Adding and Configuring a Service Provider

This topic provides instructions on how to add a new service provider. You must provide configuration details to add this service provider in the WSO2 Identity Server so that the authentication and/or provisioning happens as expected. For more information on how the service provider fits in to the WSO2 IS architecture, see Architecture.

The responsibility of the service provider configuration is to represent external service providers. The service provider configurations cover the following:

- **Define how the service provider talks to the Identity Server**
  This is via inbound authenticators. When you register a service provider, you need to associate one or more inbound authenticators with it.

- **Define how to authenticate users.**
  This can be via a local authenticator, request-path authenticator or federated authenticator. Based on this configuration, the Identity Server knows how to authenticate the user when it receives an authentication request (via an inbound authenticator) and based on the service provider who initiates it.

- **Maintain claim mapping.**
  This is to map the service provider's own set of claims to the Identity Server's claims, for example in your profile in the identity server has claim called work email (http://wso2.org/claims/emails.work) but your service provider application expect this to send this as email this will be done via claim mappings. When the authentication framework hands over a set of claims (which it gets from the local user store or from an external identity provider) to the response builder of the inbound authenticator, the framework talks to the service provider configuration component, find the claim mapping and do the claim conversion. See Configuring Inbound Authentication for a Service Provider for more information about response builder. Now the response builder will receive the claims in a manner understood by the corresponding service provider. Please read more about WSO2 IS claim management.

This topic contains the following sections.

- Adding a service provider
- Securing the Security Token Service
- Configuring a resident service provider
- Managing service providers

Adding a service provider

**Note:** This section only describes how to add a service provider using the Management Console. Instead of adding a service provider via the management console, it is also possible to add a service provider using a configuration file as described here.
1. Log in to the Management Console.
2. Navigate to the Main menu to access the Identity menu. Click Add under Service Providers.
3. Fill in the Service Provider Name and provide a brief Description of the service provider. Only Service Provider Name is a required field.

⚠️ The Service Provider Name should not contain any special characters except for fullstops (.), hyphens (-), underscores (_) and spaces.
The **20th January 2018 (2018-1-21) WUM update** provides a better user experience to manage application certificates. These certificates are used to validate the signatures in the messages (e.g., SAML requests) that come from the applications. In WSO2 IS these applications are known as service providers.

The **WUM update provides the following improvements:**

- Allows the Identity administrator to add the application certificate to the service provider creation User Interface (UI) of the management console. Previously, when configuring the service provider the Identity administrator needs to add the certificate to the keystore and then specify the certificate alias name.
- Stores the certificates in the database instead of the keystore file. Previously, after adding a new certificate, you need to start WSO2 IS since the keystore file was used to store the certificates.

**Follow the steps given below to enable the feature:**

a. WUM update your product to 20th January 2018 or beyond. For more information on updating WSO2 Identity Server, see [Updating WSO2 Products](#).

b. Add the database schema to store the certificates. Please refer the update summary PDF document that is in the `<WUM_UPDATE_PRODUCT_ARC HIVE>/updates/summary-<DATE>` directory, to get the details of the SQL queries. Once the schema change is done, the feature is available on your WSO2 IS 5.3.0 WUM updated pack.

c. Adding a service provider after enabling the feature:

   a. Click **Add** under **Service Providers**.
   b. Add a name and description for the service provider.
   c. Copy the content in your `.pem` certificate and paste it as the value for **Application Certificate**.
   d. Click **Register**.

**Example:**

![Service Providers](image)

---

**It is not mandatory** to WUM update your WSO2 IS 5.3.0 pack and enable this new feature. If you do WUM update the pack, **WSO2 IS 5.3.0 is backward compatible** and follows the previously implementation to locate the certificates in the keystore if the certificate is not added via the management console.

For example, if it is a SAML SSO flow, the certificate alias mentioned in SAML inbound authentication configuration is used when the certificate is not updated via the management console. If it is an OIDC request object signature validation, the thumb print in the JWT is used.
4. Click **Register** to add the new service provider.

**Note**: The service provider you create can be viewed by any user within your tenant domain in the **Main** view of the management console by clicking **List** under the **Service Provider** section. However, you must keep in mind that when a service provider is created, it is assigned to a "Internal" role. All users in the tenant domain must be assigned to this role in order to view the service provider created. See **Configuring Roles** for guidance on how to do this.

5. The **Service Providers** screen appears. Here you have the option of selecting if the service provider is a **SaaS Application** or not. The **SaaS Application** configuration defines which users you want to be able to log into your web application.

**Tip**: By default, the **SaaS Application** checkbox is disabled, which means the web application is not shared among tenants so only users in the current tenant (the one you use to define the service provider) will be allowed to log into the web application. Alternatively, if you enabled the **SaaS Application** checkbox, that means this web application is shared among tenants so users from any tenant will be allowed to log into the web application. For example, if there are three tenants, namely TA, TB and TC and the service provider is registered and configured only in TA.

- If the **SaaS Application** configuration is disabled, only users in TA are able to log into the web application.
- If the **SaaS Application** configuration is enabled, all TA, TB, TC users are able to log into the web application.
- For more information on creating and managing tenants, see **Creating and Managing Tenants**.
6. In the resulting screen, click the arrow buttons to expand the forms available to update.

Claim mapping for a service provider involves mapping claims that are used by the service provider to the claims local to the WSO2 Identity Server. In situations where a service provider needs some information of the user from the Identity Server where the service provider authenticates, the claim mapping is useful. Once the user is authenticated, the service provider can use these received claim details to provide its service. See the Identity Server Architecture for more information on how claim mapping fits into the overall scheme of things.

Claim mapping

- In the Claim Configuration form, select the claim mapping dialect by either choosing to use a local claim dialect or define your own custom claim dialect.
  - If you choose to Use Local Claim Dialect, you need to fill in the following details.
    1. Fill in your requested claims by clicking the Add Claim URI button.
    2. Choose your Local Claim from the drop-down. Select whether this claim is a Mandatory Claim for the Service Provider using the checkbox.

- If you choose to Define Custom Claim Dialect, you need to do the following.
  1. Add a custom claim URI by clicking on the Add Claim URI button. Clicking this button again enables you to map more claims.
  2. Add the Service Provider Claim and choose the corresponding Local Claim from the drop-down. Select whether you want the claim to be a Requested Claim and whether this claim is a Mandatory Claim for the service provider, using the relevant checkboxes.

**Note:** When mapping custom claims for a service provider configured with OpenID Connect, ensure to map the custom claims in the SP configuration as seen in the screenshot above AND also add the custom claims to a scope value in the oidc file.
To do this,

a. Click on **Browse** under **Registry** on the **Main** tab of the management console.
b. Navigate to `/system/config/` and click on the **oidc** file. Expand the **Properties** section.

c. Add the custom claims to the existing default scope **openid** by clicking on **Edit**. Alternatively, add a new scope for the custom claims by clicking on **Add New Property** and send it when getting the OIDC token along with the mandatory **openid** scope.

