

# Federation members

- Entity Group
- Identity Provider
- Service Provider
- Virtual Identity Provider

# Entity Group

## Description

An entity group is just like a folder that allows you to manage different kinds of federation members. One of the most common ways to group federation members is by trust level.

When you create an entity group, the Identity Providers and the Service Providers records will be displayed. Then you could add identities and services selecting the proper record.

## Screen overview

Entity Group :

test-demoldP\*

Url Metadata :

Url Metadata

<input type="checkbox"/>	<div>Providers</div>
	<div>Filter</div>
<input type="checkbox"/>	Identity Providers
<input type="checkbox"/>	Service Providers

Displayed rows: 2

## Standard attributes

- **Entity Group:** name of the group.
- **Url Metadata:** will be the URL of an external entity group when the entity group was external.
- **Providers:** by default, it creates two groups, an identity provider and a service provider.

# Identity Provider

## Description

“ An identity provider (abbreviated IdP or IDP) is a system entity that creates, maintains, and manages identity information for principals and also provides authentication services to relying applications within a federation or distributed network.

An Identity Provider is responsible for identifying users. Also, it is responsible for giving service providers information regarding the identified user.

Soffid allows you to configure different identity providers, you can choose the best option for you by selecting the IdP type:

- **Soffid IdP:** identifies the identity provider implemented by Soffid. Soffid IdP implements both OpenID-Connect and SAML.
- **External SAML IdP:** is used to identify providers not implemented by Soffid. For instance, it could be an ADFS (Active Directory Federation Services) or Shibboleth identity provider.
- **OpenID-Connect:** is used for third-party identity providers, like ADFS.
- **Facebook:** if you select that option, OAuth2 will be used to identify Facebook users. You will need to register Soffid as a Facebook application to use it.
- **Google:** if you select that option OpenID-Connect will be used to identify Google users. You will need to register Soffid as a Google application to use it.
- **LinkedIn:** if you select that option, OAuth2 will be used to identify LinkedIn users. You will need to register Soffid as a LinkedIn application to use it.

To create an identity provider, it is advisable to install a dedicated sync server. It can be configured as a proxy sync server as it does not need direct access to the Soffid database. Instead, it will connect to the main sync server to get users and federation information.

For more information about how to configure a dedicated sync server, you can visit the [Install Sync server page](#).

# Standard attributes

The fields for each IdP type are detailed below:

## Soffid IdP

### Identification

- **publicID:** unique name to identify the identity provider. The name has to be the same as the Public ID of the Soffid Identity Provider agent.
- **Name:** friendly user name.
- **Organization:** company name of the external IdP.
- **Contact:** email address of the external IdP.

### Service Configuration

- **Metadata:** the Metadata for an Identity Provider defines how this Identity Provider delivers its service:
  - Which security algorithms does it support.
  - The public portion of its signing and encrypting keys.
  - The SAML protocols do it support.
  - The URL of each SAML protocol endpoint.
  - Contact information.

The Metadata is the information that any application needs to use the IdP. That is an XML file that contains the public encryption keys and the services provided

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

The metadata will be created when the network data and SAML Security data. Restarting the sync server will be necessary to fill in the Metadata.

### Network

- **Host name:** public hostname that will be used by users and service providers. The full qualified name should be used.
- **Allow IdP to be included inside an IFRAME:** Soffid allows you to configure the Identity Provider to be included within a IFRAME. If this option is updated, the Sync Server must be restarted. *This attribute will be available in Federation addon 3.5.37 or higher.*
- **Network ports:**
  - **Behind a reverse proxy**
  - **Reverse proxy port number:** port where the reverse proxy is listening.
  - **Reverse proxy incoming address:** IP addresses allowed to make calls to the reverse proxy.

- **Port:** TCP port number used by the identity provider. By default, TLS will be used (default 1443).
- **Encryption:** encryption type is only allowed behind a reverse proxy.
- **Support PROXY protocol v2:** protocol between the reverse proxy and the Identity Provider.
- **Accept client certificate**
- **Certificate header:** certificate data header (only behind a reverse proxy).
- **Excluded protocols:** encryption protocols to be excluded.

