
Service Provider

Definition

The Service Providers are standard applications that rely on Identity Providers to let the users log in.

Join federation

To join the federation, the service provider management team must deliver its "Metadata". The service provider Metadata describes how the service providers behave:

- Which security algorithms does it support.
- The public portion of its signing and encrypting keys.
- The SAML protocol does it support.
- The URL of each SAML protocol endpoint.
- Contact information.

Standard attributes

The standard attributes depend on the Service provider type.

SAML

To **enable External SAML protocol** you can visit the [Authentication page](#). Also, on that page you could download the metadata XML file.

Identification

- **Identifier:** public name of the service provider. It must be unique
- **Name:** friendly user name or brief description.

Service configuration

- **Metadata:** you must provide the identity provider metadata. You can either copy it from the Soffid Identity Provider page, or instruct the service provider to download the federation metadata by itself.

To publish the federation members' metadata, the main sync server exports the member's metadata at the path **/SAML/metadata.xml**. Thus, if your sync server is listening at **soffid1.your.domain**, you can get the whole federation metadata document from:

<https://soffid1.your.domain:760/SAML/metadata.xml>

After some seconds, up to five minutes, every federation member will notice any change.

Login rules

- **Allow impersonations:** Soffid allows a service provider to connect to another service provider in a controlled manner. Here you can write the target application URL.
- **UID Script:** script to compute the user name to pass to the target application
- **Ask for consent**
- **Roles required to login:** roles that the user must have to be able to connect to the system
- **System where an enabled account is required:** System where it will be necessary for the user to have an account in order to log in.

You can visit the [Openid-connect to SAML interoperability page](#) for more detailed information.

SAML API client

Identification

- **Identifier:** public name of the service provider. It must be unique
- **Name:** friendly user name or brief description.
- **Organization:** company name of the external IdP.
- **Contact:** email address of the external IdP.

Service configuration

- **Metadata**

Leave it blank as Soffid IdP will fulfill it for you.

The metadata will be created when the network data and SAML Security data.

Login rules

- **Allow impersonations:** Soffid allows a service provider to connect to another service provider in a controlled manner. Here you can write the target application URL.
- **UID Script:** script to compute the user name to pass to the target application.
- **Ask for consent**

You can visit the [Openid-connect to SAML interoperability page](#) for more detailed information.

Network

- **Host name:** public application host name that wants to be a service provider. A fully qualified name should be used.
- **Standard port:** public application port number.
- **Disable SSL:** check it, selected value Yes, if you want to use plain TCP connections. In another case, it will be needed to comply with additional fields:
- **Assertion path:** URL to receive the response.

SAML Security

- **PublicKey:**
 - Clicking on the **Generates public / private key** button, a new private key pair will be generated. Once the private key pair is generated, you could generate a certificate request file, also known as PKC#10 or CSR file. The certificate authority will be able to create a certificate for you using this certificate request. Once you have created the public/private key, you could run other new functions:
 - **Change public/private key:** this allows you to change the public/private key generated previously.
 - **Delete public/private key:** this allows you to delete the public/private key generated previously.
 - **Generate PKCS10:** generates a PKCS10 file (Certification request standard).
 - Clicking on the **Upload PKCS12 file** button it will be able to upload a PKCS#12 file. That file must contain the private and public keys and the server certificate as well. Mind that PKCS#12 file use to be protected by a PIN.
- **Certificate chain:** text certificate chain created with one of the previous options.

OpenID Connect

Identification

- **Identifier:** public name of the service provider. It must be unique.
- **Name:** friendly user name or brief description.

Login rules

- **Allow impersonations:** Soffid allows a service provider to connect to another service provider in a controlled manner. Here you can write the target application URL.
- **UID Script:** script to compute the user name to pass to the target application.
- **Ask for consent**
- **Roles required to login:** roles that the user must have to be able to connect to the system
- **System where an enabled account is required:** System where it will be necessary for the user to have an account in order to log in.

You can visit the [Openid-connect to SAML interoperability page](#) for more detailed information.