**Why?**

This is required because OpenIDConnect claim scopes are supported from WSO2 IS 5.2.0 onwards. This means that when you request for an OIDC token, you can specify a single scope value that is bound to a set of multiple claims. When that OIDC token is sent to the UserInfo endpoint, only the claims which are common in both the OIDC scope config file and the SP claim configuration (i.e., the intersection of claims in both these configurations) will be returned.
Note: When mapping custom claims for a service provider configured with SAML2, ensure to select both "Enable Attribute Profile" and "Include Attributes in the Response Always" from the SAML2 service provider configuration as follows.

- Enable Attribute Profile
  - Include Attributes in the Response Always

Why?

This is required since Identity Server include user claims in the SAML2 response only if SAML2 attribute profile is enabled.
Information on mapping claims

The Local Claim list includes a set of standard claim values which are local to the WSO2 Identity Server. When adding a service provider, it is necessary to map the values of the claims local to the service provider with those provided in this dropdown list which are local to the Identity Server. This should be done for all values in the service provider unless they use the same claim name.

Marking a claim as a Mandatory Claim would ensure that the WSO2 Identity Server will definitely send a value for this claim to the service provider. When a user logs into this service provider, if the identity provider does not provide a value for any of the mandatory claims, the user will be prompted to provide them at the time of login as shown in the image below.

Testing mandatory claims

To test out mandatory claims,

a. Configure the travelocity sample application by following the steps in the Configuring Single Sign-On topic.
b. Configure a few claims and select the checkbox for mandatory claims.
c. Ensure that there are one or more claims which are missing in the user profile of the user you wish to login with.
d. Run the travelocity sample and try the SAML login.
e. Log in with the user credentials of the user who has a few mandatory claims missing, and click Submit.
f. A claim request will be prompted, similar to the image below. At this point, the mandatory claim rule is enforced and you cannot proceed without providing the necessary claim values. Provide the necessary claim values and click Submit.
g. You will be successfully logged in to the application.

Marking a mapped claim as a Requested Claim would ensure that the service provider definitely sends this claim to the Identity Server. This is useful particularly in cases where there are hundreds of claims and only specific ones need to be sent to the Identity Server.

b. Select the Subject Claim URI and the Role Claim URI (for custom claims) from the drop-down. The claims you mapped are listed in the drop-down and you can choose among these claims.

- **Subject Claim URI** defines the authenticated user identifier which will return with the authentication response to the service provider.
- **Role Claim URI** defines the role claim for the service provider. This is useful if you use a different claim as the role claim or if you define a custom claim mapping for the service provider.
Related Topics

- See the [Logging in to Salesforce with Facebook](#) topic for a sample of claim mapping for a service provider.

You can configure the roles and permissions of the service provider.

To configure roles and permissions for a service provider, do the following.

a. Expand the **Role/Permission Configuration** section and the **Permission** and **Role Mapping** forms.

b. Click the **Add Permission** button. Clicking this button again will enable you to add another entry.

c. Add the **Service Provider Specific Permissions**. You can delete entries by clicking the **Delete** button.

   Once the service provider details have been added/updated, the permissions added here are listed out in the permissions available when **adding roles** in the Identity Server.

d. Click the **Add Role Mapping** button. Clicking this button again will enable you to add another entry.

e. Enter the **Local Role** and the **Service Provider Role**. You can delete entries by clicking the **Delete** button.

   Here you can map the exact role name available in the Identity Server with the role in the service provider.

The responsibility of the inbound authenticator component is to identify and parse all the incoming authentication requests and then build the corresponding response. A given inbound authenticator has two parts:

- **Request Processor**
- **Response Builder**

For each protocol supported by the WSO2 Identity Server, there should be an inbound authenticator. The Identity Server includes inbound authenticators for SAML 2.0, OpenID, OpenID Connect, OAuth 2.0, Kerberos KDC, WS-Trust STS and WS-Federation (passive). The responsibility of the SAML 2.0 request processor is to accept a SAML request from a service provider, validate the SAML request and then build a common object model understood by the authentication framework and handover the request to it. The responsibility of the SAML response builder is to accept a common object model from the authentication framework and build a SAML response out of it.

Both the request processors and the response builders are protocol aware, while the authentication framework is not coupled to any protocol. See **Architecture** for more information on the complete flow where inbound authenticators come into play.
You can configure the following for inbound authentication.

**Deprecated Feature!**

OpenID 2.0 is deprecated in this release (WSO2 Identity Server 5.3.0) as it is now an obsolete specification and has been superseded by OpenID Connect. Alternatively, we recommend that you use **OpenID Connect**. This is still available for use in WSO2 IS 5.3.0 but will be removed from the base product in WSO2 IS 5.4.0.

You can configure the following for inbound authentication.

a. Expand the **SAML2 Web SSO Configuration** and click **Configure**.

b. Select one of the following modes:
   - Manual configuration
   - Metadata file configuration
   - URL configuration

---

**Metadata and URL configuration**

When configuring a service provider (SP) or a federated identity provider (Federated IdP), the user is required to enter configuration data to facilitate exchanging authentication and authorization data between entities in a standard way. Apart from manual entering of configuration data, WSO2 IS allows you to upload configuration data using a metadata XML file or refer to a metadata XML file located in a predetermined URL. These two methods of uploading configuration data enable faster entry of configuration data because it allows the user to use the same metadata xml file for multiple instances of entity configuration. In addition to SAML metadata upload, WSO2 IS also supports SAML metadata download for the resident identity provider.

**Manual configuration**
a. Select **Manual Configuration** and fill in the form that appears.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Sample value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issuer</td>
<td>Specify the <strong>Issuer</strong>. This is the <code>&lt;saml:Issuer&gt;</code> element that contains the unique identifier of the service provider. This is also the issuer value specified in the SAML Authentication Request issued by the service provider. When configuring single-sign-on across Carbon servers, ensure that this value is equal to the <code>ServiceProviderID</code> value mentioned in the <code>&lt;IS_HOME&gt;/repository/conf/security/authenticators.xml</code> file of the relying party Carbon server.</td>
<td>travelocity.com</td>
</tr>
<tr>
<td>Assertion Consumer URLs</td>
<td>Specify the <strong>Assertion Consumer URLs</strong>. This is the URL to which the browser should be redirected to after the authentication is successful. This is the Assertion Consumer Service (ACS) URL of the service provider. The identity provider redirects the SAML2 response to this ACS URL. However, if the SAML2 request is signed and SAML2 request contains the ACS URL, the Identity Server will honor the ACS URL of the SAML2 request. It should have this format: <code>https://(host-name):(port)/acs</code>. You can add multiple assertion consumer URLs for the service provider by entering the URL and clicking the <strong>Add</strong> button.</td>
<td><a href="http://wso2is.local:8080/travelocity.com/home.jsp">http://wso2is.local:8080/travelocity.com/home.jsp</a></td>
</tr>
<tr>
<td>Default Assertion Consumer URL</td>
<td>Since there can be multiple assertion consumer URLs, you must define a <strong>Default Assertion Consumer URL</strong> in case you are unable to retrieve it from the authentication request.</td>
<td><a href="http://wso2is.local:8080/travelocity.com">http://wso2is.local:8080/travelocity.com</a></td>
</tr>
</tbody>
</table>

b. Click **Register**.

