OpenID Connect features

This page describes all the OpenID Connect (OIDC) features supported by SURFconext. All features mentioned here are compliant with the OpenID Connect specification.

Getting claims (attributes)

Most services require extra information about the authenticated user, such as a name, email address or affiliation. In OpenID Connect (OIDC), this extra information comes in the form of **claims**, whereas in SAML, claims are called **attributes**. Check out our extensive documentation on claims here.

There is more then one way to get claims. By default you can use the **userinfo endpoint**. Almost any OpenID Connect libary or OIDC supported application has support for it. If your library or application expects the **claims to be present in the id_token**, you can request those using the "claims" request parameter.

Use our **OIDC Playground application** to see how this works.

Another alternative is that we always return the claims in the id_token. Please contact support if you want us to enable that option.

Supported flows

All OpenID Connect flows (code, implicit and hybrid) are supported by SURFconext. We strongly recommend you use the **code flow** for your application. If you have a client that is not able to keep a secret (e.g. a Mobile App, or a JavaScript client), using PKCE is required. This method is described below.

With the `response_type parameter` during the authorisation request you can specify which flow to use. The following values are allowed:

flow	reponse_type parameter
Authorization Code	`code`
Implicit	`id_token`
Implicit	`id_token token`
Hybrid	`code id_token`
Hybrid	`code token`
Hybrid	`code id_token token`

Proof Key for Code Exchange (PKCE)

PKCE is a security extension for public clients, originally for OAuth2. It mitigates attacks where the authorization code on public clients can be intercepted, such as Mobile or Desktop Apps which use callback URLs. For example, a rogue app can hijack the redirect URL and then obtain the authorisation code. These public clients typically do not have a secret configured when they exchange the code for an `access_token` at the token endpoint.

PKCE solves this problem as follows. When an authorisation request is made, the RP generates a secret, and for more security, a hash of that secret. Two additional parameters are used:

- `code_challenge=XXXXX`: The hash of the secret that has been generated
- `code_challenge_method=S256`: This can be either S256, meaning that the SHA-256 hash of the secret is given, or plain, meaning that the secret itself is transmitted.

The server then ensures the generated code and the supplied code_challenge are stored.

In the request to the token endpoint, the `code_verifier` is added, which is the secret that has been generated. The OIDC server then checks whether the secret is the same as the one in the original authorisation request. If not, no `access_token` and `id_token` are provided. This effectively binds the first authorisation request to the token request.

More information can be found in the specification: https://tools.ietf.org/html/rfc7636

Scopes

Currently, we only support the scope **openid**. You may use any of the other scopes as specified in the OpenID Connect specification (profile, email, address, phone) but these are silently ignored. You will receive the claims you requested during the deployment in the SP Dashboard.

Prompt = login

Adding the `prompt=login` parameter to the authorisation request will attempt to *disable* Single Sign-On, by instructing the user's Identity Provider to show the login screen and forcing the user re-authenticate. Please note that this greatly impacts the user experience, as users will have to re-enter their credentials at the IdP, even if they very recently already did so. Only use this parameter when it is *really* necessary. Don't hesitate to ask the SURFconext support team for help if you have doubts whether to apply it or not.

Important: Not all IdPs in SURFconext support this. You should also not rely on this feature as a security feature, since it is possible to disable it during a login request.

Request adding claims to the id_token

SURFconext supports the `claims` request parameter which allows RPs to have the claims in the `id_token` instead of in the user-info endpoint. Important: our minimal disclosure policy regarding the release of claims/attributes dictates which claims you are allowed to receive. You will not receive any claims you request which you are not allowed to receive.

More info on claims can be found here.

Refresh tokens

If you use OpenID Connect in combination with API security (i.e. you have a resource server) you can use refresh tokens in order to keep the lifetime of the access tokens shorter. Please refer to this page for more in depth information on refresh tokens.

Playground

In addition to this page, you can also play around with our playground application. This application allows you to review all features and, for instance, have a closer look at headers and responses with your application.