One Identity Active Roles 7.3

SPML Provider Administration Guide

Legend

WARNING: A WARNING icon indicates a potential for property damage, personal injury, or death.

CAUTION: A CAUTION icon indicates potential damage to hardware or loss of data if instructions are not followed.

IMPORTANT, NOTE, TIP, MOBILE, or VIDEO: An information icon indicates supporting information.

Active Roles SPML Provider Administration Guide
Updated - May 2019
Version - 7.3

Contents

1

Understanding Active Roles SPML Provider

Active Roles SPML Provider is designed to exchange the user, resource, and service provisioning information between SPML-enabled enterprise applications and Active Directory.

Active Roles SPML Provider supports the Service Provisioning Markup Language Version 2 (SPML v2), an open standard approved by the Organization for the Advancement of Structured Information Standards (OASIS). SPML - is an XML-based provisioning request-and-response protocol that provides a means of representing provisioning requests and responses as SPML documents. The use of open standards provides the enterprise architects and administrators with the flexibility they need when performing user management and user provisioning in heterogeneous environments.

Features

The key features of Active Roles SPML Provider are as follows:

- **Support for two operation modes**: SPML Provider can be configured to operate in *proxy mode* or in *direct access mode*. In proxy mode, SPML Provider accesses Active Directory or Active Directory Lightweight Directory Services (AD LDS, formerly known as ADAM) through Active Roles used as a proxy service, while in direct access mode, SPML Provider directly accesses Active Directory or AD LDS.
- **Support for equivalent LDAP operations:** SPML Provider can perform equivalent LDAP operations such as addRequest, modifyRequest, deleteRequest, and lookupRequest.
- **Support for Azure AD, AD, and AD LDS data management:** SPML Provider enables SPML-conformant applications to read from and write to Azure AD, Active Directory (AD), and AD LDS.
- **Search Capability support:** SPML Provider allows SPML-enabled applications to search for relevant directory objects based on various search criteria.

- **Password Capability support:** SPML Provider allows SPML-enabled applications to perform basic password management tasks such as setting and expiring user passwords.
- **Suspend Capability support**: SPML Provider allows SPML-enabled applications to effectively enable, disable and deprovision user accounts in Active Directory.
- **Flexible Configuration options:** There is support for many different configuration options that enable the administrator to adjust the behavior and optimize the SPML Provider performance.
- **IIS Security Support:** SPML Provider supports all IIS security configurations, including integrated Windows authentication, basic authentication, and basic authentication over Secure Sockets Layer (SSL).
- **Support for using Active Roles controls**: In proxy mode, you can send Active Roles controls to the Active Roles Administration Service with an SPML request to perform an administrative operation. In your request, you can also define the Active Roles controls that the Administration Service must return in the SPML response.

Use scenarios

SPML Provider can be used for a variety of purposes. Some common scenarios for using SPML Provider are as follows:

- **Non-Windows applications:** The systems running non-Windows applications that need to communicate with Active Directory can do this through SPML Provider. For example, with SPML Provider, Unix applications can manage Unix-enabled user accounts in Active Directory. In proxy mode, SPML Provider allows existing SPML-compatible provisioning systems, such as SUN Java System Identity Manager and IBM Tivoli Directory Integrator to take advantage of the functionality of Active Roles.
- **Web services:** The use of directories in Web services is growing rapidly. Additionally, XML is becoming the default language for use with Web services. SPML Provider fills the gap between XML documents and Active Directory services, enabling applications that must provide or use Web services to communicate with Active Directory.
- **Handheld and portable devices:** Data-enabled cell phones or PDAs that need an access to directory data may not contain a client for the ADSI LDAP Provider but might be able to use the SPML communication protocol to access Active Directory over the Internet.
- **Firewall access:** Certain firewalls cannot pass LDAP traffic because they cannot audit it, but these firewalls can pass XML. In such cases, applications can use SPML Provider to communicate with Active Directory across a firewall.

Basic concepts and definitions

Active Roles SPML Provider operates based on the concepts defined in SPML v2. This section introduces and describes these key concepts and definitions as applied to SPML Provider.

A **Client** (Requesting Authority or Requestor) is any SPML-compliant application that sends well-formed SPML requests to the Active Roles SPML Provider and receives responses from it. Clients can include various business applications, such as human resources (HR) databases or Identity Management systems. There is no direct contact between a client and the target (Active Roles or an Active Directory server).

Active Roles SPML Provider (Provisioning Service Provider or PSP) is a Web service that uses the Simple Object Access Protocol (SOAP) over HTTP for communications. SPML Provider can directly access Active Directory data or communicate with Active Directory using the Active Roles proxy service. SPML Provider acts as an intermediary between a client and the target (Active Directory domain controller or Active Roles).

In proxy mode, **Active Roles** represents the Provisioning Service Target (or Target) that is available for provisioning actions through SPML Provider. The target has a unique identifier (targetID) that is maintained by SPML Provider and is used in a request or a response.

AD Objects (Provisioning Service Objects or PSO) represent directory objects that SPML Provider manages. A client can add, delete, modify, or look up a directory object. Each object has a unique identifier (PSO ID). In SPML Provider, an object DN is used as a PSO ID.

NOTE: A Requestor, Provisioning Service Provider, Provisioning Service Target, and Provisioning Service Objects are key notions described in the official SPML v2 specification.

For detailed information on the concepts defined in SPML v2, see Section 2 "Concepts" of the OASIS SPML v2 specification, available for download at http://www.oasis-open.org/specs/index.php#spmlv2.0.

How SPML Provider works

With SPML Provider, applications can use SPML documents to look up, retrieve and update directory data in Active Directory, Azure AD, and AD LDS. SPML Provider converts XML elements and attributes into commands used to make changes to Active Directory and retrieve data from Active Directory. SPML Provider can also convert the response received from Active Roles or Active Directory to XML format. These conversions are based on and are in compliance with the OASIS SPML v2 - DSML v2 Profile specification.