The following points should be taken into consideration when filling the above **New Service Provider** form.
**Tip:** In a service provider initiated single sign-on setup, the following needs to be considered.

- If no ACS URL is given in the `<AuthnRequest>`, the Identity Server sends the response to the default ACS URL of the service provider (whether the request is signed or not).
- If the ACS URL in `<AuthnRequest>` matches with one of the registered URLs, the Identity Server sends the response to the matched one.
- If the ACS URL in `<AuthnRequest>` does not match any of the registered ACS URLs and if the request is signed, the Identity Server sends the response to the ACS URL in the request only if the signature is valid. Alternatively, the `<AuthnRequest>` is rejected.

In an identity provider initiated single sign-on setup, the following needs to be considered.

- If the "acs" query parameter is not present in the request, the Identity Server sends the response to default ACS URL of the service provider.
- If the "acs" parameter is present and the value of that parameter matches with any of the registered ACS URLs of the service provider, then the Identity Server sends the response to the matched one.

### About NameID formats

For SSO interactions, you can use the following types of NameID formats.

- `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:
emailAddress`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`
This specifies the name identifier format that the Identity Server wants to receive in the subject of an assertion from a particular identity provider. The following is the default format used by the identity provider.

- `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`

<table>
<thead>
<tr>
<th>Certificate Alias</th>
<th>Select the <strong>Certificate Alias</strong> from the dropdown. This is used to validate the signature of SAML2 requests and is used to generate encryption. Basically the service provider’s certificate must be selected here. Note that this can also be the Identity Server tenant’s public certificate in a scenario where you are doing atenant specific configuration.</th>
<th>wso2carbon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response Signing Algorithm</td>
<td>Specifies the ‘SignatureMethod’ algorithm to be used in the ‘Signature’ element in POST binding. The default value can be configured in the <code>&lt;IS_HOME&gt;/repository/conf/identity.xml</code> file, in the SSOService element with <code>SAMLDefaultSigningAlgorithmURI</code> tag. If it is not provided the default algorithm is RSA-SHA 1, at URI <code>http://www.w3.org/2000/09/xmldsig#rsa-sha1</code>.</td>
<td><code>http://www.w3.org/2000/09/xmldsig#rsa-sha1</code></td>
</tr>
<tr>
<td>Response Digest Algorithm</td>
<td>Specifies the ‘DigestMethod’ algorithm to be used in the ‘Signature’ element in POST binding. The default value can be configured in the <code>&lt;IS_HOME&gt;/repository/conf/identity.xml</code> file, in the SSOService element with <code>SAMLDefaultDigestAlgorithmURI</code> tag. If it is not provided the default algorithm is SHA 1, at URI <code>http://www.w3.org/2000/09/xmldsig#sha1</code>.</td>
<td><code>http://www.w3.org/2000/09/xmldsig#sha1</code></td>
</tr>
<tr>
<td>Enable Response Signing</td>
<td>Select <strong>Enable Response Signing</strong> to sign the SAML2 Responses returned after the authentication process.</td>
<td>Selected</td>
</tr>
<tr>
<td>Enable Signature Validation in Authentication Requests and Logout Requests</td>
<td>Select <strong>Enable Signature Validation in Authentication Requests and Logout Requests</strong> if you need this functionality configured. This specifies whether the identity provider must validate the signature of the SAML2 authentication request and the SAML2 logout request that are sent by the service provider.</td>
<td>Unselected</td>
</tr>
<tr>
<td>Enable Assertion Encryption</td>
<td>Enable <strong>Assertion Encryption</strong>, if you wish to encrypt the assertion.</td>
<td>Unselected</td>
</tr>
<tr>
<td>Enable Single Logout</td>
<td>Select <strong>Enable Single Logout</strong> so that all sessions are terminated once the user signs out from one server. If single logout is enabled, the identity provider sends logout requests to all service providers. Basically, the identity provider acts according to the single logout profile. If the service provider</td>
<td>Selected</td>
</tr>
</tbody>
</table>
supports a different URL for logout, you can enter a SLO Response URL and SLO Request URL for logging out. These URLs indicate where the request and response should go to. If you do not specify this URL, the identity provider uses the Assertion Consumer Service (ACS) URL.

WSO2 Identity Server supports both SAML Back-Channel Logout and SAML Front-Channel Logout methods. By default, when you select Enable Single Logout, the Back-Channel Logout is enabled. In order to enable SAML Front-Channel Logout, you can either select Front-Channel Logout (HTTP Redirect Binding) or Front-Channel Logout (HTTP POST Binding).

**Enable Attribute Profile**
Select Enable Attribute Profile to enable this and add a claim by entering the claim link and clicking the Add Claim button. The Identity Server provides support for a basic attribute profile where the identity provider can include the user’s attributes in the SAML Assertions as part of the attribute statement. Once you select the checkbox to Include Attributes in the Response Always, the identity provider always includes the attribute values related to the selected claims in the SAML attribute statement.

**Enable Audience Restriction**
Select Enable Audience Restriction to restrict the audience. You may add audience members using the Audience text box and clicking the Add button.

**Enable Recipient Validation**
Select this if you require validation from the recipient of the response.

**Enable IdP Initiated SSO**
Select the Enable IdP Initiated SSO checkbox to enable this functionality. When this is enabled, the service provider is not required to send the SAML2 request.

**Enable IdP Initiated SLO**
Select the Enable IdP Initiated SLO checkbox to enable this functionality. You must specify the URL.

**Enable Assertion Query Request Profile**
Select the Enable Assertion Query Request Profile checkbox to query assertions that are persisted to the database when you log into the service provider application. For more information, see Querying SAML Assertions.

### Metadata file configuration
This option allows you to provide the configuration data required for configuring SAML2, by uploading a metadata .xml file instead of having to manually enter the values. This enables faster entry of configuration data and allows the user to use the same metadata XML file for multiple instances of entity configuration.

a. Select **Metadata File Configuration**.

