OpenID-Connect

OpenID-Connect

Introduction

> OpenID is an open standard and decentralized authentication protocol. It allows users to be authenticated by cooperating sites (known as relying parties, or RP) using a third-party service, eliminating the need for webmasters to provide their own ad hoc login systems, and allowing users to log into multiple unrelated websites without having to have a separate identity and password for each.

It is identity layer on top of the OAuth 2.0 protocol. OpenID-Connect is based on most modern protols. It uses JSON tokens, signed and optionally encripted using JWT standard, and uses simple REST as its transport protocol.

Sometimes referred as OpenID, must not be confused with an older and deprecated standard named OpenID.

---

https://openid.net/

https://en.wikipedia.org/wiki/OpenID#OpenID_Connect_(OIDC)

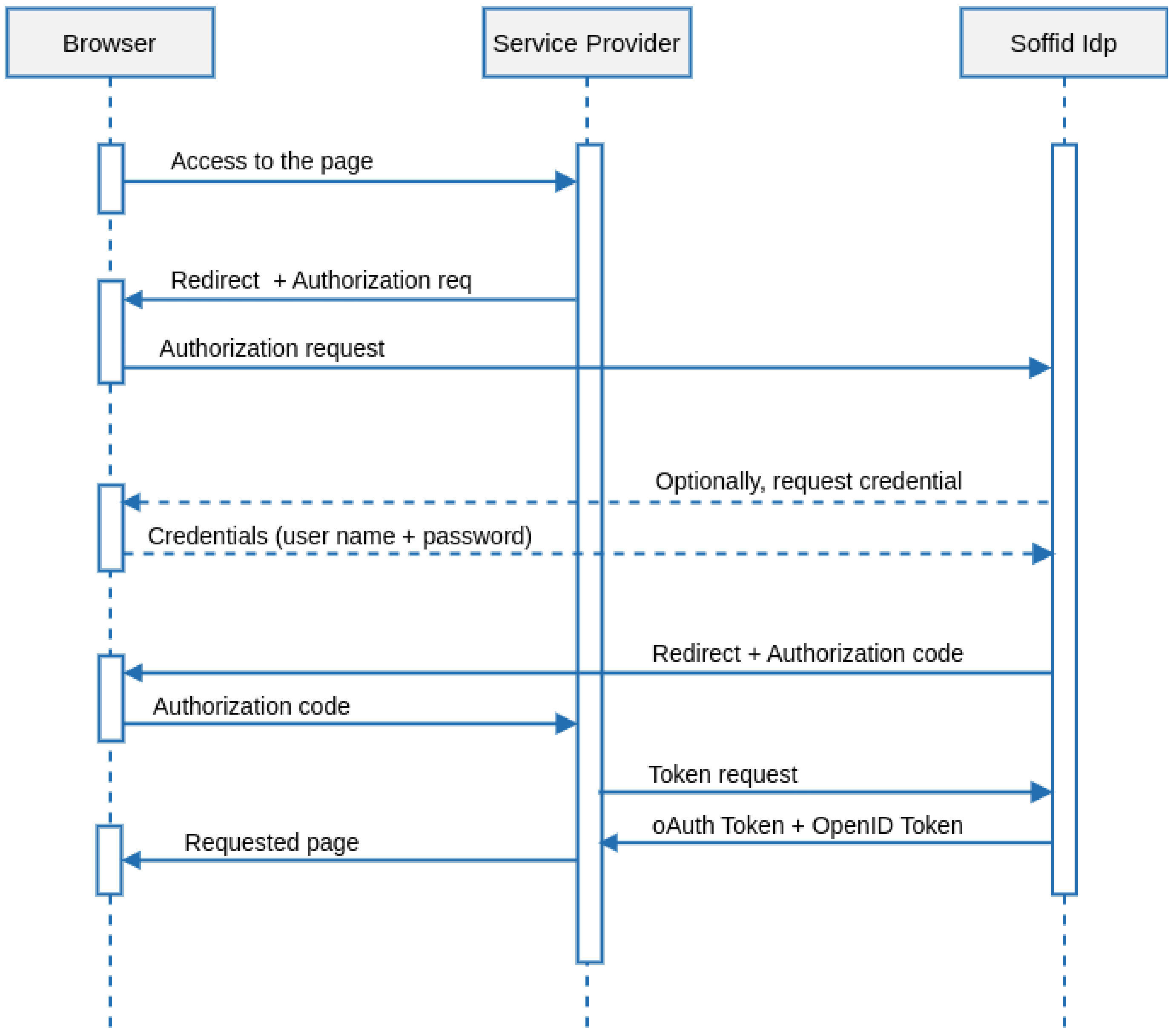
OpenID-Connect architecture

Introduction

OpenID is based on the well known protocol. It is easier to implement and deploy, as it does not require digital signature or encryption. The drawback is that it is significantly less secure. For example, the single logout protocol is not finished yet.