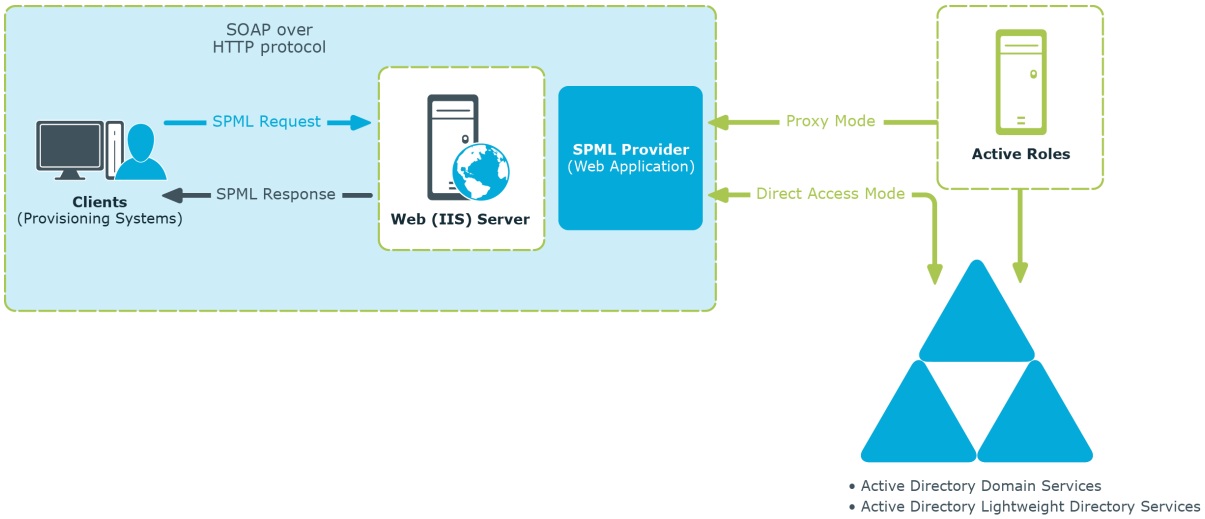
SPML Provider runs as a Web application on a Web server running Microsoft Internet Information Services (IIS), and uses SOAP over HTTP to transmit and receive directory requests from client computers.

The SPML Provider environment includes the following components:

- **Clients that use SPML v2**: These clients are applications that manage directory objects (for example, user accounts). A client issues SPML requests that describe operations to be performed on the directory object and send these requests to SPML Provider.
- **SPML Provider**: Receives and processes client requests, and returns a response to the client.
- **Active Roles:** In proxy mode, this is the endpoint for provisioning requests and the actual software that manages directory objects.
- **Active Directory, Azure AD, or AD LDS**: In proxy mode, SPML Provider can access Active Directory or Azure AD domains and AD LDS instances that are registered with Active Roles as managed domains, Azure AD tenants, and managed AD LDS instances, respectively. In direct access mode, SPML Provider can access the domain controller or the AD LDS instance defined in the `SPML.Config` file. For more information, see "Configuring SPML Provider" later in this document.

The following diagram illustrates the flow of requests and responses through the SPML Provider environment components:

Figure 1: Flow of requests and responses through the SPML Provider environment components



As shown in the diagram, the client/SPML Provider communications are based on the simple request/response protocol.

In proxy mode, SPML Provider works in the following way:

1. A client issues a well-formed SPML request using the SOAP over HTTP protocol. This request goes to a server running IIS, where it is routed to SPML Provider.
2. SPML Provider examines the request for conformance to the SPML format.
3. If the request complies with the SPML format, the SPML Provider submits the request to Active Roles. Based on the client request, Active Roles retrieves or modifies data in Active Directory, Azure AD, or in AD LDS.

4. After performing the requested operation, Active Roles sends the result of the operation back to SPML Provider.
5. SPML Provider then processes this result data and sends the result of the performed operation back to the client in the form of an SPML response.

In direct access mode, SPML Provider works in the following way:

1. A client issues a well-formed SPML request using the SOAP over HTTP protocol. This request goes to a server running IIS, where it is routed to SPML Provider.
2. SPML Provider examines the request for conformance to the SPML format.
3. If the request conforms to the SPML format, SPML Provider retrieves or modifies the relevant data in Active Directory or in AD LDS (ADAM).
4. SPML Provider sends the result of the performed operation back to the client in the form of an SPML response.

If the client request does not conform to the SPML format, the client receives an SPML response that describes the encountered error.

System requirements

Before installing the Active Roles SPML Provider, ensure your system meets the following minimum hardware and software requirements.

Hardware requirements

Ensure that the following hardware requirements are met:

- 1 GHz or higher Intel Pentium-compatible CPU.
- At least 1 GB of RAM.
- At least 100 MB of free disk space.

Software requirements

Ensure that the following software requirements are met:

- Microsoft Windows Server 2008 R2 SP 1, Microsoft Windows Server 2012, Microsoft Windows Server 2012 R2, or Microsoft Windows Server 2016 operating system.
- Microsoft .NET Framework 4.6.2.
- Microsoft Internet Information Services (IIS). For proxy mode, the IIS server must be part of an Active Directory forest where Active Roles is deployed.
- For proxy mode, Active Roles Administration Service 7.3 is required.

TIP: If you choose the proxy mode, for performance reasons, we recommend that you install the Active Roles SPML Provider on the computer running the Active Roles Administration Service.

Web Server requirements

Windows Server 2008 R2

On a Windows Server 2008 R2 based computer, ensure that the **Web Server (IIS)** server role with the following role services is installed:

- Web Server/Common HTTP Features/
  - Static Content
  - Default Document
  - HTTP Errors
  - HTTP Redirection
- Web Server/Application Development/
  - ASP.NET
  - .NET Extensibility
  - ASP
  - ISAPI Extensions
  - ISAPI Filters
- Web Server/Security/
  - Basic Authentication
  - Windows Authentication
  - Request Filtering
- Management Tools/IIS 6 Management Compatibility/
  - IIS 6 Metabase Compatibility

Use Server Manager to add the required role, role services, and features.

Windows Server 2012

On a Windows Server 2012 or Windows Server 2012 R2 based computer, ensure that the **Web Server (IIS)** sever role is installed, including:

- Web Server/Common HTTP Features/
  - Default Document
  - HTTP Errors
  - Static Content
  - HTTP Redirection

- Web Server/Security/
    - Request Filtering
    - Basic Authentication
    - Windows Authentication
- Web Server/Application Development/
    - .NET Extensibility 4.6
    - ASP
    - ASP.NET 4.6
    - ISAPI Extensions
    - ISAPI Filters
- Management Tools/IIS 6 Management Compatibility/
    - IIS 6 Metabase Compatibility

Windows Server 2016

On a Windows Server 2016 based computer, ensure that the **Web Server (IIS)** sever role is installed, including:

- Web Server/Common HTTP Features/
    - Default Document
    - HTTP Errors
    - Static Content
    - HTTP Redirection
- Web Server/Security/
    - Request Filtering
    - Basic Authentication
    - Windows Authentication
- Web Server/Application Development/
    - .NET Extensibility 4.6
    - ASP
    - ASP.NET 4.6
    - ISAPI Extensions
    - ISAPI Filters
- Management Tools/IIS 6 Management Compatibility/
    - IIS 6 Metabase Compatibility

Use Server Manager to add the required role, role services, and features.

Feature Delegation

Configure Internet Information Services (IIS) to provide **Read/Write** delegation for the following features:

- Handler Mappings
- Modules

Use **Feature Delegation** in Internet Information Services (IIS) Manager to verify that these features have delegation set to **Read/Write**.