   ![Metadata File Configuration](image)

b. Click **Choose File**, and select the .xml file containing the metadata for the service provider SAML configuration.

c. Click **Upload**.

```xml
Service provider metadata file

<entityDescriptor
    xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
    entityID="sample.com">
  <spSSODescriptor
      AuthnRequestsSigned="false"
      WantAssertionsSigned="false"
      protocolSupportEnumeration= "urn:oasis:names:tc:SAML:2.0:protocol" 
    />
  <KeyDescriptor use="signing">
    <KeyInfo
        xmlns="http://www.w3.org/2000/09/xmldsig#"
        >
      <X509Data
        >
        <X509Certificate
          MIIC+jCCAmOgAwIBAgIJAParOnPwEKkjMA0GCSqGSIb3DQEBBQUAMIGKMQswCQYD...
        </X509Certificate>
      </X509Data>
      </KeyInfo
    </KeyDescriptor
  </entityDescriptor>
</x509Certificate>
```
<X509Data>
</KeyInfo>
</KeyDescriptor>
<KeyDescriptor use="encryption">
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
<X509Data>
<X509Certificate>
MIIC+jjCCAmOgAwIBAgIjKjMA0GCSqGSIb3DQEBCwUAMIGKMQswCQYD
VQQGEwMJSzEQMA4GA1UECBMHBHVzgdGVybjEQMA4GA1UEBxMHQ29sb21ibzEWMQBQ
AUECHmNU29mdHdcmVm1ldzERMA8GA1UECxMIVHJhawW5pbmcxLDAqBgNVBAMT
INVzLnR3YJ1IFzpZXcgQ2VydGlmaWNhdGUyQXV0aG9yaXR5MB4XDTEwMjM4MDAw
MDc0OToxNzElMAsGA1UEFwoGZm9udjLEMakGA1UEEwJMY1NvZnR3YXJlIFZpZXc
c3R5cm4xEDAQBgNVBAcTB0NvbG9tYm8xEDAOBgNVBAoTDVNvZnR3YXJlIFZpZXc
ETAABgNVBAsTCTFRyYW1uaW5nMRgwFgYDVQDEw9NeSBUZXN0IFNlcnZpY2UwZ8w
DQYJKoZIhvcnNQEBBQADgY0AMIGJAoGBAAN6bi0l1IFz+R+93nLLK5BmnuF48tbODp
MBH7yGZ1/ESVUZoYm0GaPzg/ai3rX3r8BEr4TUrhpKUKBpFxZvb2q+yREIeDEKD
bHJuyVdS6hvtfa89WMJtwc7gwYyK8AoVJ94gU54GP26XyNpgdTXP0d3aH/Zt6
69xGAVo/01PaMBAAGJezB5MakGA1udEwQCMAAwHQYDVROOBBYEFNAwSmhuJSw
XGOSJnWdI1PKw9MBG8GA1uIdwQMBAAFNa3YmhDO7B0wbUqmYU1k/U6p/UUMCwG
CWCGSAGG+EIBDQ0FfHpcG5VU1NM1EdbmVyYXR1ZCBDCzJoAwpY2F0ZTABkgq
hkiG9wOBAQUAQUBgQBWcC5H+Uoaps4tDCicHqF2SXRTgF7PlAu2rLfm7jyoD
X+lFEoWDUoE5KtpMjsRlq/+2j9eTyi9xGj5sby4FvmXf8jS5L6zMKkezSb6qAv
tSHclLefKeidq6NDBJ8DhWHl/zvC9YbT0KkCToEgvCTBP2ZgsFtxTJcUKsogFA==
</X509Certificate>
</KeyInfo>
</EncryptionMethod Algorithm="https://www.w3.org/2001/04/xmldsig#">
</KeySize "xmlns="https://www.w3.org/2001/04/xmldsig#">128</KeySize>
</EncryptionMethod>
</KeyDescriptor>
<SingleLogoutService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="/sample.com:3443/federation/SPSloRedirect/metaAlias/sp"/>
<ResponseLocation="/sample.com:3443/federation/SPSloRedirect/metaAlias/sp"/>
<SingleLogoutService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="/sample.com:3443/federation/SPSloSoap/metaAlias/sp"/>
<ResponseLocation="/sample.com:3443/federation/SPSloSoap/metaAlias/sp"/>
<ManageNameIDService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="/sample.com:3443/federation/SPMiRedirect/metaAlias/sp"/>
<ResponseLocation="/sample.com:3443/federation/SPMiRedirect/metaAlias/sp"/>
<ManageNameIDService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="/sample.com:3443/federation/SPMiRedirect/metaAlias/sp"/>
<ResponseLocation="/sample.com:3443/federation/SPMiRedirect/metaAlias/sp"/>
Location="https://sample.com:3443/federation/SPMniSoap/metaAlias/sp"
ResponseLocation="https://sample.com:3443/federation/SPMniSoap/metaAlias/sp"/>
<NameIDFormat>
urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
</NameIDFormat>
<NameIDFormat>
urn:oasis:names:tc:SAML:2.0:nameid-format:transient
</NameIDFormat>
<AssertionConsumerService
isDefault="true"
index="0"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
Location="https://sample.com:3443/federation/Consumer/metaAlias/sp"/>
<AssertionConsumerService
index="1"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://sample.com:3443/federation/Consumer/metaAlias/sp"/>
</SPSSODescriptor>
</EntityDescriptor>

**URL configuration**

Metadata for a service provider may be published in a well known location via a URI. This option allows you to provide the configuration data required for configuring SAML2, by providing a URI (Ex: "https://spconfigs.com/sample-sp.xml") instead of having to manually enter the values. This enables faster entry of configuration data and allows the user to use the same metadata XML file for multiple instances of entity configuration.

a. **Select URL Configuration** and enter the URL containing the service provider metadata.

   ![Register New Service Provider](image)

   Select Mode
   - Manual Configuration
   - Metadata File Configuration
   - **URL Configuration**

b. Click **Upload**.

![Upload Service Provider Metadata From URL](image)

**Additional configurations**

- The following table provides the list of signature algorithms available and their respective URI.

<table>
<thead>
<tr>
<th>Signature algorithm name</th>
<th>Signature algorithm URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSA with SHA1</td>
<td><a href="http://www.w3.org/2000/09/xmldsig#dsasha1">http://www.w3.org/2000/09/xmldsig#dsasha1</a></td>
</tr>
<tr>
<td>ECDSA with SHA1</td>
<td><a href="http://www.w3.org/2001/04">http://www.w3.org/2001/04</a></td>
</tr>
<tr>
<td>Digest algorithm name</td>
<td>Digest algorithm URI</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>MD5</td>
<td><a href="http://www.w3.org/2001/04/xmlsigmore#md5">http://www.w3.org/2001/04/xmlsigmore#md5</a></td>
</tr>
<tr>
<td>RIPEMD160</td>
<td><a href="http://www.w3.org/2001/04/xmlenc#ripemd160">http://www.w3.org/2001/04/xmlenc#ripemd160</a></td>
</tr>
<tr>
<td>SHA1</td>
<td><a href="http://www.w3.org/2000/09/xmlsig#sha1">http://www.w3.org/2000/09/xmlsig#sha1</a></td>
</tr>
<tr>
<td>SHA256</td>
<td><a href="http://www.w3.org/2001/04/xmlenc#sha256">http://www.w3.org/2001/04/xmlenc#sha256</a></td>
</tr>
<tr>
<td>SHA384</td>
<td><a href="http://www.w3.org/2001/04/xmlsigmore#sha384">http://www.w3.org/2001/04/xmlsigmore#sha384</a></td>
</tr>
<tr>
<td>SHA512</td>
<td><a href="http://www.w3.org/2001/04/xmlenc#sha512">http://www.w3.org/2001/04/xmlenc#sha512</a></td>
</tr>
</tbody>
</table>

If you need to sign the SAML response using an authenticated user's tenant keystore, please add the following configuration. (By default, the response is signed using the certificate that belongs to the tenant where the service provider is registered). This property must be added if the SAML authenticator version in the WSO2 Carbon products that you are using is 4.2.2 or higher (org.wso2.carbon.identity.authenticator.saml2.sso_4.2.2.jar).