Single Log-in

The usual log-in process follows the next UML diagram:


Browser
Service Provider
Soffid Idp
Access to the page
Redirect + Authorization req
Authorization request
Optionally, request credential
Credentials (user name + password)
Redirect + Authorization code
Authorization code
Token request
oAuth Token + OpenID Token
Requested page

Description

1. User’s browser tries to get a web page from the service providers

2. . The service provider wants to authenticate the user identity. To get this, redirects the user to the identity provider, including the returning URL.

3. The authorization request is received by the identity provider. At this point, the identity provider verifies it is issued by an authorized service provider.

Next, the identity providers checks if the user browser does have an active SSO session. In such a case, skip to step 6.

4. The identity providers ask for credentials to the user.

5. The user enters its credentials. At this time, the identity provider verifies the user name and password are correct, and creates a new SSO session.

6. The identity provider redirects the user to the service provider, sending an authorization code.

7. The service provider connects to the identity provider, using its client id and client secret, as well as the authorization code.

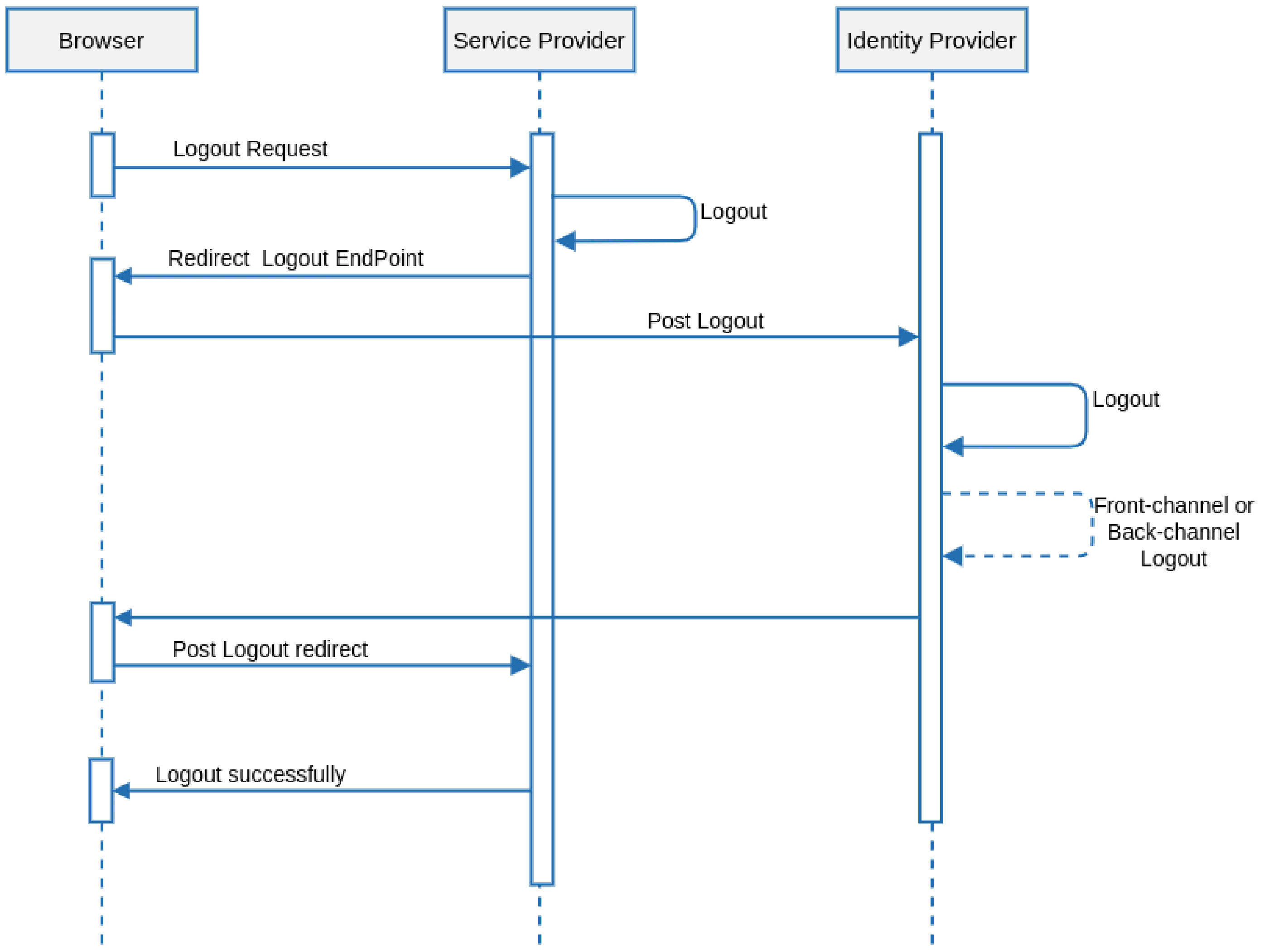
8. The identity provider verifies the authorization code and generates two tokens: the oAuth token and the OpenID token. The Auth token is a bare token that can be used by the service provider to perform additional requests.

