Developer Setup Guide: OpenID Connect Single Page Web App

Okta Inc.
301 Brannan Street
San Francisco, CA 94107

Raphael Londner

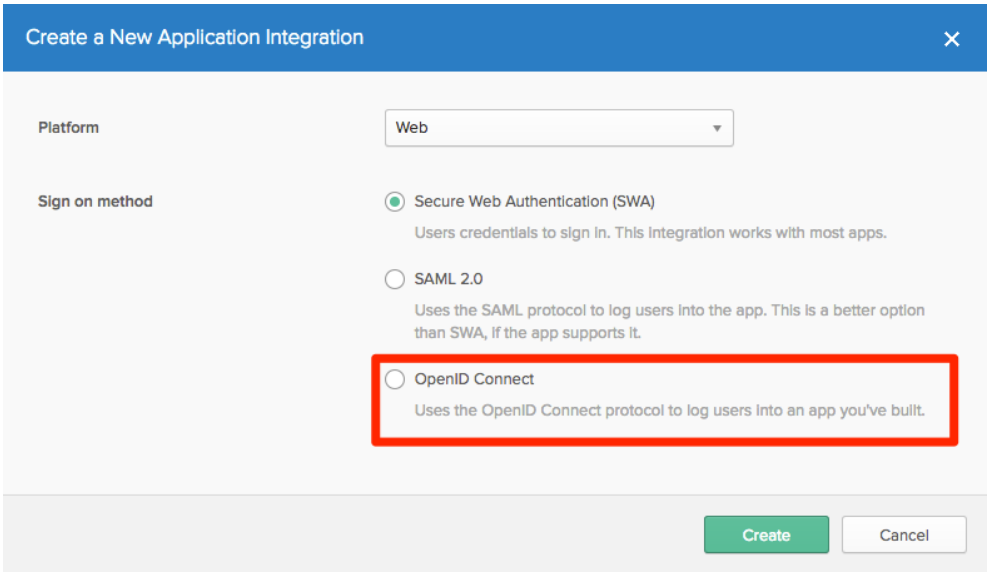
Introduction

This document highlights the process required to install and run the OpenID Connect Single Page Web Application sample available at https://github.com/oktadeveloper/okta-oauth-spa-authjs-osw. This sample demonstrates the use of the Okta Auth JavaScript SDK and the Okta Sign In Widget in a single page web app written in HTML and JavaScript (using jQuery).
The first section below will show how to use the code on your own server with a pre-configured Okta organization. The second section will describe how to configure an OpenID Connect/OAuth client application in your own Okta organization, as well as which changes are required in the sample code to make it work with your Okta client application.

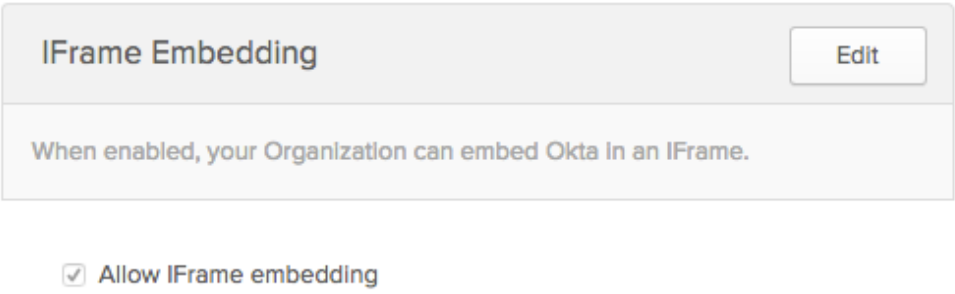
Running the sample with your Okta organization

Pre-requisites

1. Sign up at http://developer.okta.com/ for a free Developer Okta account if you don't have one yet.
2. Make sure OpenID Connect is enabled for your Okta organization (tied to your Developer account). To verify this, go to your Okta Admin dashboard, click on Applications in the top navigation bar, click on the **Add Application** button, then on the **Create New App** button. If you see the OpenID Connect option as shown in the screenshot below, you are good to go! Otherwise, please contact our Developer Evangelism team at developers@okta.com to have it enabled.



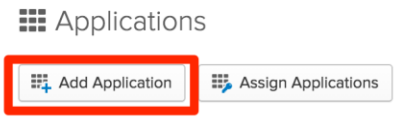
3. Go to **Settings** ➔ **Customization**, scroll down and make sure that **Allow Iframe embedding** is checked



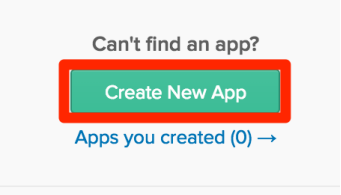
4. Click Directory in the top navigation bar and make sure you have at least one test user that you will be able to assign to your OpenID Connect application (see next section). If not, please follow the steps described in Using the Okta People Page to add a new Okta user.
5. Optionally, create Okta groups and assign your test user to them by following the instructions available in Importing and Using Groups in Okta. This will be helpful if you want to test the "groups" OAuth scope. **Important note**: Okta OpenID Connect currently does not support Active Directory groups in the "groups" scope, so make sure you use manually created Okta groups, not imported groups.
6. Please visit https://www.python.org/downloads/ to install Python, in case it's not already available. We use Python to run a simple HTTP server on your machine. You may want to use another web server (such as IIS Express with Visual Studio) if you're more familiar with it.
7. Download the GitHub repository at https://github.com/oktadeveloper/okta-oauth-spa-authjs-osw to your machine.