Add the `<UseAuthenticatedUserDomainCrypto>` property available in the `<IS_HOME>/repository/conf/identity.xml` file as shown below.

```xml
<SSOService>
  ...
  <UseAuthenticatedUserDomainCrypto>true</UseAuthenticatedUserDomainCrypto>
</SSOService>
```
Related Topics

See SAML 2.0 Web SSO for more information on SAML2 single-sign-on and see the following topics for samples of configuring single-sign-on using SAML2.

- Configuring Single Sign-On
- Logging in to WSO2 Products via Identity Server
- Configuring SAML2 Single-Sign-On Across Different WSO2 Products

See Using the SAML2 Toolkit for support on debugging issues with SAML2 configurations.

OAuth 2.0 has three main phases. They are; requesting an Authorization Grant, exchanging the Authorization Grant for an Access Token and accessing the resources using this Access Token. OpenID Connect is another identity layer on top of OAuth 2.0. OAuth applications can get authentication event information over the IDToken and can get the extra claims of the authenticated user from the OpenID Connect UserInfo endpoint.

To enable OAuth support for your client application, you must first register your application. Follow the instructions below to add a new application.

a. Expand the Inbound Authentication Configuration section and then expand OAuth/OpenID Connect Configuration. Click Configure.

b. Fill in the form that appears. For the Allowed Grant Types you can disable the ones you do not require or wish to block.

⚠️ Note: The grant type highlighted below is a custom grant type. This will only appear on the UI if you have configured the JWT grant type. The value specified in the <Grant TypeName> property of the identity.xml file when creating the custom grant type is the value that will appear on the UI. For more information on writing a custom grant type, see Writing a Custom OAuth 2.0 Grant Type.
When filling out the **New Application** form, the following details should be taken into consideration.

<table>
<thead>
<tr>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OAuth Version</strong></td>
<td>Selecting <strong>OAuth Version</strong> as <strong>1.0a</strong> removes all the configurable <strong>Allowed Grant Types</strong>. This is because this version of OAuth does not support grant types.</td>
</tr>
<tr>
<td><strong>Callback Url</strong></td>
<td>This is the exact location in the service provider’s application where an access token would be sent. This is a required field (if the grant type is anything other than ‘Code’ or ‘Implicit’) and it is important to configure, as it is imperative that the service provider receives the access token. This is necessary for security purposes to ensure that the token is not compromised.</td>
</tr>
</tbody>
</table>

---

### Configure multiple callback URLs

From IS 5.2.0 onwards, regex based consumer URLs are supported when defining the callback URL. This enables you to configure multiple callback URLs for one application by entering a regex pattern as the value for the callback URL field.

For example, if you have two service providers that use the same application, you can now define a regex pattern which will work for both callback URLs instead of having to configure two different applications for the two service providers. Assume the two callback URLs for your two service providers are as follows:

- `https://myapp.com/callback`
- `https://testapp:8080/callback`

To configure the callback URL to work for **both** of these URLs, set it using a regex pattern as follows:
6. **b.** Allowed Grant Types - The following are the grant types that are used to get the access token:

<table>
<thead>
<tr>
<th>Grant Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Code</strong></td>
<td>Entering the user name and password required at the service provider will result in a code being generated. This code can be used to obtain the access token. For more information on this grant type, see this <a href="#">Authorization Code specification</a>.</td>
</tr>
<tr>
<td><strong>Implicit</strong></td>
<td>This is similar to the code grant type, but instead of generating a code, this directly provides the access token. For more information on this grant type, see this <a href="#">Implicit Grant specification</a>.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>This authenticates the user using the password provided and the access token is provided. For more information on this grant type, see this <a href="#">Resource Owner Password Credentials Grant specification</a>.</td>
</tr>
<tr>
<td><strong>Client Credential</strong></td>
<td>This is the grant type for the client key and client secret. If these two items are provided correctly by the service provider, the access token is sent. For more information on this grant type, see this <a href="#">Client Credentials specification</a>.</td>
</tr>
<tr>
<td><strong>Refresh Token</strong></td>
<td>This will enable the user to obtain an access token by using the refresh token once the originally provided access token is used up. For more information on this grant type, see this <a href="#">Refresh Token specification</a>.</td>
</tr>
<tr>
<td><strong>SAML</strong></td>
<td>This uses SAML assertion to obtain the access token. For more information on this grant type, see this <a href="#">SAML2 Bearer specification</a>.</td>
</tr>
<tr>
<td><strong>IWA-NTLM</strong></td>
<td>This is similar to the password grant type, but it is specific to Microsoft Windows users.</td>
</tr>
<tr>
<td><strong>urn:ietf:params:oauth:grant-type:jwt-bearer</strong></td>
<td>This is a custom grant type. It uses a JWT token to obtain the access token. For more information about this grant type, see this <a href="#">JWT specification</a>.</td>
</tr>
</tbody>
</table>

You can also configure a regex pattern that contains dynamic values as seen below:

```text
regexp=https://mchcon.clance.local\?id=(.*)
```

**To send a request with multiple callback URLs, click** here.

**You must have the prefix `regexp=` before your regex pattern. To define a normal URL, you can specify the callback URL without this prefix.**


You can also configure a regex pattern that contains dynamic values as seen below:

```text
regexp=https://mchcon.clance.local\?id=(.*)
```
c. Click Add. The following information is added for your service provider.

<table>
<thead>
<tr>
<th>Mandatory</th>
<th>security measure used to mitigate a code interception attack. See Mitigating Authorization Code Interception Attacks for more information.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support PKCE 'Plain' Transform Algorithm</td>
<td>Select this if you are using PKCE.</td>
</tr>
</tbody>
</table>

- **OAuth Client Key** - This is the client key of the service provider, which will be checked for authentication by the Identity Server before providing the access token.
- **OAuth Client Secret** - This is the client secret of the service provider, which will be checked for authentication by the Identity Server before providing the access token.

Click the **Show** button to view the exact value of this.

- **Actions** -
  - **Edit**: Click to edit the OAuth/OpenID Connect Configurations
  - **Revoke**: Click to revoke (deactivate) the OAuth application. This action revokes all tokens issued for this application. In order to activate the application, you have to regenerate the consumer secret.
  - **Regenerate Secret**: Click to regenerate the secret key of the OAuth application.
  - **Delete**: Click to delete the OAuth/OpenID Connect Configurations.