Image

Behind a reverse proxy :

Yes ☐

Reverse proxy port number :

443

Reverse proxy incoming address :

172.18.0.\*

Port :

1443

Encryption :

TLSv1.3

Support PROXY protocol v2 :

☐ No ☐

Accept client certificate :

Yes ☐

Certificate header :

X-SSL-CERT

Excluded protocols :

Excluded protocols

Warning: The sync server must be restarted to apply network changes

Close

- **TLS PublicKey:** there are three available options
  - **Leave in blank** and Soffid IdP will generate a self-signed certificate.
  - 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 PKCS#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:** allows you to change the public/private key generated previously.
    - **Delete public/private key:** 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.
- **TLS Certificate chain:** text certificate chain created with one of the previous options.

**Server certificate management:** there are two options for certificate management. You can visit the [Server certificate management page](#) for more information.

# 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:** allows you to change the public/private key generated previously.
    - **Delete public/private key:** 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 to contain the private an 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.

## Session management

- **Session timeout (secs):** time in seconds that will take the session. If the user has been authenticated, and later is requested to authenticate again, the user will be authenticated without any intervention as long as the timeout has not been elapsed.
- **oAuth Session timeout (secs):** time in seconds that will take the oAuth session. The oAuth has its own life cycle, regardless the session timeout.
- **Maximum session duration (secs) :** maximum time during which session can be renewed
- **SSO Cookie name:** name of the cookie that will keep the session id, you can change the name. This SSO cookie is not really needed, as the identity provider will store a session cookie to track the SSO session. This SSO cookie is needed in two circumstances:
  - When the identity provider is restarted, the session cookie is lost. This SSO Cookie allows the identity provider to restart the lost session.
  - When you have more than one identity provider instance, this cookie allows all the identity providers to handle the session as if only was one identity provider. The SSO cookie can be allocated by any identity provider, and it will be accepted by any other one.
- **SSO Cookie domain:** is needed when you have more than one identity provider instance and they are using different host names. If all the identity providers are serving the same virtual host name, the SSO Cookie domain will be needed.

## Authentication

- **Authentication methods:** matrix to define the authentication methods that will be required to successfully authenticate the user. Each row indicates the first authentication method, and each column indicates the second factor to use.
  - Password

- Kerberos
- External IdP
- OTP
- Email
- SMS
- PIN Certificate
- FIDO
- Push
- **Adaptive authentication:** that option allows you to add an additional authentication matrix which will be run when the condition defined was complied with. That is the way to change the authentication method depending on the environment.
  - **Description:** rule description to identify it.
  - **Condition:** script to enable that rule. The result of the rule must be true or false.
 

There are some available vars to create the condition. You can visit the [Condition for Adaptive authentication page](#) for more information and some examples.
  - **Matrix:** to define the authentication methods that will be required to successfully authenticate the user. Each row indicates the first authentication method, and each column indicates the second factor to use.
- **Always ask for credentials:** if checked (the selected value is Yes), the IdP will always request credentials from users who meet the condition defined in this rule.
- **Register OTP when required:** if it is checked (selected value is Yes), Soffid will allow registering the OTP to users who meet the condition and do not have one previously.
- **Kerberos domain:** allows you to pick up a file to configure the Kerberos authentication method. For more information, you can visit the [How to enable Kerberos authentication page](#).

## Advanced Authentication

- **Allow user to recover password:** if it is checked (selected value is Yes), and the password recovery addon is installed, the user will be allowed to execute the password recovery mechanism.
- **Register OTP when required:** if it is checked (selected value is Yes), Soffid will allow to register the new OTP to the user during the login process.
- **Allow user to self-register:** if it is checked (selected value is Yes), the user will be allowed to register itself. This option sends an email to the user to verify the email address is correct, and then lets the user to enter a new password.
  - **Registration process:** workflow selected to create the new identity.
  - **User Type:** identifies the password policy that is to be applied. More information on this link [User Type](#).
  - **Primary Group:** select which organization unit this user belongs to.
- **Register identities identified by external IdPs:** allows Soffid IdP to automatically register a new identity when a user authenticates with a third-party IdP, and this identity does not exist yet in Soffid database. Furthermore, at the third party IdP configuration page, one can tune how this identity is going to be created.

- **Store last user name in browser:** allows the browser to save the last user name when Yes is selected.
- **Enable reCaptcha v3 service:** (\*) helps to keep save your website. You can enable it by selecting the Yes option. When you select the Yes option, you must fill in the following fields:
  - **Captcha site key:** this key is used to invoke the reCAPTCHA service
  - **Captcha site secret:** the secret key to communicate your web site with reCAPTCHA service. This secret key authorizes the communication.
  - **Captcha threshold (1 for highest confidence, 0 for low confidence):**

## Profiles

A profile is a protocol or subset of protocols implemented by the Identity Provider. There are some accepted protocols, those allows a custom config dependent on the selected profile.