Creating the OpenID Connect application in Okta

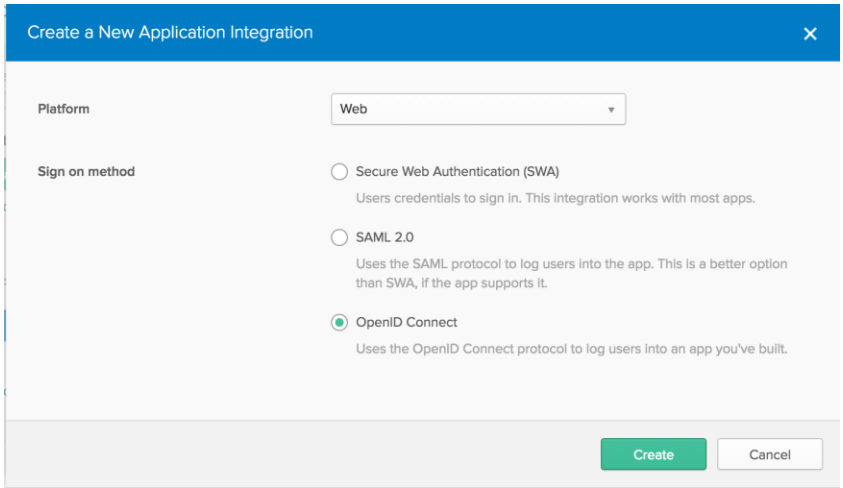
1. Sign in to your Okta organization as an organization administrator.
2. Go to the Admin section of your Okta organization by pressing the **Admin** button in the top-right corner of the home page.
3. In the top navigation, click on **Applications** and then press the **Add Application** button.



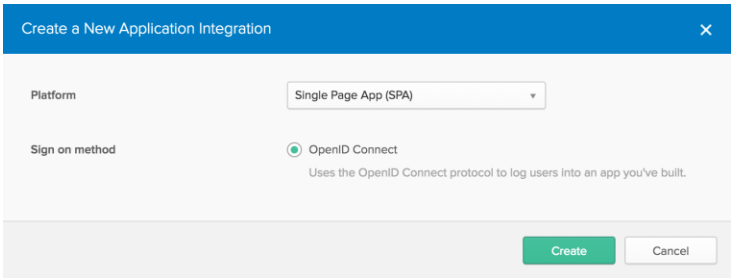
4. Press the **Create New App** button.



5. The following pop-up window shows up:

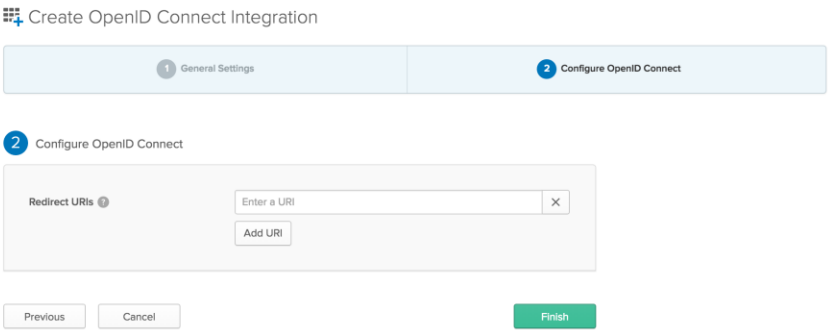


6. Select **Single Page App (SPA)** in the Platform combo box (the **OpenID Connect** option automatically becomes the only option in the radio list).

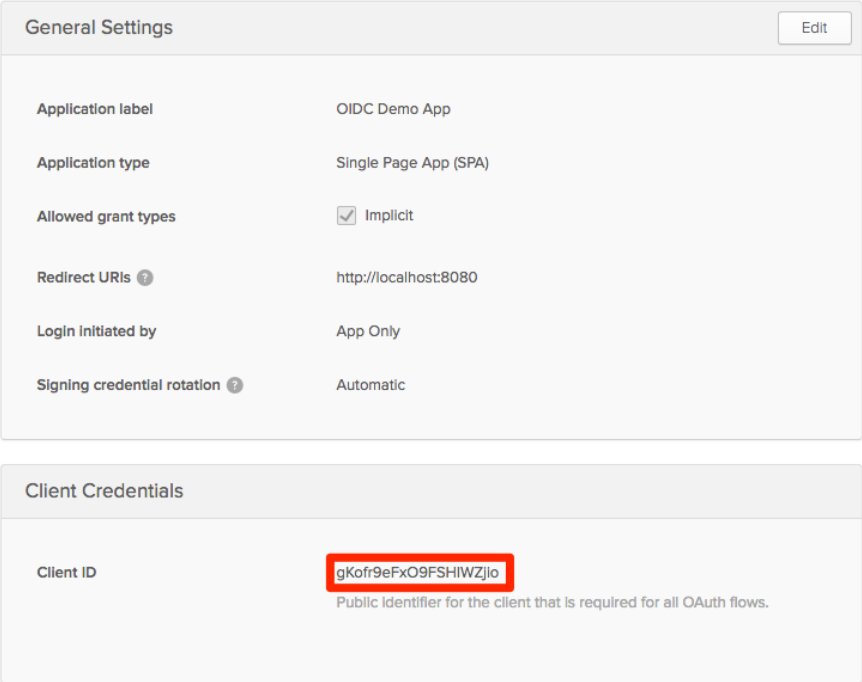


7. Press **Create**. In the next page that appears, enter an **Application Name** of your choice. This name will be used in your list of Applications (seen as an administrator), as well as on the chiclet that users assigned to your app may see on their Okta dashboard.