Tip: The OAuth client key and client secret are stored in plain text. To encrypt the client secret, access token and refresh token, do the following:

Open the `identity.xml` file found in the `<IS_HOME>/repository/conf/identity` directory and change the `<TokenPersistenceProcessor>` property as follows:

```
<TokenPersistenceProcessor>org.wso2.carbon.identity.oauth.tokenprocessor.EncryptionDecryptionPersistenceProcessor</TokenPersistenceProcessor>
```

After updating the configuration, make sure to restart the server for the changes to be applied on WSO2 IS.

WSO2 Identity Server enables defining OAuth token validity periods for all OAuth service providers as well as for a particular service provider.

- To learn how to configure the token validity periods for all service providers, see OAuth Token Validity Period.
- To learn how to configure the token validity periods of each service provider, see Configuring OAuth Token Validity Period for a Service Provider.
WSO2 Identity Server’s passive security token service (Passive STS) is used as the **WS-Federation** implementation. The Passive STS is capable of issuing SAML 1.1 and 2.0 security tokens.

To request a SAML 2.0 security token, the Request Security Token (RST) should be sent to the passive STS endpoint with the TokenType ‘SAMLV2.0’ when sending the token request. If there is no RST specified, the WSO2 Identity Server will issue a SAML 1.1 token by default.

**Configuring passive STS**

a. See [here](#) for details on adding a service provider.
b. Expand the **Inbound Authentication Configuration** followed by the **WS-Federation (Passive) Configuration** section and provide the following values.
   - **Passive STS Realm** - This should be an unique identifier for the web app. Provide the same realm name given to the web app you are configuring WS-Federation for.
   - **Passive STS WReply URL** - Provide the URL of the web app you are configuring WS-Federation for. This endpoint URL will handle the token response.

   **Tip**

   If you want to configure an expiration time for the security token, you need to add the following configuration in the `<IS_HOME>/repository/conf/carbon.xml` file, under the `<Server>` element:

   ```xml
   <STSTimeToLive>1800000</STSTimeToLive>
   ```

   Here, the expiration time should be specified in milliseconds.

c. Expand the **Claim Configuration** section and map the relevant claims. See [Configuring Claims for a Service Provider](#) for more information.
d. Click **Update** to save changes.
Currently the signing algorithm used for passive STS by default is rsa-sha1 and the digest algorithm used is sha1. To change the default algorithms, add the following configuration under the <security> tag in the carbon.xml file found in the <IS_HOME>/repository/conf directory. The example given below sets the signing algorithm to rsa-sha256 and the digest algorithm to sha256.

```xml
<STSSignatureAlgorithm>http://www.w3.org/2001/04/xmldsig-more#rsa-sha256</STSSignatureAlgorithm>
<STSDigestAlgorithm>http://www.w3.org/2001/04/xmlenc#sha256</STSDigestAlgorithm>
```

To configure this, apply the 3758 WUM update to WSO2 IS 5.6.0 using the WSO2 Update Manager (WUM). To deploy a WUM update into production, you need to have a paid subscription. If you do not have a paid subscription, you can use this feature with the next version of WSO2 Identity Server when it is released. For more information on updating WSO2 Identity Server using WUM, see Getting Started with WUM in the WSO2 Administration Guide.

### Related Topics

- To test out WSO2 Identity Server's passive security token service using a sample, see Testing Identity Server's Passive STS.

WSO2 Identity Server's security token service (STS) is used as the WS-Trust implementation. The STS is capable of issuing SAML 1.1 and 2.0 security tokens and has a SOAP/XML API for token issuance. This API can be secured with the UserNameToken or with any other WS-Security mechanism as explained below.

### Securing the Security Token Service

According to the Trust Brokering model defined in the WS-Trust specification, the users should authenticate themselves to the STS before obtaining a token. STS may use this authentication information when constructing the security token. For example, STS may populate the required claims based on the user name provided by the subject. Therefore, the STS service needs to be secured.

STS is configured under the Resident Identity Provider section of the Identity Server Management Console. Follow the instructions below to secure the Security Token Service by logging into the management console.

1. Log in as an admin to access the management console.
2. Configure the Resident Identity Provider. See here for more detailed information on how to do this.
3. In the Resident Identity Provider page, expand the Inbound Authentication Configuration section along with the Security Token Service Configuration section.
d. Click **Apply Security Policy**.

![Resident identity Provider configuration](image)

**Select Domain** drop-down lists many domains. The listed **User Groups** can vary depending on the domain selected.

e. Select **Yes** in the **Enable Security?** dropdown and select a pre-configured security scenario according to your requirements. In this case, we will use **UsernameToken** under the **Basic Scenarios** section.

![Security for the service](image)

You can find further details about the security policy scenarios from the **view scenario** option.

f. Click **Next**.

![Fitness may vary as per the security scenario that you have chosen under point (5) above. Below, is for **UsernameToken** scenario.](image)

Next steps may vary as per the security scenario that you have chosen under point (5) above. Below, is for **UsernameToken** scenario.

g. Select **ALL-USER-STORE-DOMAINS** from the drop-down.

h. In the resulting page, select the role you created, to grant permission to access secured service. In this example, admin role is used. Next, click **Finish**.
6. i. Click **Ok** on the confirmation dialog window that appears.
   j. Click **Update** to complete the process.

Now, STS is configured and secured with a username and password. Only the users with the Admin role can consume the service.

The next step is to add a service provider to consume the STS.

**Adding a service provider for the STS client**

Do the following steps if you are using a Holder of Key **subject confirmation method**. See Configuring STS for Obtaining Tokens with Holder-Of-Key Subject Confirmation for more information.

---

**i.** The **Subject confirmation methods** define how a relying party (RP), which is the end service, can make sure a particular security token issued by an STS is brought by the legitimate subject. If this is not done, a third party can take the token from the wire and send any request it wants including that token. The RP trusts that illegitimate party.

---

a. See Configuring a Service Provider for details on adding a service provider.
b. Expand the **Inbound Authentication Configuration** section and the **WS-Trust Security Token Service Configuration** section. Click **Configure**.

In the resulting screen, enter the trusted relying party's **endpoint address** that is the endpoint address of the Security Token Service. For more information, see **Broker Trust Relationship with WSO2 Identity Server** and upload the **public certificate** of the trusted relying party.

You need to add the certificate of the relying party to the truststore. For more information on how to create the certificate and add it to the truststore, see [here](#).

The endpoint must be used as the service **URL** to which the token gets delivered by the STS client. Then select the public certificate imported. Tokens issued are encrypted using the public certificate of the trusted relying party. Therefore, the consumer who obtains this token, to invoke the RP service won’t be able to see the token.

