous execution. The `requestID` is the token that identifies this request for the `lookupResponse`. The `returnData` attribute specifies the content that is returned in the reponse.

```
<spml:psoID ID="user1" targetID="Service:spml20 test">
</spml:psoID>
```

The `psoID` element contains an `ID` and a `targetID` attribute. The `ID` attribute contains the user's name that is to be looked up. The `targetID` attribute is the target service.

The `lookupResponse` returns a status attribute that indicates if the data was successfully returned. If a successful return occurs, the `lookupResponse` contains a `pso` element, which may contain a data element.

Suspend Capability Operations Examples

<suspendRequest>

This operation may only be used by a delegated administrator. It is used to disable a user account in a service and to disable a user account in Select Identity. The suspend request may be used with both service and user targets.

The following is a partial example of a `suspendRequest`. The `targetID` pertains to the user target.

```
<spml2:suspendRequest xmlns:spml2="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:hpsi="urn:hp:si:spml20:common"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" requestID="S1101">
```

The `suspendRequest` disables a user's capabilities in Select Identity. All capabilities assigned are disabled and remain disabled until counter-acted by a `resumeRequest`. The `requestID` of the provisioning transaction identifies the transaction for asynchronous transactions.

The user name and password of the delegated administrator is in the `Security` element in the `soap:Header` element of the message.

The Select Identity ID of the user to suspend is within the `psoID` element. The `ID` attribute provides the user name. The user target is set in the `targetID` attribute and may be either a service target or the "Identity:User" target.

```
<spml2:psoID ID="suspend1" targetID="Identity:User"/>
```

<resumeRequest>

This operation may only be used by a delegated administrator. It is used to enable a user in a service and to enable a user in Select Identity. The resume request may be used with both service and user targets.

The following is a partial example of a `resumeRequest`. The `targetID` pertains to User target. These attributes are service form specific.

```
<spml2:resumeRequest xmlns:spml2="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:hpsi="urn:hp:si:spml20:common"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" requestID="RE1101">
```

The `resumeRequest` allows the administrator to enable a user whose account was disabled. The `requestID` identifies the transaction for asynchronous transactions.

The user name and password of the delegated administrator authorized to enable a disabled user account is in the `Security` sub-element in the `soap:Header` element of the message.

The Select Identity ID of the user whose account is being enabled is within the `psoID` element. The `ID` attribute provides the user name. The user attribute in the `targetID` attribute identifies the capability changed.

```
<spml2:psoID ID="suspend1" targetID="Identity:User"/>
```

<activeRequest>

Operations using the `activeRequest` element queries a user status in Select Identity or in a specified service to determine if that user's membership is enabled or disabled.

The following is a partial example of an `activeRequest`. The `targetID` identifies the user target.

```
<spml2:activeRequest xmlns:spml2="urn:oasis:names:tc:SPML:2:0:suspend"
xmlns:hpsi="urn:hp:si:spml20:common"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" requestID="ac1101">
```

The `activeRequest` allows the administrator to determine the status of a user service membership.

The Select Identity ID of the user whose account is being enabled is within the `psoID` element. The `ID` attribute provides the user name. The `targetID` attribute identifies the service or user target.

```
<spml2:psoID ID="suspend1" targetID="Service1"/>
<spml2:psoID ID="suspend1" targetID="Identity:User"/>
```

<setPasswordRequest>

This is a an administrator and self-service operation where users change their own passwords or an administrator changes a user's password using `setPasswordRequest`.

The following is a partial example of a `setPasswordRequest`. The `targetID` provides the attribute for the attribute target. A new password is mandatory in this request.

```
<spmlpwd:setPasswordRequest xmlns:spml="urn:oasis:names:tc:SPML:2:0"
xmlns:spmlpwd="urn:oasis:names:tc:SPML:2:0:password"
xmlns:hp="urn:hp:si:spml20:common"
```

```
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" requestID="S1001"
  executionMode="asynchronous">
```

The `setPasswordRequest` allows users to manually set a new password for themselves. The `requestID` of the provisioning transaction identifies the transaction for asynchronous transactions.

The user name and password of the user or administrator authorized to perform a password reset is in the `Security` element in the `soap:Header` element of the message.