8. Press **Next**. The following page appears:



9. In the Redirect URIs field, enter http://localhost:8080 (or any other local url you want to use for the app running on your machine). Press the **Finish** button. A page similar to the following screenshot appears:

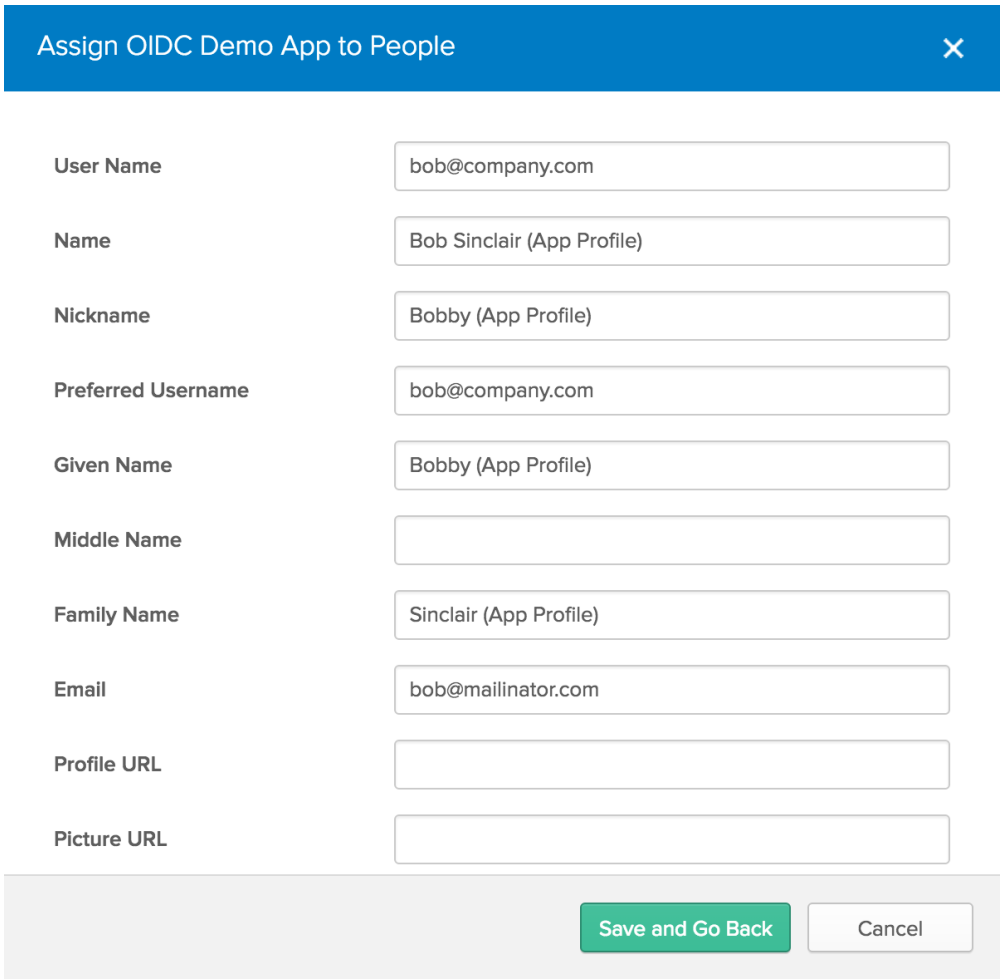


10. From the root of your local GitHub repository copy, please navigate to the JavaScripClient/js sub-folder and in the `config.js` file, change the following parameters:
    a. **orgUrl**: update to the url of your Okta preview org, such as https://acme.oktapreview.com
    b. **clientId**: update with the Client ID value highlighted in the screenshot above