### STS Configuration

![STS Configuration](image)

```
Add new trusted service

Endpoint Address*  https://localhost:9444/services/echo
Certificate Alias   wso2carbon

Apply
```

c. In the resulting screen, enter the trusted relying party's endpoint address that is the endpoint address of the Security Token Service. For more information, see **Broker Trust Relationship with WSO2 Identity Server** and upload the **public certificate** of the trusted relying party.

d. Click **Update** to save the changes made to the service provider.
The responsibility of the local authenticators is to authenticate the user with locally available credentials. This can be either username/password or even IWA (Integrated Windows Authentication) or FIDO (Fast IDentity Online). Local authenticators are decoupled from the Inbound Authenticators. Once the initial request is handed over to the authentication framework from an inbound authenticator, the authentication framework talks to the service provider configuration component to find the set of local authenticators registered with the service provider corresponding to the current authentication request.

Once the local authentication is successfully completed, the local authenticator will notify the framework. The framework will now decide no more authentication is needed and hand over the control to the corresponding response builder of the inbound authenticator. See Architecture for more information on this.

You can configure the following for local and outbound authentication.

a. Expand Local & Outbound Authentication Configuration.

- **Assert identity using mapped local subject identifier**: This option will use the local subject identifier when asserting the identity.
- **Always send back the authenticated list of identity providers**: This option will send back the list of identity providers that the current user is authenticated by.
- **Use tenant domain in local subject identifier**: This option will append the tenant domain to the local subject identifier.
- **Use user store domain in local subject identifier**: This option will append the user store domain that the user resides in the local subject identifier.
- **Enable Authorization**: This option enables you to engage authorization policies for the service provider. For more information, see [Configuring Access Control Policy for a Service Provider](#).

b. Select the **Authentication Type** you require from the available options. This is a required field.
- If you choose **Local Authentication**, you need to select the local authentication option from the dropdown list.
- If you choose **Federated Authentication**, you need to select the identity provider from the dropdown list.
- If you choose **Advanced Configurations**, you can configure additional authentication steps and additional authentication options.

i. There are two types of multi-factor authentication that can be configured here.
   1. **Multi-step authentication**: Click **Add Authentication Step**. Clicking this again will enable you to create another authentication step.
   2. **Multi-option authentication**: Click **Add Authenticator** to add either a Local or Federated authenticator after selecting it from the dropdown. Adding more than one of these within a single step enables multi-option authentication.

ii. Select whether to **Use subject identifier from this step, Use attributes from this step** or both. In the case of multiple steps, you can have only one step to use subject identifier from this step and one to use attributes from this step.
   For example, let's say, We configure 1st step as Facebook and enable **Use subject identifier from this step**. Then configure Google for 2nd step and enable **Use attributes from this step**. Once authentication is complete subject id will be taken from the Facebook claims and Google claims will be used as users attributes.

iii. Click the plus button to add a **Local Authenticator**. You can choose the type of authenticator using the dropdown. Clicking the plus button again will enable you to add a second local authenticator. Basic authentication allows you to authenticate users from the enterprise user store.

iv. Click the plus button to add a **Federated Authenticator**. You can choose the type of authenticator using the dropdown. Clicking the plus button again will enable you to add a second federated authenticator.

v. Click the **Update** button. This will return you to the previous screen with your newly configured authentication steps.

c. Add a local authenticator under **Request Path Authentication Configuration** by clicking the **Add** button. Clicking the **Add** button again enables you to add another local authenticator. The two types of local authenticators available are as follows.
- OAuthRequestPathAuthenticator
- BasicAuthRequestPathAuthenticator

Look through the following for more details on the various authentication types.

<table>
<thead>
<tr>
<th>Authentication Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td></td>
</tr>
</tbody>
</table>
This is the default authenticator sequence for a configured service provider in the Identity Server. This sequence can be modified by updating following section in the `<IS_HOME>/repository/conf/identity/application-authentication.xml` file.

```
<Sequences>
  <!-- Default Sequence. This is mandatory -->
  <Sequence appId="default">
    <Step order="1">
      <Authenticator name="BasicAuthenticator"/>
      <Authenticator name=""></Step>
  </Sequence>
</Sequences>
```

<table>
<thead>
<tr>
<th>LocalAuthentication</th>
<th>In this case, Identity Server itself authenticate the user. There are three types of local authenticators OOTB in a fresh Identity Server pack.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- The <strong>basic</strong> authenticator is used to authenticate the user using the credentials available in the Identity Server.</td>
</tr>
<tr>
<td></td>
<td>- <strong>IWA</strong> stands for Integrated Windows Authentication and involves automatically authenticating users using their Windows credentials.</td>
</tr>
<tr>
<td></td>
<td>- <strong>FIDO</strong> authenticator is a local authenticator that comes with the WSO2 Identity Server. This will handle FIDO authentication requests related key validation against stored keys, the public key, keyhandler, and the counter, attestation certificate of FIDO registered users.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FederatedAuthentication</th>
<th>In this case, Identity Server trust third-party Identity provider to perform the user authentication. These Identity providers use various protocols to transfer authentication/authorization related messages. Currently, the Identity Server only supports the following federated authenticators OOTB.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- SAML2 Web SSO</td>
</tr>
<tr>
<td></td>
<td>- OAuth2/OpenID Connect</td>
</tr>
<tr>
<td></td>
<td>- WS-Federation (Passive)</td>
</tr>
<tr>
<td></td>
<td>- Facebook</td>
</tr>
<tr>
<td></td>
<td>- Microsoft (Hotmail, MSN, Live)</td>
</tr>
<tr>
<td></td>
<td>- Google</td>
</tr>
<tr>
<td></td>
<td>- SMS OTP</td>
</tr>
<tr>
<td></td>
<td>- Email OTP</td>
</tr>
<tr>
<td></td>
<td>- Twitter</td>
</tr>
<tr>
<td></td>
<td>- Yahoo</td>
</tr>
<tr>
<td></td>
<td>- IWA Kerberos</td>
</tr>
<tr>
<td></td>
<td>- Office365</td>
</tr>
</tbody>
</table>

| AdvancedConfiguration | Advanced configurations enable you to add multiple options or steps in authentication. When multiple authentication steps exist, the user is authenticated based on each and every one of these steps. If only one step is added then the user is only authenticated based on the local and/or federated authenticators added in a single step. However, in the case of local and/or federated authenticators, the authentication happens based on any one of the available authenticators. |

| Request path authenticators | A request path authenticator will get executed only if the initial authentication request brings the applicable set of credentials with it. See Request Path Authentication for more details. |

---

Copyright © WSO2 Inc. 2005-2017
Related Topics

- See Multi-factor Authentication using FIDO for more information on configuring multi-step and multi-option authentication using FIDO.
- See Request Path Authentication for information on a local authenticator that is executed if the initial authentication request brings a set of credentials with it.
- See Try Request Path Authentication for more information on how the request path authenticator works using the WSO2 playground sample.

See Inbound Provisioning for more information on inbound provisioning. To configure inbound provisioning for the service provider, follow the steps below.

Do the following to configure inbound provisioning.

a. Expand the Inbound Provisioning Configuration section and expand the SCIM/SOAP Configuration form.

b. Select the user store domain name from the drop down list to provision users and groups.