The Select Identity ID of the user whose password is being set is within the `psoID` element. The `ID` attribute provides the user name. The `targetID` attribute contains the Select Identity password attribute name. All resources mapped to this Select Identity password attribute are changed with this request. If the specified attribute is designated as the Select Identity login attribute, then the login password for the user is also changed.

```
<spmlpwd:psoID ID="setPassword1" targetID="Attribute:Password">
</spmlpwd:psoID>
```

The following `Password` element identifies the user's current password in the system, and is ignored during the `setPasswordRequest` as the requestor is authenticated using the login information in the `soap:Header`. The new value for the user's password is set in the `currentPassword` element.

```
<spmlpwd:password>thisvalueisignored</spmlpwd:password>
<spmlpwd:currentPassword>newPassword</spmlpwd:currentPassword>
```

The typical flow of a set or reset operation is as follows:

1. The administrator sends a `lookupRequest` with the `username` to Select Identity. The response returns the resources that user is a member of along with the mapped Select Identity password attribute names.
2. The administrator sends a `listTargetsRequest` using "Attribute:<password attribute name>" to get the password schema.
3. The client formulates a set or reset password request based on the returned schema in the `listTargetsResponse`.
4. The client sends a `resetPasswordRequest`.
5. Select Identity returns the status (success=completed request, pending=accepted request and pending completion) in the `resetPasswordResponse`.
6. The client sends a `statusRequest` with the `requestID` from the `resetPasswordResponse`.
7. Select Identity responds with the status (success).

<resetPasswordRequest>

This is an administrator operation. This request resets a user's password using `resetPasswordRequest`.

The following is a partial example of a `resetPasswordRequest`. The `targetID` provides the attribute target.

```
<spmlpwd:resetPasswordRequest xmlns:spml="urn:oasis:names:tc:SPML:2:0"
xmlns:spmlpwd="urn:oasis:names:tc:SPML:2:0:password"
xmlns:hp="urn:hp:si:spml20:common" xmlns:xs="http://www.w3.org/2001/
XMLSchema-instance" requestID="R1004" executionMode="asynchronous">
```

The `resetPasswordRequest` generates a new password for a user. The `requestID` of the provisioning transaction identifies the transaction for asynchronous transactions.

The user name and password of the user or administrator authorized to perform a password reset is in the `Security` element in the `soap:Header` element of the message.

The Select Identity ID of the user whose password is being reset is within the `psoID` element, the `targetID` attribute contains the Select Identity password attribute name as shown in the following snippets. All resources mapped to this Select Identity password attribute are reset with this request. If the specific attribute is designated as the Select Identity login attribute, then the login password for the user is also changed. The password attribute contained in the `targetID` attribute generates a password if the attribute is associated with `AutoGenerateOnReset=true` in the attribute definitions.

```
<spmlpwd:psoID ID="resetPassword1" targetID="Attribute:Password"></
spmlpwd:psoID>
```

Name a specific resource so that the user's password is reset only on that resource. The following snippet shows a password reset on a single resource.

```
<spmlpwd:psoID ID="resetPassword1" targetID="Resource:spml20auth"></
spmlpwd:psoID>
```

Asynchronous Requests

The following requests are treated by Select Identity as asynchronous requests as Select Identity creates an internal request and returns a response with status=pending. To query the completion status, the requestor sends a `statusRequest`.

- `addRequest`
- `modifyRequest`
- `deleteRequest`
- `suspendRequest`
- `resumeRequest`
- `setPasswordRequest`
- `resetPasswordRequest`

The asynchronous flow is as follows:

1. The RA sends one of the above asynchronous requests to Select Identity.
2. If synchronous mode is specified for any of the above requests, Select Identity returns an error response.
   - If the mode is not specified, Select Identity assumes asynchronous mode as it is the only mode supported for the above requests.
   - If the `requestId` is not provided in the request, Select Identity generates one.
3. Select Identity creates an internal request for processing one of the above asynchronous requests (for example, a Select Identity request is created).

4 Select Identity returns a response (failure or pending). Select Identity returns the requestor's `requestID` in the response. If the `requestID` is not available, then a Select Identity generated `requestID` is returned in the response.

The `requestID` returned in the response is different from an internal Select Identity `requestID` that is part of an internal Select Identity request.

5 The requestor queries the status of the asynchronous request using the `requestID` that is returned in the response.

<statusRequest>

The `statusRequest` requires a `requestID` from the original request you are querying. If a `requestID` was not included in the original request, then Select Identity automatically generates one and returns it in the request response. The `requestID` must be provided in the `statusRequest` to identify the original request action. The `statusRequest` acts upon any asynchronous request. The response returns a status of pending, failure, or success.

```
<spml2:statusRequest xmlns:spml2="urn:oasis:names:tc:SPML:2:0:async"
xmlns:hpsi="urn:hp:si:spml20:common" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" requestID="DS1101" asyncRequestID="D1101">
</spml2:statusRequest>
```

The following is a typical flow showing `statusRequest`.