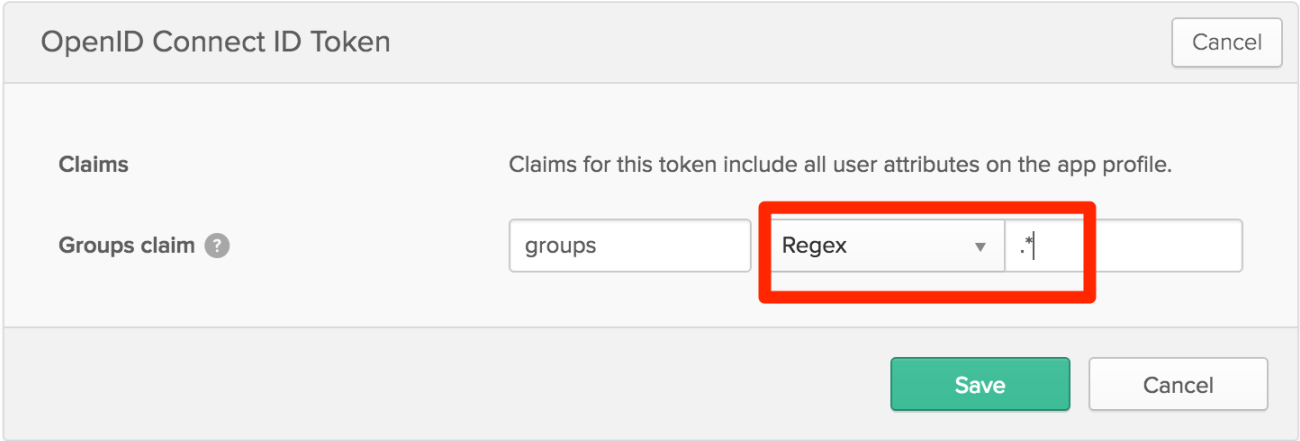
c. **redirectUri**: optionally update with the base url of your application, if different from http://localhost:8080

Configuring your Okta OpenID Connect application

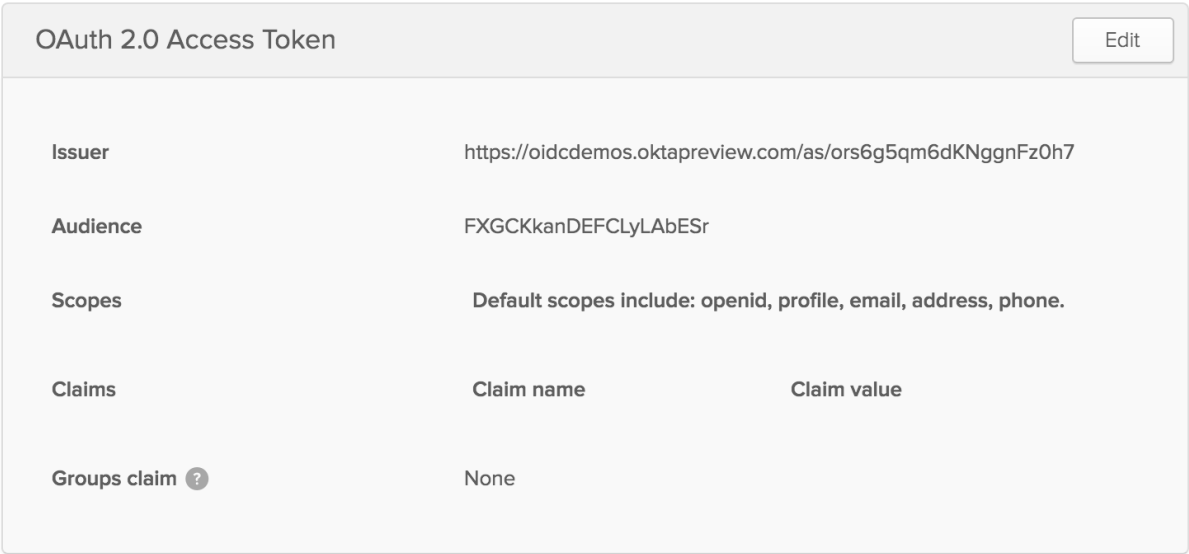
1. Select the **People** tab of your Okta OpenID Connect app and press the **Assign to People** button. Assign your test user to the application by pressing the Assign button next to your test user(s).
2. A new window opens that allows you to configure the application user profile (with default values coming from Okta's Universal Directory profile). In the screenshot, we have customized the Name, Nickname, Given Name and Family Name application user profile attributes.