You can visit the [Profiles chapter](#) for more information about each one.

## Look and feel

Soffid allows you to personalize your login page by adding some style elements, as well as header and footer elements.

- **Logo:** this logo will be displayed for user in Windows desktop.
- **CSS Style:** allows you to add a CSS style for your login page.
- **Html header:** allows you to add an Html header.
- **Html footer:** allows you to add an Html footer.
- **Language (2 characters code)**

## External SAML IdP

### Identification

- **publicID:** unique name to identify the identity provider.
- **Name:** friendly user name.
- **Organization:** company name of the external IdP.
- **Contact:** email address of the external IdP.

### Service Configuration

- **Metadata:** the Metadata for an Identity Provider defines how this Identity Provider delivers its service:
  - Which security algorithms does it support.
  - The public portion of it's signing and encrypting keys.



- The SAML protocols does it support.
- The URL of each SAML protocol endpoint.
- Contact information.

The Metadata is the information that any application need to use the IdP. That is an XML file that contains the public encryption keys and the services provided

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

## Login Rules

- **User regular expression:** regular expression to detect users of this identity provider.
- **Login hint script:** script to help to login. Return the text to help.
- **Identity provisioning script:** script to bind or register a new identity. Return the user name of the owner identity for the authenticated account.

## OpenID-Connect

### Service Configuration

- **Metadata:** there are some required parameters:
  - **authorization\_endpoint:** contains the OAuth endpoint to forward the user to get the authorization token.
  - **token\_endpoint:** contains the OAuth endpoint to get the access token, based on the authorization token got at previous step.
  - **userinfo\_endpoint:** if remote IdP is OpenID-connect compliant, the token endpoint should have sent an access token along a JWT OpenID token containing user claims. If this is not the case, Soffid will use this user\_info endpoint to fetch user claims. This mechanism is needed for OAuth2 servers.
  - **scopes\_supported:** The list of scopes specified here will be used at first step, when redirecting the user to the authorization endpoint.

```
{
  "authorization_endpoint":
    "https://server/oauth2/auth",
  "token_endpoint":
    "https://server/oauth2/token",
  "userinfo_endpoint":
    "https://server/oauth2/userinfo",
  "scopes_supported": [
    "openid","email","profile"]
}
```

- **oAuth key:** is the identifier token generated by the oAuth server.
- **oAuth secret:** is the secret generated by the oAuth server.

The Metadata is the information that any application need to use the IdP. That is an XML file that contains the public encryption keys and the services provided

## Login rules

- **User regular expression:** regular expression to detect users of this identity provider.
- **Login hint script:** script to help to login. Return the text to help.
- **Identity provisioning script:** script to bind or register a new identity. Return the user name of the owner identity for the authenticated account.

```
sn =
attributes{"screen_name"};
i = sn.indexOf(" ");
if (i > 0) {
    user.firstName = sn.substring(0,
    i);
    user.lastName =
    sn.substring(i+1);
} else {
    user.firstName = "?";
    user.lastName = sn;
}
return attributes{"name"};
```

## Facebook

### Identification

- **publicID:** unique name to identify the identity provider. Soffid will fulfill with the Facebook URL.
- **Name:** friendly user name.
- **Organization:** company name of the external IdP.
- **Contact:** email address of the external IdP.

### Service Configuration

- **Click here to obtain a client id and client secret:** allows you to get the oAuth key and secret.
- **oAuth key:** is the identifier token generated by the oAuth server.

- **oAuth secret:** is the secret generated by the oAuth server.

## Login rules

- **User regular expression:** regular expression to detect users of this identity provider.
- **Login hint script:** script to help to login. Return the text to help.
- **Identity provisioning script:** script to bind or register a new identity. Return the user name of the owner identity for the authenticated account.

## Google

### Identification

- **publicID:** unique name to identify the identity provider. Soffid will fulfill with the Google URL.
- **Name:** friendly user name.
- **Organization:** company name of the external IdP.
- **Contact:** email address of the external IdP.

### Service Configuration

- **Click here to obtain a client id and client secret:** allows you to get the oAuth key and secret.
- **oAuth key:** is the identifier token generated by the oAuth server.
- **oAuth secret:** is the secret generated by the oAuth server.