1 The Web Service client sends the `statusRequest` to Select Identity with the requestor name and the `requestID` from the targeted request.

2 Select Identity responds with the status (success).

<cancelRequest>

As in the `statusRequest`, a `cancelRequest` requires a `requestID` from the original request you are querying. If a `requestID` was not included in the original request, then Select Identity automatically generates one and returns it in the request response. The `requestID` must be provided in the `statusRequest` to identify the original request action. The `cancelRequest` acts upon any asynchronous request. The response returns a status of "failure" or "success."

```
<spml2:cancelRequest xmlns:spml2="urn:oasis:names:tc:SPML:2:0:async"
xmlns:hpsi="urn:hp:si:spml20:common" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" requestID="DS1101" asyncRequestID="D1101">
</spml2:cancelRequest>
```

The following is a typical flow showing `cancelRequest`.

1 The Web Service client sends the `cancelRequest` to Select Identity with the requestor name and the `requestID` from the targeted request.

2 Select Identity terminates the targeted request and responds with the status (success=action completed or failure=action failed).

Request Responses

Responses follow the same general structure. The data element varies depending upon the type of request and the context of the request. The following is an example of an `addResponse`.

```
<addResponse status="pending" requestID="RA4380"
xmlns="urn:oasis:names:tc:SPML:2:0">
```

The reponse returns with a status attribute that displays either “failure” or “pending” for asynchronous operations. If the `requestID` attribute isn’t provided in the request, then a `requestID` is autogenerated in the response. The `requestID` can be used to identify the status using any other requests.

The `<pso>` element contains the identifying information about the user and the namespaces used. The `psoID` element contains the user name in the `ID` attribute.

```
<pso>
   <psoID ID="johnsmith" xmlns:xsi="http://www.w3.org/2001/
   XMLSchema-instance" xmlns:hpsi="urn:hp:si:spml20:common"
   xmlns:spml2="urn:oasis:names:tc:SPML:2:0"/>
```

The `data` element contains the returned data from the request. This example contains the `requestID`, and the `ServiceName` identifies the service used. The `UserInfo` element contains the profile attributes of the user.

```
      <data xsi:type="urn:RequestStatusType"
      xmlns:urn="urn:hp:si:spml20:common" xmlns:xsi="xsi">
         <RequestID xmlns="urn:hp:si:spml20:common">4380</RequestID>
         <ServiceName>spml20 test</ServiceName>
         <UserInfo>
            <FirstName>John</FirstName>
            <LastName>Smith</LastName>
            <Email>john.smith@hp.com</Email>
            <UserName>johnsmith</UserName>
         </UserInfo>
```

The `RequestorID` element is the name of the administrator making this request. The `RequestType` identifies the request as delegated or self service. The `Status` element shows where in the process this response occurred and what status the request has.

```
         <RequestorID>sisa</RequestorID>
         <RequestType>
            <Value>Delegated Request</Value>
         </RequestType>
         <Status>
            <Value>In Process</Value>
         </Status>
      </data>
   </pso>
```

Self-Service Requests

Web services requests can be used for self-service operations. Self service requests are submitted by a user on his or her own behalf. Self service requests are identical to delegated requests in every regard except for the access level and permissions granted to the user. Self service accounts normally have a lower level of access and administration privileges than does an administrator or a user who makes delegated requests.

Users who originate their own `addRequest` for secondary user IDs must have the *Add Account* permission in their role on the service where the account is to be added.

WSDL

Web services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). In the context of Select Identity, WSDL operates in conjunction with SOAP 1.1 to point to the endpoints within the Select Identity installation. Access to the schemas through WSDL is allowed only after authentication.

WSDL is accessed through the following steps

1. Paste the following URL into your browser: **http://localhost:7001/lmz/schema/wsdls**. Select Identity displays the login page.
2. Enter an administrator name and password and click **Sign In**. The Select Identity Published web services page displays.
3. Click **spml20.wsdl** under the File Name: column.
   The WSDL schema displays in the browser window.

   The `location` attribute of the `soap:address` element provides the SPML 2.0 endpoint URL to which requests will be sent.

The endpoint URL is different for SMPL 1.0 and SPML 2.0 and allows both systems to co-exist on the same installation.

The location of each supporting schema is presented in a separate `wsdl:import` element.

All schemas may be reached through the browser using the URL **http://localhost:7001/lmz/schema/xsd/** and specifying the file as follows.

Core	spml20core
Password	spml20password
Aysnc	spmlasync
Suspend	spml20suspend
Select Identity	spml20sicommon

Valid XML requests may be built from the elements discovered within these standard schemas.