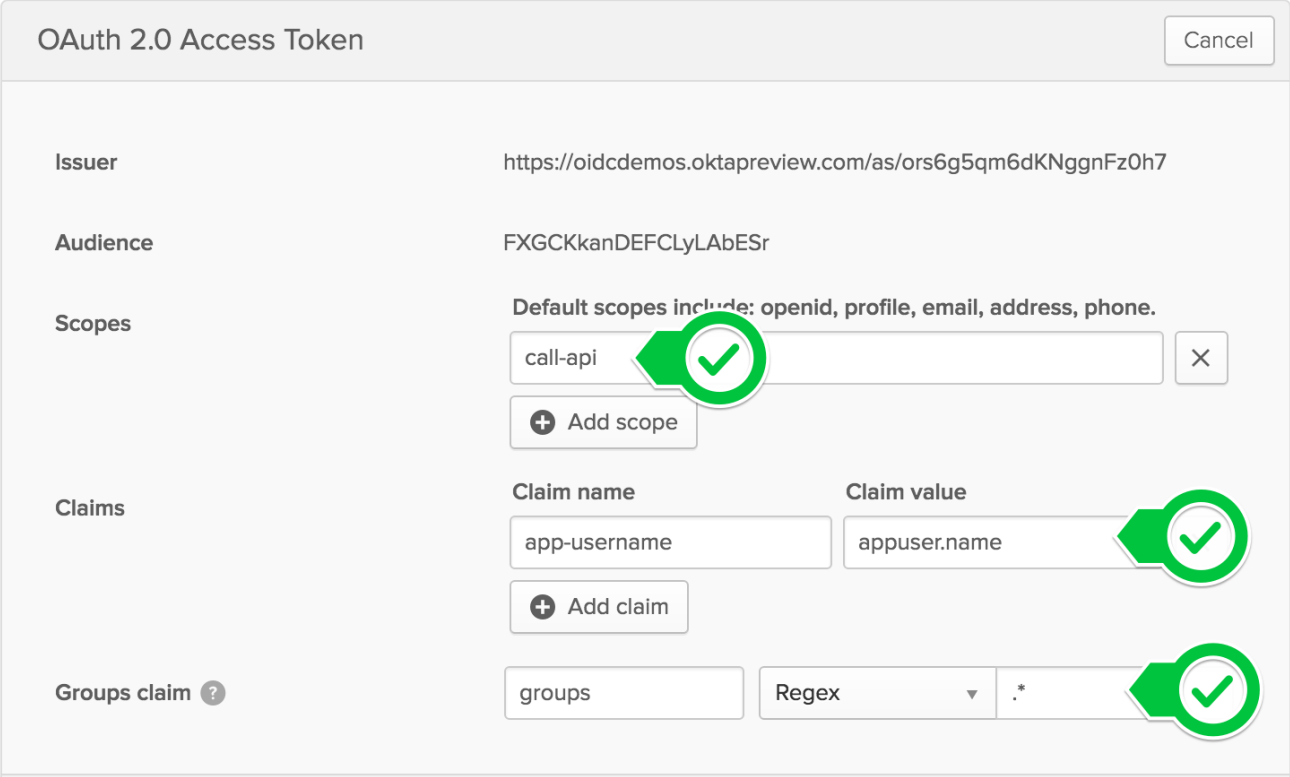
3. Press **Save and Go Back**, then **Done**.
4. (Optional) Select the **Authorization Server** tab and press the **Edit** button if you want to use the groups scope in the ID Token.
5. Select the Regex value in the Groups claim dropdown list and enter ".*" to include all the user's groups in the groups claim of the ID Token.



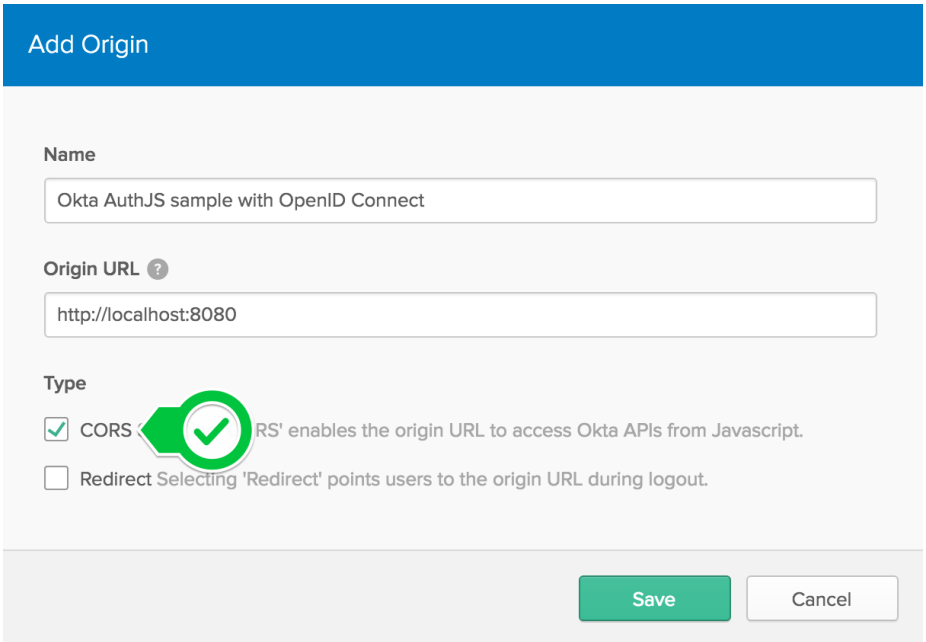
6. If you have access to the **API Access Management** SKU, the OAuth 2.0 Access Token section appears below the OpenID Connection ID Token section:



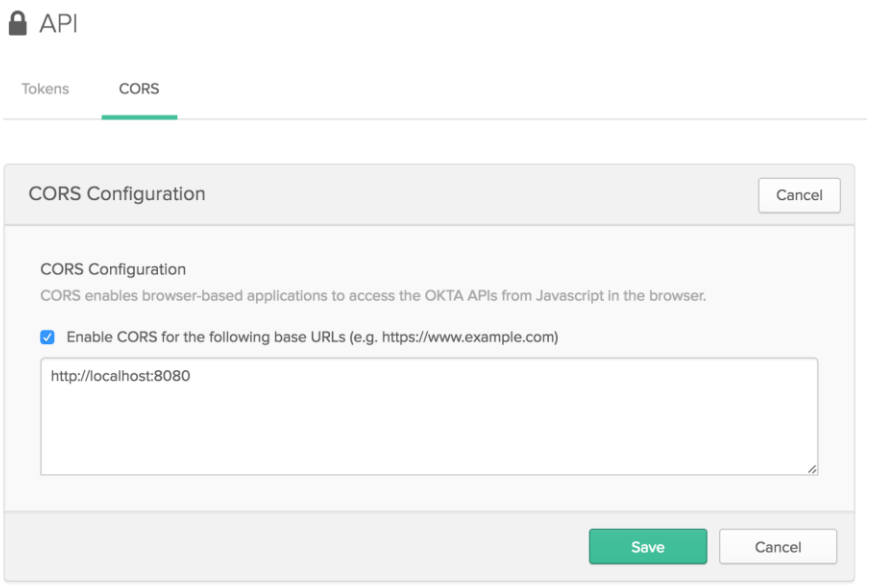
7. Press the **Save** button. Press the Edit button in that section, and add the **call-api** custom scope and **app-username** custom claim as configured below and press the **Save** button.