Installing Active Roles SPML Provider

To install Active Roles SPML Provider

1. Log on to the computer on which you want to install Active Roles SPML Provider.
2. Navigate to the network location of the Active Roles SPML Provider installation files.
3. Start the Active Roles SPML Provider Installation Wizard by double-clicking `Setup.exe`.
4. On the **Welcome** page, click **Next**.
5. On the **License Agreement** page, click **I accept the license agreement,** and then click **Next.**
6. On the **User Information** page, enter the required user information, and then click **Next.**
7. On the **Select Features** page, optionally, click **Browse** to specify a new installation folder. Click **Next**.
8. On the **Access to Active Directory** page, specify how you want SPML Provider to access Active Directory. The following options are available:
   - **Access through Active Roles (local Administration Service)** SPML Provider accesses Active Directory through the Active Roles Administration Service running on the computer where you install SPML Provider.
   - **Access through Active Roles (specified Administration Service)** SPML Provider accesses Active Directory through the Active Roles Administration Service on a different network computer. Type the fully qualified domain name of that computer in the **Administration Service on** text box.
   - **Direct access (local domain controller)** SPML Provider directly accesses Active Directory using domain controller running on the computer where you install SPML Provider.
   - **Direct access (specified domain controller)** SPML Provider directly accesses Active Directory using domain controller running on a different network computer. Type the name of that domain controller in the **Domain controller name** text box.

9. On the **Ready to Install the Application** page, click **Next** to begin installation.
10. Click **Finish** to complete the installation.

The SPML Provider Installation Wizard creates a virtual directory under Default Web Site, with the name of the virtual directory set to ARServerSPML. This enables clients to access SPML Provider by using the following URL:
http://<HostName>/ARServerSPML/SPMLProvider.asmx.

NOTE: The Installation Wizard installs SPML Provider and several sample HTML pages that demonstrate various SPML v2 operations.

Configuring Active Roles SPML Provider

Configuration settings allow the administrator to configure SPML Provider and its schema in order to adjust the SPML Provider behavior. Administrators can, for example, specify the required managed objects and attributes in the schema, or choose the type of execution (disabling or deprovisioning objects) for the Suspend operation.

Configuration settings in SPML.Config

The SPML Provider configuration settings can be found in the SPML.Config file located in the **Web** sub-folder of the SPML Provider installation folder. The SPML.Config file contains data in the XML format. You can open and edit the configuration file with a common text editor such as Notepad.

NOTE: After you modify configuration settings, the IIS application pool for the SPML Provider Web site must be restarted in order for the changes to take effect.

The following table describes the XML elements used in the SPML Provider configuration file.

Table 1: XML elements used in the SPML Provider configuration file

Element	Parent element	Description
service	configuration	In proxy mode, specifies the name of the computer running the Active Roles Administration Service. In direct access mode, specifies the name of the AD domain controller or AD LDS server. The name of the AD LDS server must be in the form *<servername:portnumber>*.
adsiProvider	configuration	Specifies the progID of the ADSI Provider. In proxy

Element	Parent element	Description
		mode, the progID is EDMS. In direct access mode, the progID is LDAP.
schemaFile	configuration	Contains the name of the file that defines the DSML Profile schema for SPML Provider. By default, the file name is SPMLSchema.Config. The schema file must be located in the same folder as the SPML.Config file.
defaultMaxSelect	search	Specifies the maximum number of search results that SPML Provider can return without page splitting. The default value is 1000.
pageSize	search	Specifies the maximum number of search results per page. The default value is 25. NOTE: If **pageSize** is set to **0**, SPML Provider returns search results without page splitting.
class	password	Contains the LDAP display name of the schema class of objects on which SPML Provider is expected to perform the Password Capability-related operations such as **setPassword** and **expirePassword**.
class	suspend	Contains the LDAP display name of the schema class of objects on which SPML Provider is expected to perform the Suspend Capability-related operations such as **suspend**, **resume**, and **active**.
suspendAction	suspend	Possible values: disable or deprovision. The default value is disable. If **suspendAction** is set to **disable**, SPML Provider disables the specified user account on the target. If **suspendAction** is set to **deprovision**, SPML Provider deprovisions the specified user account in accordance with the deprovisioning policies defined by Active Roles.
checkOutput	configuration	Possible values: true or false. The default value is false. **true** causes SPML Provider to check the string attribute values retrieved from the underlying directory before adding them to a response. If an attribute value contains illegal characters that could break the XML parser on the client side, SPML Provider converts the attribute value to the base64binary format and then adds the result of the conversion to

Element	Parent element	Description
		the response. Note that this option may result in performance degradation of SPML Provider as checking every attribute value is a resource-intensive operation. **false** causes SPML Provider not to check the string attribute values retrieved from the underlying directory. An attribute value is added to the response without any conversion even if the value contains illegal characters. NOTE: In accordance with the XML specification, the legal character range is as follows: #x9 | #xA | #xD | [#x20-#xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF]. With **checkOutput** set to **true**, SPML Provider ensures that attribute values in a response contain only characters from the legal character range.

Sample configuration file

The following is an example of the configuration file for SPML Provider configured to operate in proxy mode. If SPML Provider and the Active Roles Administration service are installed on the same computer, the default configuration settings look as follows:

```
<?xml version="1.0"?>
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="urn:quest:names:SPMLProvider">
<service>localhost</service>
<adsiProvider>EDMS</adsiProvider>
<schemaFile>SPMLSchema.Config</schemaFile>
<capabilities>
<search>
<defaultMaxSelect>1000</defaultMaxSelect>
<pageSize>25</pageSize>
</search>
<password>
<appliesTo>
<class>user</class>
</appliesTo>
```

```
</password>
<suspend>
<appliesTo>
<class>user</class>
</appliesTo>
 <suspendAction>disable</suspendAction>
</suspend>
</capabilities>
<checkOutput>false</checkOutput>
</configuration>
```

Extending the SPML Provider schema

The SPML Provider schema defines the XML structure of the objects and attributes that SPML Provider manages. You can modify the schema to manage new types of objects or object properties. Thus, you can add the class and attribute definitions to the schema in order to meet the needs of your organization.

NOTE: In proxy mode, you can add only those object classes and attributes that are valid according to the Active Roles schema.

The SPML Provider schema is stored in the SPMLSchema.Config file. The SPMLSchema.Config file is located in the **Web** sub-folder of the SPML Provider installation folder.

The schema format corresponds to the DSML Version 2 profile (DSMLv2). For detailed information on the DSML v2 profile, refer to the OASIS SPML v2 - DSML v2 Profile specification. The specification describes the use of the DSML protocol as a data model for SPML- based provisioning and can be accessed from the OASIS Web site at http://www.oasis-open.org/specs/index.php#spmlv2.0.

Using Active Roles SPML Provider

To access SPML Provider, use the following URL:

`http://<HostName>/ARServerSPML/SPMLProvider.asmx`

where the <*HostName*> stands for the name of the computer where SPML Provider is installed.

NOTE: The SPML Provider Web service is described by a Web Services Description Language (WSDL) file. To obtain a WSDL description of SPML Provider, navigate to http://<*HostName*>/ARServerSPML/SPMLProvider.asmx?WSDL.