## Login rules

- **User regular expression:** regular expression to detect users of this identity provider.
- **Login hint script:** script to help to login. Return the text to help.
- **Identity provisioning script:** script to bind or register a new identity. Return the user name of the owner identity for the authenticated account.

## Linkedin

### Identification

- **publicID:** unique name to identify the identity provider. Soffid will fulfill with the LinkedIn URL.
- **Name:** friendly user name.
- **Organization:** company name of the external IdP.
- **Contact:** email address of the external IdP.

### Service Configuration

- **Click here to obtain a client id and client secret:** allows you to get the OAuth key and secret.
- **OAuth key:** is the identifier token generated by the OAuth server.
- **OAuth secret:** is the secret generated by the OAuth server.

## Login rules

- **User regular expression:** regular expression to detect users of this identity provider.
  - **Login hint script:** script to help to login. Return the text to help.
  - **Identity provisioning script:** script to bind or register a new identity. Return the user name of the owner identity for the authenticated account.
- 

*(\*) What is CAPTCHA --> <https://support.google.com/a/answer/1217728?hl=en>*

*(\*) <https://www.google.com/recaptcha/about/>*

# 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.
- **NameID format:**
  - Persistent

- 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:

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

- **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**
- **NameID format:**
  - Persistent
  - Email
  - Unspecified
  - Transient

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.

 **Image**



Search

?

[Main Menu](#) > [Administration](#) > [Configuration](#) > [Web SSO](#) > [Identity & Service providers](#) ◀ 8 / 8

**Identification**

Type :

Identifier :

Name :

Organization :

Contact :

**Service configuration**

Metadata :

NameID format :

**Login rules**

Allow impersonations :

UID Script :

Ask for consent :

Roles required to login :

System where an enabled account is required :

**Network**

Host Name :

Standard port :

Disable SSL :

Assertion path :

**SAML Security**

PublicKey :

Certificate chain :

## 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.

Image

sof.id

Main Menu

Administration

Configuration

Web SSO

Identity & Service providers

6 / 7

Search

Identification

Type :

OpenID Connect

Identifier :

angularApp

Name :

angularApp

OpenID authorization flow

Implicit :

☒ Yes
 ☐ No

Authorization code :

☐ Yes
 ☒ No

User's password :

☐ Yes
 ☒ No

User's password + Client credentials :

☐ Yes
 ☒ No

Client id :

angularApp

Client secret :

\*\*\*\*

Sector identifier URI :

Response URL :

http://localhost:4204/home

Response URL

RP-Initiated logout response URL's :

RP-Initiated logout response URL's

Front-channel logout endpoint :

Front-channel logout endpoint

Back-channel logout endpoint :

Back-channel logout endpoint

oAuth Session timeout (secs) :

oAuth Session timeout (secs)

Allowed scopes

Scope name	Required roles
<input checked="" type="checkbox"/> *	
<input type="checkbox"/> email	
<input type="checkbox"/> openid	
<input type="checkbox"/> profile	

Displayed rows: 4

Note that the scope 'openid' will always be accepted.

A scope with no roles will be granted always.

A scope with roles will be granted if the identified user has the required role.

Add the scope \* to allow any scope

Undo

Apply changes

Login rules

Allow impersonations :

UID Script :

Script to compute the user name to pass to the target application

Ask for consent :

☒ Yes
 ☐ No

Roles required to login :

Roles required to login

System where an enabled account is required :

System where an enabled account is required

# 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

 Image

[Main Menu](#) > [Administration](#) > [Configuration](#) > [Web SSO](#) > [Identity & Service providers](#) 8 / 8

### Identification

Type :

Identifier :

Name :

### OpenID authorization flow

Implicit : ☒ No

Authorization code : ☒ No

User's password : ☒ No

User's password + Client credentials : ☒ No

Sector identifier URI :

Scope name	Required roles
<input type="text" value="Filter"/>	<input type="text" value="Filter"/>

Displayed rows: 0

Note that the scope 'openid' will always be accepted.  
A scope with no roles will be granted always.  
A scope with roles will be granted if the identified user has the required role.  
Add the scope \* to allow any scope

### Login rules

UID Script :

Ask for consent : ☒ No

Roles required to login :

System where an enabled account is required :

### Registration token

Token :

Allowed servers :

## 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

Image

sofid

Search

Main Menu > Administration > Configuration > Web SSO > Identity & Service providers

8 / 8

Identification

Type : CAS client

Identifier \*

Name

CAS Configuration

Response URL :

RP-Initiated logout response URL's :