8. We will also need the Issuer value in that section to configure our OAuth Resource Server . But for now, select the **Security** ➔ **API** menu and select the **Trusted Origins** or **CORS** tab (depending on your Okta organization). If you see the Trusted Origins tab, press the **Add Origin** button, enter a name and the local url of your application in the list, such as http://localhost:8080 and select the **CORS** checkbox:



9. If you see the CORS tab, press the **Edit** button, check the **Enable CORS for the following based URLs** checkbox and enter the local url of your application in the list, such as http://localhost:8080:

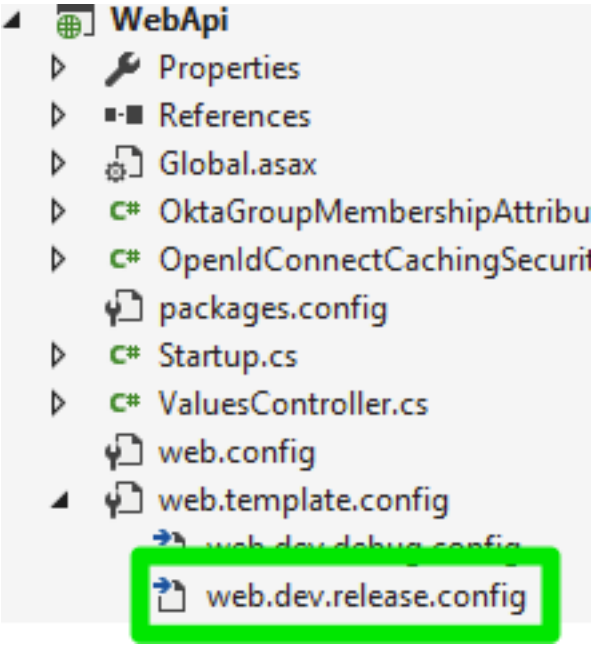


10. Press the **Save** button.

Configuring the OAuth Resource Server

If you are running the sample on Windows, you can configure the resource server the SPA client app can call. By default, the client app with use the Access Token to call the resource server API but you can change that default behavior to use the ID Token (for instance, in case you don't have access to Okta's API Access Management product SKU).

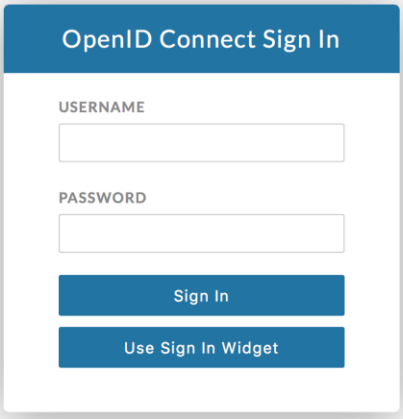
1. Launch the **Okta.Samples.OAuth.AuthJS.sln** solution file in Visual Studio 2015 and take a look at the **web.dev.release.config** file in the **WebApi** project.



2. Make sure you properly configure the **okta:OIDC_Issuer**, **okta:ClientId** and **okta:RequiredGroupMemberships** values in **web.dev.debug.config**. You can also optionally update the **okta:OAuth_Issuer** and **okta:IDorAccessToken** parameters if you plan to use the Access Token (instead of the ID Token) to call the resource server API. The meaning of each of these parameters is available in the **web.dev.release.config** file.
3. If you want to test the **call-api** custom scope (cf. OAuth 2.0 Access Token section), edit the **config.js** file is the **JavaScriptClient/js** folder and change the scope value to include **call-api** (or any custom scope you may have configured). You also must set **callApiWithAT** to true in ***config.js*** as well as set the **okta:IDorAccessToken** to **"access"** in the WebApi **web.dev.debug.config** file.
4. If you run into any issue while running the sample, please look at the Chrome Developers Tool console for any relevant message. The WebApi project should also output any error that may occur when validating the ID or Access Token in the OAuth OWIN middleware.
5. You should now be good to go to run and test this sample OpenID Connect/OAuth sample app!