Related Topics

- See Configuring User Stores for more information.
- See Inbound Provisioning for configuring user stores and service providers for inbound provisioning.

You might want to outbound provision users that self-register to your applications (service provider) to other entities, such as Google or Salesforce. This section guides you on how to configure outbound provisioning for a service provider.

Scenario

For example, your organization has a registration portal to add new employees and a registration portal to add customers to your system. WSO2 Identity Server acts as your Identity Provider. You want the newly added users to be provisioned to Google and Salesforce because Gmail and Salesforce are used for day to day activities in the organization. The customer details only need to be stored in the WSO2 Identity Server's user store. Therefore, you configure the Service Provider that was created for the employees' registration application to handle outbound provisioning too.

Do the following to configure outbound provisioning.

a. Expand the Outbound Provisioning Configuration. For this section to be configurable, you should already have added an identity provider and enabled an outbound provisioning connector (Google, Salesforce, SCIM, SPML, etc.).

b. Choose the identity provider you require from the drop-down and click .
c. Click **Update** to save your changes.

![Outbound Provisioning Configuration](image)

- If you select **Blocking**, the authentication flow is blocked until the provisioning finishes. If **Blocking** is not selected, provisioning happens in a different thread. In the blocking mode, the authentication flow will be blocked till the provisioning finishes while in the non-blocking mode, provisioning happens in a different thread. If you want to allow a user to access your application only if the user is authenticated as well as provisioned then you have to use blocking mode, in other words, if you don't care whether the user gets provisioned or not in order to access your application you can use non-blocking mode.
- If you select **Enable JIT Outbound**, the users are provisioned to the WSO2 IS user store on-the-fly as and when they are authenticated using just-in-time provisioning. To enable this, you need to configure JIT provisioning for the Identity Provider. For more information, see Configuring Just-In-Time Provisioning.
- If you select **Enable Rules**, the users are provisioned based on the pre-defined XACML rules. For more information, see Rule Based Provisioning.
- The **Delete** button to remove the identity provider you added.

**Related Topics**

- See [Outbound Provisioning](#) for more information on configuring user stores and service providers for outbound provisioning.

7. Click the **Update** button to update the details of the service provider.

### Configuring a resident service provider

WSO2 Identity Server can mediate authentication requests between service providers and identity providers. At the same time, the Identity Server itself can act as a service provider and an identity provider. When it acts as a service provider it is known as the resident service provider.

The Identity Server mainly acts as a resident service provider while adding users to the system. You can enable provisioning configurations for the resident service provider. For example, if you try to add users to the system via the SCIM API (You must use a privileged local account to invoke the API to authenticate with HTTP Basic Authentication.), the system will read the provisioning configurations from the resident service provider.

At the same time, if you want to configure outbound provisioning for any user management operation done via the management console, SOAP API or the SCIM API, you must configure outbound provisioning identity providers against the resident service provider. So, based on the outbound configuration, users added from the management console will also be provisioned to external systems like Salesforce and Google Apps.

Follow the instructions below to configure a resident service provider in the WSO2 Identity Server.

1. Sign in. Enter your username and password to log on to the **Management Console**.
2. Click **Resident** under the **Service Providers** on the **Main** tab.
3. The **Resident Service Provider** page appears.

   ![Resident Service Provider](image)

   a. Select the user store domain to provision users and groups for inbound authentication for SCIM or SOAP requests.
   b. For outbound provisioning configurations, select the identity provider from the dropdown list available and click the plus button to add this identity provider for provisioning. For an identity provider to appear on this list you have to add the identity provider in the Identity Server. The following are the names that would appear for each type of provisioning connector:
      - Google provisioning connector - Google and googleapps
      - Salesforce provisioning connector - salesforce.com and salesforce
      - SCIM provisioning configuration - scim
      - SPML provisioning configuration - spml

4. Click **Update**.

**Managing service providers**

This topic provides instructions on how to manage service providers once they are created.

**Viewing service providers**

Follow the instructions below to view the list of service providers added in the WSO2 Identity Server.

1. Sign in. Enter your username and password to log on to the Management Console.
2. In the **Main** menu under the **Identity** section, click **List** under **Service Providers**. The list of service providers you added appears.

<table>
<thead>
<tr>
<th>Service Provider ID</th>
<th>Description</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workday</td>
<td>Workday SSO</td>
<td>Edit</td>
</tr>
<tr>
<td>Playground</td>
<td>Request path authentication with WSO2 Playground</td>
<td>Edit</td>
</tr>
<tr>
<td>Salesforce</td>
<td></td>
<td>Edit</td>
</tr>
<tr>
<td>Liferay</td>
<td>Liferay Portal</td>
<td>Edit</td>
</tr>
</tbody>
</table>

**Editing service providers**

Follow the instructions below to edit a service provider's details.

1. Sign in. Enter your username and password to log on to the Management Console.
2. In the **Main** menu under the **Identity** section, click **List** under **Service Providers**. The list of service providers you added appears.
3. Locate the service provider you want to edit and click on the corresponding **Edit** link.

<table>
<thead>
<tr>
<th>Service Provider ID</th>
<th>Description</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workday</td>
<td>Workday SSO</td>
<td>Edit</td>
</tr>
<tr>
<td>Playground</td>
<td>Request path authentication with WSO2 Playground</td>
<td>Edit</td>
</tr>
<tr>
<td>Salesforce</td>
<td></td>
<td>Edit</td>
</tr>
<tr>
<td>Liferay</td>
<td>Liferay Portal</td>
<td>Edit</td>
</tr>
</tbody>
</table>

4. You are directed to the edit screen. See [here](#) for details on the editable form.

**Deleting service providers**

Follow the instructions below to delete a service provider.
1. Sign in. Enter your username and password to log on to the Management Console.
2. In the **Main** menu under the **Identity** section, click **List** under **Service Providers**. The list of service providers you added appears.
3. Locate the service provider you want to delete and click on the corresponding **Delete** link.

<table>
<thead>
<tr>
<th>Service Provider ID</th>
<th>Description</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workday</td>
<td>Workday SSO</td>
<td></td>
</tr>
<tr>
<td>Playground</td>
<td>Request path authentication with WSO2 Playground</td>
<td></td>
</tr>
<tr>
<td>Salesforce</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lifesize</td>
<td>Lifesize Portal</td>
<td></td>
</tr>
</tbody>
</table>

4. Confirm your request in the WSO2 Carbon window. Click the **Yes** button.

### Related Topics

See the following topics for information on configuring service providers using different specifications.

- See the [Single Sign-On](#) topic for details on how to configure a service provider for single sign-on using different specifications.
- See [Identity Provisioning](#) for information on configuring inbound and outbound provisioning with a service provider.

See the following topics to configure different applications as service providers in Identity Server.

- Configuring SimpleSAMLphp as a Service Provider
- Logging in to Salesforce using the Identity Server
- Logging in to Salesforce with Facebook
- Logging in to Salesforce with Integrated Windows Authentication
- Logging in to WSO2 Products via the Identity Server