The Openid token contains some user attributes. The included attributes and its value can vary depending on the service provider that will receive it. This token can be signed using JWT standard.

9. The service provider receives the both tokens, parsing the JSON document contained in the JWT OpenID token.

Single Log-out

One generic logout process diagram:



Description

1. The user requests to log out the application.

2. Logout in the Service Provider, for instance, delete cookies.

3. Redirect to the Identity Provider logout endpoint

4. Logout in the Identity Provider, for instance, delete cookies.

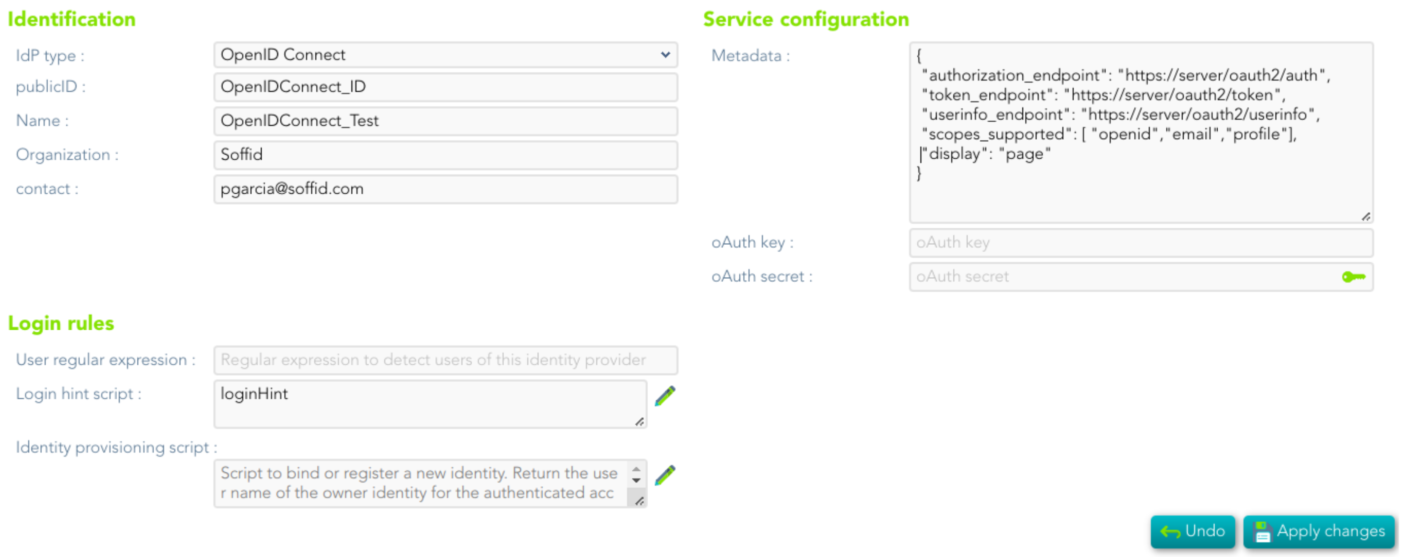
5. The Identity Provider can trigger logout from other Service Providers using Font-channel or Back-channel.