Testing the sample with your Okta OpenID Connect application

1. Navigate to the *JavaScriptClient* folder inside your repository root folder and run either **`server.bat`** (if you're on Windows) or **`./server.sh`** from Terminal (if you're on Mac OS). This will start the Python Simple HTTP Server on port 8080. If you have configured your app to run on another port, edit the **`server.py`** file to match your local settings.
2. Navigate to http://localhost:8080. The following page should appear:



3. Enter the credentials of your test user and press the **Sign In** button:
4. A page similar to the following screenshot appears:


```
{
  "sub": "00u6yjbti4MYAVDkA0h7",
  "name": "Bob Sinclar",
  "email": "bob.sinclar@okta.com",
  "ver": 1,
  "iss": "https://example.oktapreview.com",
  "aud": "ViczvMucBWT14qg3lAM1",
  "iat": 1470177773,
  "exp": 1470181373,
  "jti": "ID.yNI8sD7lUh5kK6muTFH6o8InuAm_grJ1fxgz8sVLLvw",
  "amr": [
    "pwd"
  ],
  "idp": "00o5ivsvqlJSJVBme0h7",
  "nonce": "NjPM0C7NXa25eZZYVcQJ0O1fluazORXVrndRDvhqthHKSY0IxuH04BaCp59320ro",
  "preferred_username": "bob@example.com",
  "auth_time": 1470177772,
  "at_hash": "U0hquMzgbJ0pGFFu4sXSZg"
}
```


Notice that the ID token only contains the openid and email scopes. This is because we make a request to authorize endpoint by requesting response_type_id_token token in which case the ID token is by design kept small (cf. http://developer.okta.com/docs/api/resources/oidc.html#scope-dependent-claims-not-always-returned):

The client can also optionally request an Access Token along with the ID Token. In this case, in order to keep the size of the ID Token small, the ID Token body does not contain all the scope dependent claims. Instead, the ID token contains the name and preferred_username claims if the profile scope was requested and email claim if the email scope was requested.

If you want to get a full ID Token (including the user's group, press the **Renew ID Token** button:


```
{
  "sub": "00u6yjbti4MYAVDkA0h7",
  "name": "Bob Sinclar",
  "profile": "https://en.wikipedia.org/wiki/Bob_Sinclar",
  "locale": "US",
  "email": "bob.sinclar@okta.com",
  "picture": "https://upload.wikimedia.org/wikipedia/commons/thumb/1/18/Bob_Sinclar_2011.j
  "website": "http://www.bobsinclar.com/",
  "gender": "Male",
  "birthdate": "10 May 1969",
  "ver": 1,
  "iss": "https://example.oktapreview.com",
  "aud": "ViczvMucBWT14qg3lAM1",
  "iat": 1470177544,
  "exp": 1470181144,
  "jti": "ID.-hT-LunOQf0nM2Rah67l4MTMlHuTaQMBTCeqv7zCLb8",
  "amr": [
    "pwd"
  ],
  "idp": "00o5ivsvqlJSJVBme0h7",
  "nonce": "s0wW3bbM3popKWIkoQxva0zYlyG0eCcGIBW55y1aJacVghbkNp1xdUMwoIQ4gwEb",
  "nickname": "Chris The French Kiss",
  "preferred_username": "bob@example.com",
  "given_name": "Christophe",
  "middle_name": "Snoop ",
  "family_name": "Le Friant",
  "zoneinfo": "America/Los_Angeles",
  "updated_at": 1469833354,
  "email_verified": true,
  "phone_number": "4155833872",
  "auth_time": 1470176922,
  "address": {
    "street_address": "301 Brannan St.",
    "locality": "San Francisco",
    "region": "CA",
    "postal_code": "94107",
    "country": "US",
    "formatted": "301 Brannan St (formatted)"
  },
  "groups": [
    "Finance",
    "Marketing",
    "Everyone"
  ]
}
```

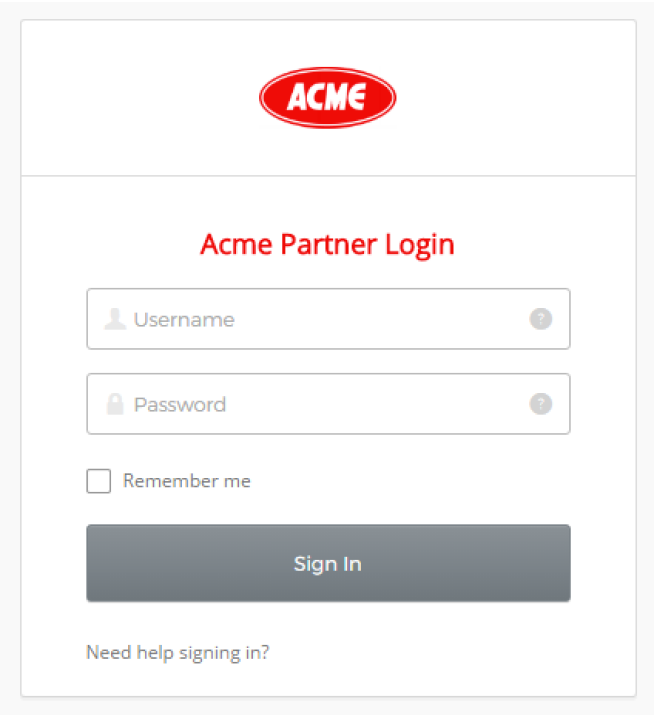

5. Notice that, as expected, the decoded ID Token displayed here leverages information from the user's app profile, not the Okta main profile (also referred to as the UD profile – UD stands for "Universal Directory"). Note also that the ID Token displays scope-independent and scope-dependent claims as documented at http://developer.okta.com/docs/api/resources/oidc.html#claims-in-the-payload-section (the scope parameters sent to Okta' Authorization end point is defined in the **`/JavaScriptClient/js/config.js`** file)
6. Press the **Renew ID Token** button again and notice that the **iat** (Issued At) and **exp** (Expiration Time) claims get slightly incremented by a few seconds. Note that the process we use to renew the ID Token does not

use any refresh token (which are not available in the OpenID Connect Implicit Flow), but calls the /oauth2/v1/token endpoint to request a new ID Token using the current Okta user session. If the Okta user session has expired, the user will be prompted to sign in with Okta again.