Operation mode

SPML Provider can be configured to operate in:

- **Proxy mode** In this mode, SPML Provider accesses Active Directory, Azure AD, or AD LDS using the Active Roles proxy service. In proxy mode, SPML Provider extends Active Roles. Because SPML Provider uses open standards such as HTTP, XML, and SOAP, a greater level of interoperability with Active Roles is possible than is available with the Active Roles ADSI Provider.
- **Direct access mode** In this mode, SPML Provider directly accesses Active Directory, Azure AD, or AD LDS.

In proxy mode, SPML Provider can manage objects in Active Directory domains and AD LDS instances that are registered with Active Roles as managed domains and managed AD LDS instances, respectively. In direct access mode, SPML Provider can manage only objects in the domain or AD LDS instance to which SPML Provider is connected using the configuration setting such as the domain controller or AD LDS server.

TIP: To take advantages of the powerful functionality of Active Roles, we recommend that you use proxy mode whenever possible

Support for Active Roles controls

Active Roles implements special parameters called Active Roles controls (hereafter *controls*). The controls allow you to customize request processing.

In proxy mode, SPML Provider clients can send controls to the Active Roles Administration Service with an SPML request to perform an administrative operation. The Administration Service can process the controls. On the other hand, the Administration Service can return its own control to the SPML Provider client, and then the client can process that control. The controls a client sends to the Administration Service are referred to as *InControls* whereas the controls the Administration Service returns to the client are referred to as *OutControls*.

This section covers the following subjects:

- Sending the InControl-type controls to the Active Roles Administration Service with an SPML request.
- Specifying a set of the OutControl-type controls that the Active Roles Administration Service will return with an SPML response.

For more information about Active Roles controls and for the list of available built-in controls, see Active Roles SDK.

IMPORTANT: All elements described in this section must be defined at the beginning of your SPML request. For a sample of use, see later in this document.

Sending controls to the Active Roles Administration Service

This section covers the `controls` and `control` XML elements that your SPML request must include to send controls to the Active Roles Administration Service.

Element name: `controls`

Element description: Specifies a collection of InControl-type controls to send to Administration Service.

Child elements: `control`

Attributes:

Table 2: Controls attributes

attribute name	attribute description
xmlns	Declares the namespase for all child elements of the `controls` element. This attribute must be set to `quest:ars:SPML:2:0`

Element name: `control`

Element description: Describes a control to send to the Administration Service.

Parent elements: `controls`

Child elements: None

Attributes:

Table 3: Control attributes

attribute name	attribute description
name	Specifies the name of the control.

The control value in the `control` element body must be specified as follows:

`<control name=%control name%>%control value%</control>`

To send an empty control, use the following syntax:

`<control name=%control name% />`

Specifying controls to return to the SPML Provider client

This section covers the `controlsForOutput` and `control` XML elements that your SPML request must include to specify a set of controls to return to the SPML Provider client.

Element name: `controlsForOutput`

Element description: Specifies a collection of OutControl-type controls to return to SPML client.

Child clements: `control`

Attributes:

Table 4: Attributes for controlsForOutput

attribute name	attribute description
xmlns	Declares the namespase for all child elements of the `controls` element. This attribute must be set to `quest:ars:SPML:2:0`

Element name: `control`

Element description: Describes a control to return to SPML Provider client with an SPML response.

Parent elements: `controlsForOutput`

Child elements: None

Attributes:

Table 5: Attributes for control

attribute name	attribute description
name	Specifies the name of the control.

The `control` elements used to specify controls to return with SPML response must be defined as follows:

```
<control name=%control name% />
```

Sample SPML request

This section provides a sample SPML request and the SPML response that illustrate how to use Active Roles controls in your SPML requests.

This sample shows how an SPML Provider client can send a request to modify the specified user object. With this request, the client sends the `AllowApproval` built-in control set to `Confirm`, and the `CustomControl` control set to `MyCustomValue`. The request also contains the `controlsForOutput` element, which specifies that Active Roles Administration service will return values of the `OperationStatus` and `CustomControl` controls in the SPML response.

TIP: For more information about the use of the `AllowApproval` and `OperationStatus` controls, refer to the Active Roles SDK.

NOTE: You need to modify the sample SPML request in order to adjust it to your environment. Before using this sample, set the `ID` attribute of the `psoID` element to the distinguished name of the user account you want to modify.

SPML request

```
<?xml version="1.0"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<spml:modifyRequest xmlns:spml="urn:oasis:names:tc:SPML:2:0">
<controls xmlns="quest:ars:SPML:2:0">
<control name="AllowApproval">Confirm</control>
   <control name="CustomControl">MyCustomValue</control>
</controls>
<controlsForOutput xmlns="quest:ars:SPML:2:0">
    <control name="OperationStatus"/>
```

```
<control name="CustomControl"/>
</controlsForOutput>
  <spml:psoID ID="CN=JDOE,OU=Users,DC=mycompany,DC=com"/>
<spml:modification>
 <modification name="description" operation="replace"
xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>New description</value>
</modification>
</spml:modification>
</spml:modifyRequest>
</soap:Body>
</soap:Envelope>
```

SPML response

```
<?xml version="1.0" encoding="UTF-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<modifyResponse status="success" xmlns="urn:oasis:names:tc:SPML:2:0">
<controls xmlns="quest:ars:SPML:2:0">
<control name="OperationStatus">Completed</control>
<control name="CustomControl">ReturnedValue</control>
</controls>
<pso>
<psoID ID="CN=JDOE,OU=Users,DC=mycompany,DC=com"/>
<data>
<attr name="cn" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value xsi:type="xsd:string">Admin1</value>
</attr>
<attr name="objectClass" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value xsi:type="xsd:string">top</value>
<value xsi:type="xsd:string">person</value>
<value xsi:type="xsd:string">organizationalPerson</value>
<value xsi:type="xsd:string">user</value>
</attr>
```

```
<attr name="objectCategory" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value
xsi:type="xsd:string">CN=Person,CN=Schema,CN=Configuration,DC=dom,DC=lab,DC=local</val
ue>
</attr>
<attr name="objectGUID" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value xsi:type="xsd:base64Binary">Aodvua6TAE+Ja9O3vnRntg==</value>
</attr>
<attr name="description" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value xsi:type="xsd:string">New description</value>
</attr>
</data>
</pso>
</modifyResponse>
</soap:Body>
</soap:Envelope>
```

Supported Azure Features

- Active Roles 7.3 SPML Provider supports Azure user, group, and contact creation.

NOTE: You must complete Azure AD configuration, before using SPML for user, group, and contact creation in Azure AD. For more information, see *Azure AD and Office 365 Management Administrator Guide*.