6. The Identity Provider redirects to the Service Provider EndPoint

7. The Service Provider returns successfully logout

OpenID-Connect example

Identity Provider



Identification

IdP type :	OpenID Connect
publicID :	OpenIDConnect_ID
Name :	OpenIDConnect_Test
Organization :	Soffid
contact :	pgarcia@soffid.com

Service configuration

Metadata :

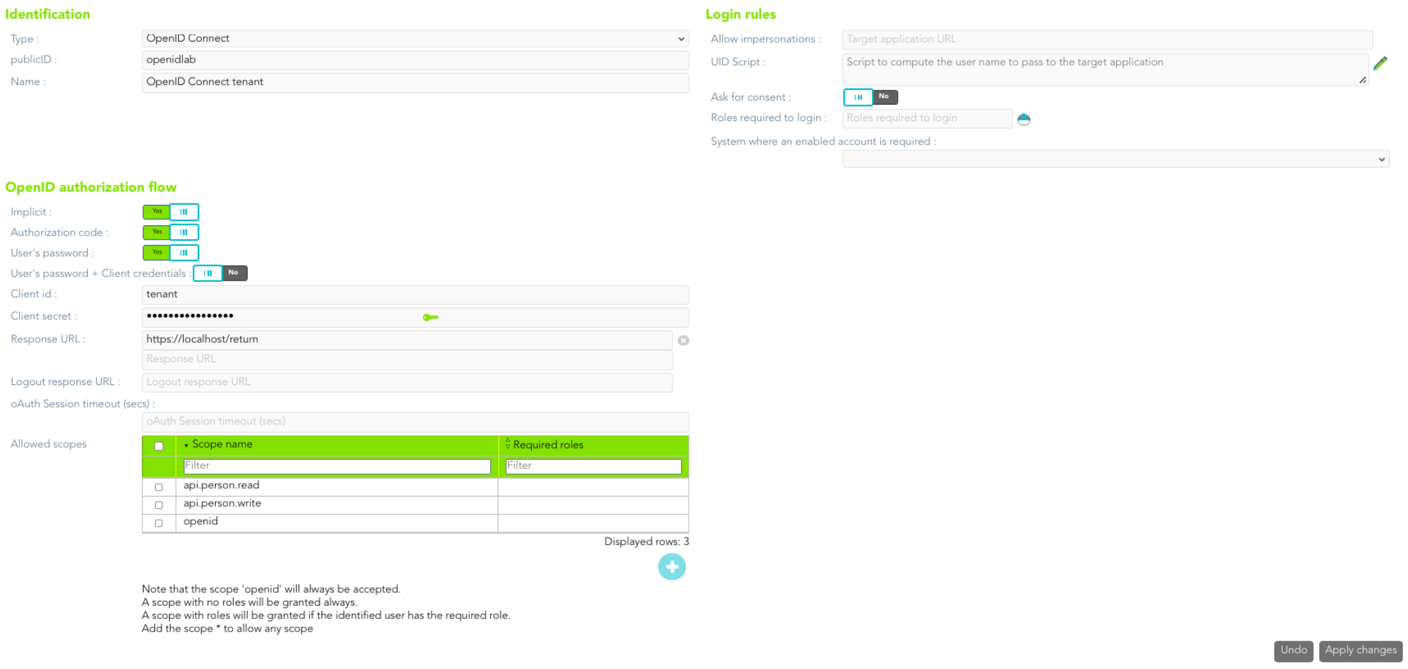
```
{
 "authorization_endpoint": "https://server/oauth2/auth",
 "token_endpoint": "https://server/oauth2/token",
 "userinfo_endpoint": "https://server/oauth2/userinfo",
 "scopes_supported": [ "openid","email","profile"],
|"display": "page"
}
```

oAuth key :	oAuth key
oAuth secret :	oAuth secret

Login rules

User regular expression :	Regular expression to detect users of this identity provider
Login hint script :	loginHint
Identity provisioning script :	Script to bind or register a new identity. Return the user name of the owner identity for the authenticated acc

Service Provider



Identification

Type :	OpenID Connect
publicID :	openidlab
Name :	OpenID Connect tenant

OpenID authorization flow

Implicit :	Yes
Authorization code :	Yes
User's password :	Yes
User's password + Client credentials :	No
Client id :	tenant
Client secret :	••••••••••••••••
Response URL :	https://localhost/return
	Response URL
Logout response URL :	Logout response URL
oAuth Session timeout (secs) :	oAuth Session timeout (secs)

Allowed scopes

	Scope name	Required roles
	Filter	Filter
☐	api.person.read	
☐	api.person.write	
☐	openid	

Displayed rows: 3

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

Allow impersonations :	Target application URL
UID Script :	Script to compute the user name to pass to the target application
Ask for consent :	No
Roles required to login :	Roles required to login
System where an enabled account is required :	