OpenID authorization flow

- **Implicit:** application server redirects the end user to the IdP, that in turn, returns the OAuth token along with the OpenID token.
 - **Authorization code:** application server redirects the user to the IdP, which in turn, returns an authorization code that can be used to retrieve the token and the OpenID token from the token endpoint.
 - **User's password:** the server access directly to the token endpoint, sending the username and password, to retrieve the OAuth and OpenID token. This mechanism is highly insecure, as allows unauthenticated clients to impersonate end users
 - **User's password + Client credential:** it is a secure version of the previous one, requiring the client to use its client secret.
 - **Client id:** the identifier used by the application server.
 - **Client secret:** password used by the application server. It is used in the Authorization code flow as well as "User's password + Client credentials" flow.
 - **Response URL:** set the URL to return the control after authenticating a user.
 - **RP-Initiated logout response URL's**
 - **Front-channel logout endpoint**
 - **Back-channel logout endpoint**
 - **OAuth Session timeout (secs):** time in seconds that will take the OAuth session. The OAuth has its own life cycle, regardless of the session timeout.
 - **Allowed scopes:** you can define a scope list with the proper scopes that users will need to interact with the final system.
 - **openid:** default scope.
 - **custom scopes:** you can add the custom scopes that can be requested by the service provider.
 - *****: the scope * means that any scope requested by the service provider will be granted.
-

OpenID Connect Dynamic Registration

Identification

- **Identifier:** public name of the service provider. It must be unique
- **Name:** friendly user name or brief description.

Login rules

- **UID Script:** script to compute the user name to pass to the target application.
- **Ask for consent**
- **Roles required to login:** roles that the user must have to be able to connect to the system.
- **System where an enabled account is required:** System where it will be necessary for the user to have an account in order to log in.

OpenID authorization flow

- **Implicit:** application server redirects the end user to the IdP, that in turn, returns the OAuth token along with the OpenID token.
- **Authorization code:** application server redirects the user to the IdP, which in turn, returns an authorization code that can be used to retrieve the token and the OpenID token from the token endpoint.
- **User's password:** the server access directly to the token endpoint, sending the username and password, to retrieve the OAuth and OpenID token. This mechanism is highly insecure, as allows unauthenticated clients to impersonate end users
- **User's password + Client credential:** it is a secure version of the previous one, requiring the client to use its client secret.
- **Sector identifier URI**
- **Allowed scopes:** you can define a scope list with the proper scopes that users will need to interact with the final system.
 - **openid:** default scope.
 - **custom scopes:** you can add the custom scopes that can be requested by the service provider.
 - *****: the scope * means that any scope requested by the service provider will be granted.

Registration token

- **Token:** unique identifier
 - **Valid until:** maximum validity date
 - **Allowed servers:** maximum number of servers that can be registered
-

Cas client

Identification

- **Identifier:** public name of the service provider. It must be unique.
- **Name:** friendly user name or brief description.

Login rules

- **Allow impersonations:** Soffid allows a service provider to connect to another service provider in a controlled manner. Here you can write the target application URL.
- **UID Script:** script to compute the user name to pass to the target application.
- **Ask for consent**
- **Roles required to login:** roles that the user must have to be able to connect to the system
- **System where an enabled account is required:** System where it will be necessary for the user to have an account in order to log in.

CAS configuration

- **Response URL**
 - **Logout response URL**
-

Radius client

Identification

- **Identifier:** public name of the service provider. It must be unique.
- **Name:** friendly user name or brief description.

Login rules

- **UID Script:** script to compute the user name to pass to the target application.
- **Roles required to login**
- **System where an enabled account is required**

Radius configuration

- **Source IPs:** origin IP or origin IP range.
 - **Radius secret:** password
-

TACACS+

Identification

- **Identifier:** public name of the service provider. It must be unique.
- **Name:** friendly user name or brief description.

Login rules

- **Roles required to login**
- **System where an enabled account is required**

Tacacs+ configuration

- **Source IPs:** origin IP or origin IP range.
- **Tacacs+ secret:** password.
- **Authorization rules:** allows you to add additional authorization rules to elevate privileges. Available context variables:
 - **user:** remote user name
 - **priv_level:** privilege level
 - **remote_address:** remote address
 - **port:** port
 - **optionalArguments:** modifiable map of optional attributes.
 - **mandatoryArguments:** modifiable map of mandatory attributes.
 - **return** true if the action is authorized.

<https://www.rfc-editor.org/rfc/rfc8907.html>

Revision #52

Created 8 September 2021 09:43:27 by pgarcia@soffid.com

Updated 11 April 2023 14:15:57 by pgarcia@soffid.com