Sample SPML request for Azure user, group, and contact creation

Sample SPML request for Azure User Creation

```
<?xml version="1.0"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<addRequest xmlns="urn:oasis:names:tc:SPML:2:0" returnData="everything">
<containerID ID="OU=AzureOU, DC=Sample,DC=local,DC=com"/>
<data>
```

```
<attr name="cn" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>AzureUser</value>
</attr>
<attr name="description" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>Azure test user</value>
</attr>
<attr name="sAMAccountName" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>AzureUser</value>
</attr>
<attr name="objectClass" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>user</value>
</attr>
<attr name="mail" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>AzureUser@ARStestdev.onmicrosoft.com</value>
</attr>
<attr name="otherHomePhone" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>12135555555</value>
<value>12134444444</value>
</attr>
<attr name="edsaPassword" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>P@ssw0rd123</value>
</attr>
<attr name="edsaAccountIsDisabled" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>FALSE</value>
</attr>
<attr name="userPrincipalName" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>AzureUser@ARStestdev.onmicrosoft.com</value>
</attr>
<attr name="edsvaAzureOffice365Enabled" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>TRUE</value>
</attr>
<attr name="edsaAzureUserPrincipalName" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>AzureUser@ARStestdev.onmicrosoft.com</value>
</attr>
<attr name="edsaAzureUserAccountEnabled" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>TRUE</value>
```

```
</attr>
<attr name="edsaAzureUserDisplayName" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>AzureUser</value>
</attr>

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

Sample SPML request for Azure Group Creation.

```
<?xml version="1.0"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<addRequest xmlns="urn:oasis:names:tc:SPML:2:0" returnData="everything">
<psoID ID="CN=GroupName,OU=AzureOU,DC=Sample,DC=local,DC=com"/>
<data>
<attr name="objectClass" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>group</value>
</attr>
<attr name="description" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>My test group</value>
</attr>
<attr name="mailEnabled" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>false</value>
</attr>
<attr name="mail" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value> GroupName@company.com</value>
</attr>
<attr name="mailNickName" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value> GroupName</value>
</attr>
<attr name="edsvaAzureOffice365Enabled" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>TRUE</value>
</attr>
```

```
<attr name="edsaAzureGroupDisplayName" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value> GroupName</value>
</attr>
<attr name="edsaEstablishGroupEmail" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>false</value>
</attr>
<attr name="edsaAzureGroupType" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>-2147483646</value>
</attr>
</data>
</addRequest>
</soap:Body>
</soap:Envelope>
```

Sample SPML request for Azure Contact Creation

```
<?xml version="1.0"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<addRequest xmlns="urn:oasis:names:tc:SPML:2:0" returnData="everything">
<containerID ID="OU=AzureOU,DC=Sample,DC=local,DC=com"/>
<data>
<attr name="cn" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>AzureContact</value>
</attr>
<attr name="description" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>AzureContact</value>
</attr>
<attr name="objectClass" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>Contact</value>
</attr>
<attr name="edsvaAzureOffice365Enabled" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>TRUE</value>
</attr>
<attr name="edsaAzureContactEmail" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>AzureContact@test.com</value>
```

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

Supported operations

SPML Provider implements the SPML v2 core protocol and supports core operations that are required for conformance to the official SPML v2 specification. The following table lists the core operations supported by SPML Provider.

Table 6: Core operations supported by SPML Provider

Operation	Description
listTargets	Lists targets available for provisioning through SPML Provider and the SPML Provider's supported set of capabilities for targets.
add	Creates a new object on the target.
modify	Changes the specified object on the target.
lookup	Obtains the XML that represents the specified object on the target.
delete	Removes the specified object from the target.

In addition to core operations required for conformance to the SPML v2 specification, SPML Provider supports a set of optional operations (Capabilities) that are functionally related. The following tables list the Capabilities supported by SPML Provider.

Search capability

Table 7: Capabilities supported by SPML Provider

Operation	Description
search	Obtains every object that matches the specified query.
iterate	Obtains the next set of objects from the result set selected for a search operation.
closeIterator	Informs SPML Provider that the client no longer intends to iterate the search result.

Suspend capability

Table 8: Suspend capability

Operation	Description
suspend	Disables/deprovisions the specified object on the target.
resume	Re-enables the specified object on the target.
active	Checks whether the specified object on the target has been suspended.

Password Capability

Table 9: Password capability

Operation	Description
setPassword	Specifies a new password for a user account.
expirePassword	Marks as invalid the current password for a user account.

For detailed information on the SPML v2 operations, refer to the "Operations" section in the official SPML v2 specification, available for download at http://www.oasis-open.org/specs/index.php#spmlv2.0.

Samples of use

SPML Provider implements the SPML v2 core protocol and supports the DSML v2 Profile for SPML operations. SPML Provider comes with a sample client that includes examples illustrating how to construct SOAP messages that contain SPML payloads to perform common directory operations.

To work with the examples in the SPML Provider sample client

1. From the **Start** menu on the computer on which SPML Provider is installed, select **Active Roles SPML Provider** to open the home page of the sample client in your Web browser.
2. On the **Samples of Use** home page, under **How do I**, click the example you want to examine.

   For instance, you might click **Create new user** to view, modify, and perform the SPML v2 request that creates a user object.
3. On the page that opens, in the **SPMLv2 request** box, view the SOAP message that will be sent to SPML Provider.

   You may need to modify the SOAP message in order to adjust it to your environment. Thus, with the **Create new user** example, you have to set the ID attribute of the `<ContainerID>` element to the distinguished name (DN) of the container where you want to create a new user.

4. Click the **Send Request** button to send the SOAP message to SPML Provider.
5. In the **SPMLv2 response** box, view the SOAP message returned by SPML Provider in response to your request.
6. To examine another example, return to the home page, and then click the desired example.

Configuration settings in sample.config

Support for configuration options enables administrators to set the SPML Provider sample client configuration in order to test the SPML Provider functionality under actual conditions. Administrators can, for example, specify the desired settings for the sample container object (OU) that will be used in sample SPML v.2 operations.

The configuration settings of the SPML Provider sample client can be found in the `sample.config` file located in the **Samples** sub-folder of the SPML Provider installation folder.

The `sample.config` file contains data in the XML format. You can open and edit the configuration file with a common text editor such as Notepad. The default configuration settings in the `sample.config` file look as follows:

```
<samples>
<server>localhost</server>
<url>ARServerSPML/spmlprovider.asmx</url>
<sampleContainerName>OU=MyOU,DC=Company,DC=com</sampleContainerName>
</samples>
```

The following table provides reference information for XML elements used in the sample.config. file.

Table 10: XML elements used in the sample.config. file

Element	Parent element	Description
server	samples	Specifies the name of the computer running SPML Provider.
url	samples	Specifies Web address of SPML Provider. The default address is `ARServerSPML/spmlprovider.asmx`.
sampleContainerName	samples	Specifies the distinguished name of the container (OU) used in the sample SPML v.2 requests.

Core Operation samples

The following table lists all examples included in the Core Operation samples.

Table 11: Core operation samples

Operation	Description
List targets available for provisioning with SPML Provider	This example illustrates how to retrieve the targets available for provisioning with SPML Provider. To do this, SPML Provider performs the **listTargets** operation. The request message includes the following XML elements: • The `<soap:Envelope>` and `<soap:Body>` SOAP elements enclose the SPML payload. • The `<listTargetsRequest>` element asks SPML Provider to declare the set of targets that SPML Provider exposes for provisioning operations. The response lists the supported targets, including the schema definitions for each target and the set of capabilities that SPML Provider supports for each target. The contents of the `<listTargetsResponse>` element conform to the OASIS SPML v2 specification.
Create new user Create new user (using direct access mode)	These examples illustrate how to create a user account object in two operation modes. To create a new object, SPML Provider performs the **add** operation. The request message includes the following XML elements: • The `<soap:Envelope>` and `<soap:Body>` SOAP elements enclose the SPML payload. • The `<addRequest>` element asks SPML Provider to create a new object. • The `<containerID>` element specifies the distinguished name of the container in which to create the new object. • The `<data>` element encloses the elements that specify attribute values on the new object. Thus, in accordance with the `objectClass` attribute value, SPML Provider is requested to create a user account. The operation response indicates whether the user account is successfully created.

Operation	Description
	Note that in direct access mode, to provision a user account, you should complete the following steps: • Issue a request to create a new user account (see above). • Issue a request to set the user password (see "Set user password" in "Password capability samples," later in this document). • Issue a request to enable the user account (see "Resume user account" in "Suspend capability samples," later in this document).
Create new user (approval aware)	This example illustrates how to create a user account if this operation is subject to approval by designated approvers. For more information about approval activities and workflows, refer to Active Roles Help and Active Roles SDK. If the creation of user is subject to approval, to perform the operation, your SPML request *must* contain the `AllowApproval` built-in control. For information about how to use controls in SPML requests, see Support for Active Roles controls earlier in this document. To create a new object, SPML Provider performs the **add** operation. The request message includes the following XML elements: • The `<soap:Envelope>` and `<soap:Body>` SOAP elements enclose the SPML payload. • The `<addRequest>` element asks SPML Provider to create a new object. • The `<controls>` element includes the child element `<control>` that sets the `AllowApproval` control to the `Confirm` value. • The `<controlsForOutput>` element includes the child element `<control>`, which specifies that the `OperationStatus` control will be returned with the SPML response. • The `<containerID>` element specifies the distinguished name of the container in which to create the new object. • The `<data>` element encloses the elements that specify attribute values on the new object. Thus, in accordance with the `objectClass` attribute value,

Operation	Description
	SPML Provider is requested to create a user account. The operation response contains the `OperationStatus` control value that indicates the creation operation status. For example, if the user creation operation is subject to approval, the `OperationStatus` control returns the `Pending` value. In this case, the operation is waiting for approval by designated approvers. For more information about possible values of the `OperationStatus` control, see Active Roles SDK.
Create a user whose logon name is not in compliance with Active Roles policies	This example illustrates an attempt to create a new user account whose logon name does not conform to the Active Roles policies. Because the user logon name does not conform to the Active Roles policies, the creation operation fails and the operation response includes an error message returned by Active Roles. For example, an attempt to set the `sAMAccountName` attribute to a string of more than 20 characters causes the user creation operation to fail, with the response containing a message that provides some details on the error condition.
Create new group	This example illustrates how to create the group object **SPMLGroup** in the **mycompany.com** domain. To create a new object, SPML Provider performs the **add** operation. The request message includes the following XML elements: • The `<soap:Envelope>` and `<soap:Body>` SOAP elements enclose the SPML payload. • The `<addRequest>` element asks SPML Provider to create a new object. • The `<psoID>` element specifies the distinguished name of the object to be created. • The `<data>` element encloses the elements that specify attribute values on the new object. Thus, in accordance with the `objectClass` attribute value, SPML Provider is requested to create a group object.
Modify user attributes	This example illustrates how to modify the `description` attribute of the **John Smith** user object in the **mycompany.com** domain. To modify the object attribute, SPML Provider performs the **modify** operation.

Operation	Description
	The request message includes the following XML elements: • The `<soap:Envelope>` and `<soap:Body>` SOAP elements enclose the SPML payload. • The `<modifyRequest>` element asks SPML Provider to make changes to a specified object. • The `<psoID>` element specifies the distinguished name of the user account to be modified. • The `<modification>` element specifies the type of change as `replace`, causing the new values to replace the existing attribute values. • The `<data>` element encloses the elements that specify the new attribute values.
Modify Shared mailbox user permissions	Modify or replace the **edsaUserMailboxSecurityDescriptorSddl** attribute of the Shared mailbox object. To modify the object attribute, SPML Provider performs the **modify** operation. The request message includes the following XML elements: • The `<soap:Envelope>` and `<soap:Body>` SOAP elements enclose the SPML payload. • The `<modifyRequest>` element asks SPML Provider to make changes to a specified object. • The `<psoID>` element specifies the distinguished name of the user account to be modified. • The `<modification>` element specifies the type of change as `replace`, causing the new values to replace the existing attribute values. • The `<data>` element encloses the elements that specify the new attribute values, in SDDL format along with the SID of the user specified. For example, see Sample request to modify Shared mailbox user permissions.
Add user to group	This example illustrates how to add the **John Smith** user account to the **SPMLGroup** group object in the **mycompany.com** domain. To do this, SPML Provider preforms the **modify** operation. • The request message includes the following XML elements:

Operation	Description
	• The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <modifyRequest> element asks SPML Provider to make changes to a specified object. • The <psoID> element specifies the distinguished name of the group object to be modified. • The <modification> element specifies the type of change as add, causing the new values to be appended to the existing attribute values. • The <data> element encloses the elements that specify the distinguished name of the user account to be appended to the existing values of the member attribute.
Look up user attributes	This example illustrates how to get the XML representation of the **John Smith** user in the **mycompany.com** domain. To get the XML representation of an object, SPML Provider performs the **lookup** operation. The request message includes the following XML elements: • The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <lookupRequest> element asks SPML Provider to return the XML document that represents a specified object. • The <psoID> element specifies the distinguished name of the object. The response contains the object identifier, the XML representation of the object and its attributes, and information about SPML Provider capabilities that are supported on the object (the capability-specific data that is associated with the object).
Delete user	This example illustrates how to delete the **John Smith** user account. To do this, SPML Provider performs the **delete** operation. The request message includes the following XML elements: • The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <deleteRequest> element asks SPML Provider to delete a specified object.

Operation	Description
	• The <psoID> element specifies the distinguished name of the user account to delete.
Delete group	This example illustrates how to delete the **SPMLGroup** group object in the **mycompany.com** domain. To do this, SPML Provider performs the **delete** operation. The request message includes the following XML elements: • The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <deleteRequest> element asks SPML Provider to delete a specified object. • The <psoID> element specifies the distinguished name of the group object to delete.

Sample request to modify Shared mailbox user permissions

This section provides a sample request that illustrate how to use Active Roles controls in your SPML requests to modify Shared mailbox user permissions.

Sample request to modify Shared mailbox user permissions