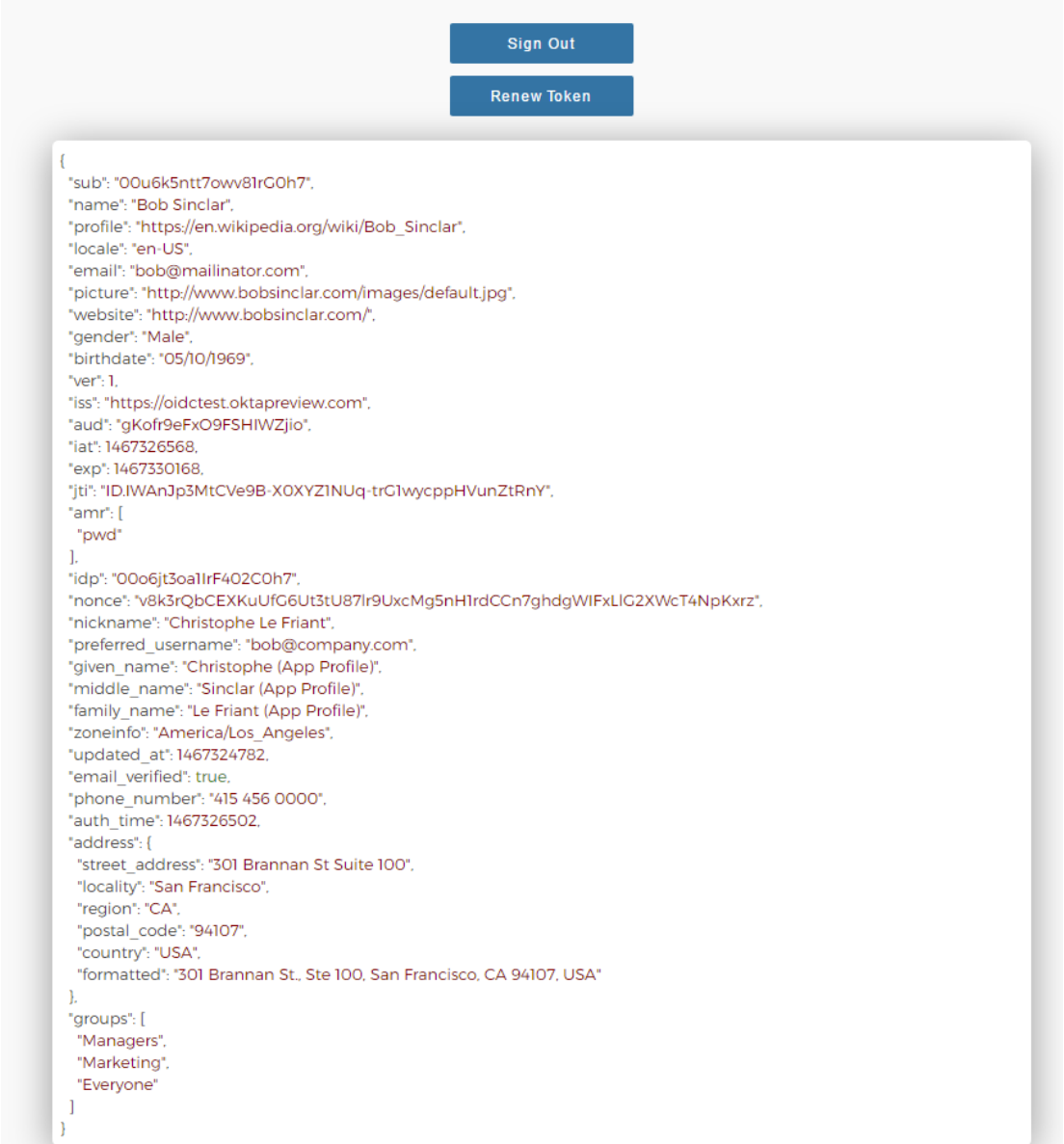
7. If you are running the sample on Windows and launched the **Okta.Samples.OAuth.AuthJS** solution in Visual Studio 2015, run the solution from there: this will launch the ASP.NET Resource Server and you can press the **Call Resource Server (API)** button to call the **/protected** endpoint of that resource server. If you have the right setup and configuration, you should get the following message:

All good. You only get this message if you are authenticated (as) AND you belong to either the Marketing or Finance group(s).

8. Pres the **Sign Out** button and then the **Use Sign In Widget** button. The following page appears:



9. Again, enter the credentials of your test user and press the **Sign In** button:
10. A page similar to the following screenshot appears:



11. Press the **Renew Token** button to renew the user's ID Token, with a process similar to the one used in the Authentication JavaScript SDK example above.
12. Press the **Sign Out** button to return to the page displaying the **Okta Sign In Widget**.