RP-Initiated logout response URL's

Login rules

Allow impersonations :

UID Script :

Ask for consent : No

Roles required to login :

System where an enabled account is required :

Target application URL

Script to compute the user name to pass to the target application

Roles required to login

System where an enabled account

Undo

Apply changes

## 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

Image

[Main Menu](#) > [Administration](#) > [Configuration](#) > [Web SSO](#) > [Identity & Service providers](#) ◀ 8 / 8

### Identification

Type :

Identifier :

Name :

### Radius configuration

Source IPs :

Radius secret :

Client certificate :

Free radius agent : ☐ Yes ☒ No

### Login rules

Roles required to login :

System where an enabled account is required :

# 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.

Image



?

[Main Menu](#) > [Administration](#) > [Configuration](#) > [Web SSO](#) > [Identity & Service providers](#) ◀ 8 / 8

### Identification

Type :

Identifier :

Name :

### Tacacs+ configuration


Source IPs :

Tacacs+ secret :

Authorization rules

<input type="checkbox"/>	Authorization rules
	<input type="text" value="Filter"/>



Displayed rows: 0



### Login rules

Roles required to login :

System where an enabled account is required :

 Undo  Apply change

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

## WS-Federation

### 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.

### WS-Federation

- **Response URL**

 Image

### Identification

Type :

Identifier :

Name :

### WS-Federation

Response URL :

### Login rules

Allow impersonations :

UID Script :

Ask for consent : ☐ Yes ☒ No

Roles required to login :

System where an enabled account is required :

[Undo](#)[Apply change](#)



# Virtual Identity Provider

## Definition

A single identity provider usually offers different profiles or service levels to different service providers. To be able to define this behavior, any Identity Provider can be split into many virtual identity providers. Those identity providers will be served by the same actual identity provider, but they will have different profile configurations.

## Standard attributes

### Identification

- **publicID:** unique name to identify the identity provider.
- **Name:** user friendly name to identify the identity provider.
- **Organization:** company name of the external IdP.
- **Contact:** email address of the external IdP.

### Service configuration

- **Metadata:** the Metadata for an Identity Provider defines how this Identity Provider delivers its service:
  - Which security algorithms does it support.
  - The public portion of its signing and encrypting keys.
  - The SAML protocols does it support.
  - The URL of each SAML protocol endpoint.
  - Contact information.

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

## SAML Security

- **Public key:**
  - **Generate public/private key:**
    - **Delete public/private key:** allows you to delete the public/private key generated previously.
    - **Generate PKCS10:** generates a PKCS10 file (Certification request standard)
  - **Upload PKCS12 file:** allows you 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.

## Authentication

- **Authentication methods:** matrix to define the authentication methods that will be required to successfully authenticate the user. Each row indicates the first authentication method, and each column indicates the second factor to use.
- **Adaptive authentication:** that option allows you to add additional authentication matrix which will be run when the condition defined was comply.
  - **Description:** rule description to identify it.
  - **Condition:** script to enable that rule. The result of the rule must be true or false.

There are some available vars to create the condition. You can visit the [Condition for Adaptive authentication page](#) for more information and some examples.

- **Matrix:** to define the authentication methods that will be required to successfully authenticate the user. Each row indicates the first authentication method, and each column indicates the second factor to use.

## Advances authentication

- **Allow user to recover password:** if it is checked (selected value is Yes), and the password recovery addon is installed, the user will be allowed to execute the password recovery mechanism.
- **Allow user to self-register:** if it is checked (selected value is Yes), the user will be allowed to register itself. This option sends an email to the user to verify the email address is correct, and then lets the user to enter a new password.
- **Registet identities identified by external IdPs:** allows Soffid IdP to automatically register a new identity when a user authenticates with a third-party IdP, and this identity does not exist yet in Soffid database. Furthermore, at the third party IdP configuration page, one can tune how this identity is going to be created.

## Profiles

A profile is a protocol implemented by the Identity Provider. There are some accepted protocols, those allows a custom config dependent on the selected profile

- OpenIDProfile
- SAML1ArtifactResolutionProfile
- SAML1AttributeQueryProfile
- SAML2ArtifactResolutionProfile
- SAML2AttributeQueryProfile
- SAML2ECPPProfile
- SAML2SSOProfile

You can visit the [Profiles chapter](#) for more information about each one.

## Service Providers

It will be necessary to bind any service provider to the virtual identity provider. When no such bind exists for a service provider, the actual identity provider profile configuration applies.