```
<?xml version="1.0"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<spml:modifyRequest xmlns:spml="urn:oasis:names:tc:SPML:2:0">
<spml:psoID ID="CN=shmb1,OU=NOV_OU,DC=ars,DC=cork,DC=lab,DC=local"/>
<spml:modification>
<modification name="edsaUserMailboxSecurityDescriptorSddl" operation="replace"
xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>O:PSG:PSD:AI(A;CI;RC;;;S-1-5-21-2064067869-2662360268-1970296196-3772)
(A;CI;RC;;;S-1-5-21-2064067869-2662360268-1970296196-3773)
</value>
</modification>
</spml:modification>
</spml:modifyRequest>
</soap:Body>
</soap:Envelope>
```

Capability samples

The following tables list all examples included in the Capability samples, grouped by Capability.

Search Capability samples

Table 12: Search Capability samples

Operation	Description
Perform one-level search	This example illustrates how to obtain a list of the child objects (direct descendants) of the **Active Directory** container object. In proxy mode, you can use this example to list the domains that are registered with Active Roles (managed domains). To do this, SPML Provider performs the **search** operation. The request message includes the following XML elements: • The `<soap:Envelope>` and `<soap:Body>` SOAP elements enclose the SPML payload. • The `<searchRequest>` element asks SPML Provider to perform a search and return the identifiers of the objects found. • The `<query>` element determines that SPML Provider is to perform a one-level search (that is, to search only direct descendants of the object specified by `<basePsoID>`). • The `<basePsoID>` element specifies the distinguished name of the container object to search. The response contains the identifiers (distinguished names) of the objects residing in the container object specified by the `<basePsoID>` element.
Perform subtree search	This example illustrates how to obtain a list of objects that reside below the **Active Directory** object in the directory tree. You can use this example to list the objects that reside in a given domain. To do this, SPML Provider performs the **search** operation. The request message includes the following XML elements:

Operation	Description
	• The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <searchRequest> element asks SPML Provider to perform a search and return the identifiers of the objects found. • The <query> element determines that SPML Provider is to perform a subtree search (that is, to search any direct or indirect descendant of the object specified by <basePsoID>). • The <basePsoID> element specifies the distinguished name of the container object to search. For instance, this could be the distinguished name of a domain that is registered with Active Roles (managed domain). The response contains the identifiers (distinguished names) of the objects that reside in the directory tree below the container object specified by the <basePsoID> element.
Perform base search	This example illustrates how to obtain an XML representation of the specific object. To do this, SPML Provider performs the **search** operation. The request message includes the following XML elements: • The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <searchRequest> element asks SPML Provider to perform a search and return the XML representation of the object found. • The <query> element determines that SPML Provider is to perform a base search (that is, to search only the object identified by <basePsoID>). • The <basePsoID> element specifies the distinguished name of the object to search. For instance, this could be the distinguished name of a user account. The response contains the identifier of the object and the XML representation of the object (as defined in the schema of the target).
Iterate search results	This example illustrates how to obtain the next set of objects from the result set that SPML Provider selected for a search operation. In this case, SPML Provider performs the **iterate** operation. The request message includes the following XML elements:

Operation	Description
	• The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <iterateRequest> element asks SPML Provider to return additional objects that matched a previous search request but that the Provider has not yet returned to the client. • The <iterator> element supplies the iterator ID found either in the original search response or in a subsequent iterate response.
Stop iterating search results	This example illustrates how to tell SPML Provider that the client has no further need for the search results that a specific iterator represents. In this case, SPML Provider performs the **closeIterator** operation. The request message includes the following XML elements: • The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <closeIteratorRequest> element tells SPML Provider that the client no longer intends to iterate search results. • The <iterator> element specifies the ID of the iterator to close. This could be the iterator ID found in the original search response or in a subsequent iterate response.
Find inactive users	This example illustrates how to get a list of inactive (disabled or deprovisioned) user accounts found within a specified container. To do this, SPML Provider performs the **search** operation. The request message includes the following XML elements: • The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <searchRequest> element asks SPML Provider to perform a search and return the identifiers of the objects found. • The <query> element determines SPML Provider is to perform a subtree search. • The <basePsoID> element specifies the distinguished name of the container object to search. For instance, this could be the distinguished name of a certain

Operation	Description
	organizational unit. • The <filter> element encloses the elements that direct SPML Provider to search for inactive user accounts. Thus, the <equalityMatch> elements are configured so as to limit the search to user accounts; the <isActive> element combined with the <not> element causes SPML Provider to select the user accounts that are inactive. • The response contains the identifiers (distinguished names) of the inactive user accounts that exist in the directory tree below the container object specified by the <basePsoID> element.
Perform complex search	This example illustrates how to have SPML Provider find all objects that meet certain search criteria and return the values of certain attributes of the objects found. In this case, SPML Provider performs the **search** operation. The request message includes the following XML elements: • The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <searchRequest> element asks SPML Provider to perform a search and return the identifiers and attribute values of the objects found. • The <query> element determines the scope of the search. • The <basePsoID> element specifies the distinguished name of the container object to search. For instance, this could be the distinguished name of a certain organizational unit. • The <filter> element encloses the elements that specify the search criteria. • The <attributes> element specifies the object attributes to be included in the response. The response contains the identifiers (distinguished names) of the objects found and, for each object, the values of the attributes specified by the <attributes> element in the search request.
Find only security groups	This example illustrates how to obtain a list of security groups found in a specified container. In this case, SPML Provider performs the **search** operation.

Operation	Description
	The request message includes the following XML elements: • The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <searchRequest> element asks SPML Provider to perform a search and return the identifiers of the objects found. • The <query> element determines that SPML Provider is to perform a subtree search. • The <basePsoID> element specifies the distinguished name of the container object to search. For instance, this could be the distinguished name of a certain organizational unit. • The <filter> element encloses the elements that direct SPML Provider to search for security groups. Thus, the <equalityMatch> elements are configured so as to limit the search to group objects; the <extensibleMatch> element specifies a matching rule that is equivalent to the LDAP filter (groupType:1.2.840.113556.1.4.803:=2147483648) where 2147483648 is the decimal equivalent of the ADS_GROUP_TYPE_SECURITY_ENABLED flag (0x80000000). The response contains the identifiers (distinguished names) of the security groups that exist in the directory tree below the container object specified by the <basePsoID> element.

Password Capability samples

Table 13: Password capability samples

Operation	Description
Set user password	This example illustrates how to set a new password for the specific user account. To set a new password, SPML Provider performs the **setPassword** operation. The request message includes the following XML elements: • The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <setPasswordRequest> element asks SPML Provider to change to a specified value the password that is associated with a certain user account.

Operation	Description
	• The <psoID> element specifies the distinguished name of the user account. • The <password> element specifies the new password to assign to the user account.
Expire user password	This example illustrates how to force a given user to change the password at next logon. To do this, SPML Provider performs the **expirePassword** operation. The request message includes the following XML elements: • The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <expirePasswordRequest> element asks SPML Provider to mark expired the current password that is associated with a certain user account. The remainingLogins attribute is set to 1 so as to disallow grace logons once the expirePassword operation is completed, forcing the user to change the password at next logon. • The <psoID> element specifies the distinguished name of the user account.

Suspend Capability samples

Table 14: Suspend capability samples

Operation	Description
Suspend user account	This example illustrates how to either disable or deprovision a specified user account, depending on the SPML Provider configuration (see the description of the <suspendAction> element in the "Configuring SPML Provider" section earlier in this document). To do this, SPML Provider performs the **suspend** operation. The request message includes the following XML elements: • The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <suspendRequest> element asks SPML Provider to perform the suspend action on a certain user account (either disable or deprovision, depending on the configuration of SPML Provider).

Operation	Description
	• The <psoID> element specifies the distinguished name of the user account to suspend.
Resume user account	This example illustrates how to enable a disabled user account. This operation requires that the suspend action be set to disable in the SPML Provider configuration file (see the description of the <suspendAction> element in the "Configuring SPML Provider" section earlier in this document). In this case, SPML Provider performs the **resume** operation in order to enable a disabled user account. The request message includes the following XML elements: • The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <resumeRequest> element asks SPML Provider to re-enable a user account that has been disabled. • The <psoID> element specifies the distinguished name of the user account to re-enable.
Check whether user is active	This example illustrates how to determine whether a specified user account is active, that is, has not been suspended. A user account is considered to be suspended if the suspend action was performed on that account. The suspend action can be either disable or deprovision, depending on the SPML Provider configuration (see the description of the <suspendAction> element in the "Configuring SPML Provider" section earlier in this document). The request message includes the following XML elements: • The <soap:Envelope> and <soap:Body> SOAP elements enclose the SPML payload. • The <activeRequest> element asks SPML Provider to check whether the suspend action has been performed on a given user account (either disable or deprovision, depending on the SPML Provider configuration). • The <psoID> element specifies the distinguished name of the user account to check. The <activeResponse> element in the response message has the active attribute that indicates whether the specified user account is suspended. If the user account is suspended, the active attribute is set to false. Otherwise, the active attribute is set to true.

6

Active Roles SPML Provider terminology

Direct Access Mode

In this mode, SPML Provider directly connects to the specified domain or AD LDS instance.

Capabilities

A set of optional, functionally related operations defined in SPML v2.

Core Operations

The minimum set of operations that a provider must implement to conform to the official SPML v2 specification.

Extensible Markup Language (XML)

A meta-markup language that provides a format for describing structured data. This facilitates more precise declarations of content and more meaningful search results across multiple platforms. In addition, XML enables a new generation of Web-based data viewing and manipulation applications.

Organization for the Advancement of Structured Information Standards (OASIS)

An international consortium that drives the development, convergence, and adoption of e-business and Web service standards.

Provider

See Provisioning Service Provider.

Provisioning Service Object (PSO)

Represents a data entity or an information object on a target.

Provisioning Service Provider (PSP)

A software component that listens for, processes, and returns the results for well-formed SPML requests from a known requestor.

Provisioning Service Target (PST)

Represents a destination or endpoint that a provider makes available for provisioning actions.

Proxy Mode

In proxy mode, SPML Provider accesses directory data using the Active Roles proxy service.

Requesting Authority (RA)

A software component that issues well-formed SPML requests to a Provisioning Service Provider.

Requestor

See Requesting Authority.

Simple Object Access Protocol (SOAP)

An XML/HTTP-based protocol for platform-independent access to objects and services on the Web. SOAP defines a message format in XML that travels over the Internet using HyperText Transfer Protocol (HTTP). By using existing Web protocols (HTTP) and languages (XML), SOAP runs over the existing Internet infrastructure without being tied to any operating system, language, or object model.

SPML

An XML-based framework for exchanging user, resource, and service provisioning information between cooperating organizations.

SPML v2

An OASIS standard that provides a means of representing provisioning requests and responses as SPML documents.

Target

See Provisioning Service Target.

Target Schema

Defines the XML structure of the objects (PSO) that the target may contain.

Troubleshooting SPML Provider

This section briefly discusses some error statements that you may encounter when using SPML Provider.

Cannot remove the specified item because it was not found in the specified Collection

When sending a request to remove a user from a group (see the example below), the requested operation fails with the error statement "Cannot remove the specified item because it was not found in the specified Collection."

Resolution

This error has one of the following causes:

- The `<value>` element of the `<attr>` element specifies a user account that is not a member of the group.
- The Distinguished Name fields, such as CN or OU, used in the distinguished name of the user account to be removed, have invalid spelling or case. The Distinguished Name fields must be in upper case. So the use of `cn=Robert Smith` instead of `CN=Robert Smith` generates this error.

Verify that the `<value>` element specifies the distinguished name of the user that is the group member. Make sure that the Distinguished Name fields are in upper case.

The following example illustrates how to create a request to remove user **Robert Smith** from the **Sales** group.

```
<?xml version="1.0"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<modifyRequest xmlns="urn:oasis:names:tc:SPML:2:0" returnData="everything">
<psoID ID="CN=Sales,OU=SPML2,DC=Mycompany,DC=com"/>
<modification modificationMode="delete">
<data>
<attr name="member" xmlns="urn:oasis:names:tc:DSML:2:0:core">
<value>CN=Robert Smith,OU=Staff,DC=MyCompany,DC=com</value>
</attr>
</data>
</modification>
</modifyRequest>
</soap:Body>
</soap:Envelope>
```

Some of the specified attributes for the '<object class name>’ object class are not defined in the schema

When sending a request to change values of an object virtual attribute, the requested operation fails with the error statement “Some of the specified attributes for the '*<object class name>*' object class are not defined in the schema.”

Resolution

This error has one of the following causes:

- The `spmlschema.config` configuration file has changed since you started SPML Provider.
- The Default Application Pool idle timeout period has ended.

To resolve this issue, recycle the Default Application Pool or change its settings using Internet Information Services (IIS) Manager.

8

What’s new

This version of Active Roles SPML Provider has the same features and functions as the previous version, 1.4.0. The new version adds support for:

- Active Roles 7.3, allowing you to use the latest version of the Active Roles Administration Service.
- Adding users, groups, and contacts in Azure AD.

This version of Active Roles SPML Provider requires a 64-bit (x64) operating system, and cannot be installed on a 32-bit (x86) system (see System requirements earlier in this document).

About us

One Identity solutions eliminate the complexities and time-consuming processes often required to govern identities, manage privileged accounts and control access. Our solutions enhance business agility while addressing your IAM challenges with on-premises, cloud and hybrid environments.

Contacting us

For sales or other inquiries, visit https://www.oneidentity.com/company/contact-us.aspx or call +1-800-306-9329.

Technical support resources

Technical support is available to One Identity customers with a valid maintenance contract and customers who have trial versions. You can access the Support Portal at https://support.oneidentity.com/.

The Support Portal provides self-help tools you can use to solve problems quickly and independently, 24 hours a day, 365 days a year. The Support Portal enables you to:

- Submit and manage a Service Request
- View Knowledge Base articles
- Sign up for product notifications
- Download software and technical documentation
- View how-to-videos at www.YouTube.com/OneIdentity
- Engage in community discussions
- Chat with support engineers online
- View services to assist you with your product