WSSO TO Okta

Primary Option

Table Of Contents

[Introduction 2](#_Toc92816998)

[Step 1: Angular Setup in Local 2](#_Toc92816999)

[Local 2](#_Toc92817000)

[In VDI 3](#_Toc92817001)

[Step 2: Install Okta Dependencies 4](#_Toc92817002)

[Step 3: Update Environment.ts 4](#_Toc92817003)

[Step 4: Update app.module.ts 5](#_Toc92817004)

[Step 5: Update app-routing.module.ts 6](#_Toc92817005)

[Step 6: LoaderInterceptorService.ts 6](#_Toc92817006)

[Step 7: Update Logout URL 7](#_Toc92817007)

[Step 8: Server side (Spring Boot) 7](#_Toc92817008)

[App to App 7](#_Toc92817009)

[UI 7](#_Toc92817010)

[Step 9: Update Spring Boot Okta Dependencies 8](#_Toc92817011)

[build.gradle 8](#_Toc92817012)

[pom.xml 8](#_Toc92817013)

[Step 10: Application.properties 8](#_Toc92817014)

[App to App 8](#_Toc92817015)

[UI 8](#_Toc92817016)

[References 8](#_Toc92817017)

## Introduction

This document captures all the generic code changes that needs to be made  
when WSSO is converted to Okta using Primary Option.

**Note: ESC to Okta Authorization for which documentation will be updated later.**

## Step 1: Angular Setup in Local

Software’s required to setup Angular in Local

1. Node JS -> Latest version preferred
2. Install Git
3. Install VS code
4. Python 2.7

Get Admin rights from the Mphasis IT team, Login as Admin and Install it.

### Local

1. Open command prompt and type npm config edit

Text

Description automatically generated

.npmrc file would open, Add the below configurations

prefix=C:\Users\{your profile name}\AppData\Roaming\npm

registry=https://registry.npmjs.org/

strict-ssl=false

open environment variables and update the below path  
C:\Users\{your profile name}\AppData\Roaming\npm

After adding them, File->Save & File->Exit. Reopen the file using the same command. Your file   
should be like below.

Text

Description automatically generated

1. Go to the Path where Package.json is present and open gitbash
2. Run the below command to Install Angular

Install the global version first - npm install -g @angular/cli

Install the npm install @angular/cli (use the version mentioned in package.json)

and do npm install again

1. If any dependency version mismatch,

npm install {@dependency version} needs to be installed.

1. Start the server  
   ng serve -o --port 8080

### In VDI

1. Open command prompt and type npm config edit

Text

Description automatically generated

.npmrc file would open, Add the below configurations

strict-ssl=false

https-proxy=http://{LDAP ID}:{LDAP PWD}@internet.proxy.fedex.com:3150/

proxy=http:// {LDAP ID}:{LDAP PWD}@internet.proxy.fedex.com:3150/

registry=https://nexus.prod.cloud.fedex.com:8443/nexus/repository/registry-npm-fedex/

After adding them, File->Save & File->Exit. Reopen the file using the same command. Your file   
should be like below.

Graphical user interface, text, application, email

Description automatically generated

Follow the rest of the steps From **Local setup Step 2**

Step 2: Onboard your application into Okta  
  
Migration team will be onboarding the EAI in Okta

## Step 2: Install Okta Dependencies

Install the Okta dependencies NPM packages as below  
npm i @okta/okta-angular@3.0.1

## Step 3: Update **Environment.ts**

Location : src->Environment

environment.prod.ts

environment.development.ts

environment.release.ts

environment.staging.ts

environment.ts

The below values (Client-ID, Issuer URI) will be given by the Migration team when they onboard the EAI into Okta

All these below values will vary based on the environment.

* clientID
* issuer URI
* Login redirect URI

**Note: Login redirect URI will be provided by the remediation team to the Migration team**

**For Unit testing use the below configuration.**

*oktaConfig: {*

*issuerUri: "https://purpleid-stage.oktapreview.com/oauth2/default", //environment uri*

*clientId: “0oaobumrdb5SJjd6G0h7" // {yourClientId}",*

*redirectUri: "http://localhost:8080/authorization-code/callback", // environment uri*

*pkce: true,*

*tokenManager: {*

*storage: "sessionStorage",*

*expireEarlySeconds: 300,*

*autoRenew: true,*

*},*

*testing: {*

*disableHttpsCheck: false,*

*},*

*}*

## Step 4: Update app.module.ts

Open the app.module.ts file and update the modules which include your project components.

In Addition to the existing code, below code changes have to be implemented

*import { OktaAuthModule, OKTA\_CONFIG } from "@okta/okta-angular";*

*@NgModule({*

*imports: [*

*... {Existing code}*

*OktaAuthModule*

*...*

*]*

*providers: [*

*{existing code}*

*{ provide: OKTA\_CONFIG, useValue: environment.oktaConfig },*

*{existing code}*

*],*

## Step 5: Update app-routing.module.ts

In the app-routing module, guard the route with the Angular route guard.

*import { OktaAuthGuard, OktaCallbackComponent } from "@okta/okta-angular";*

*Under routes :*

*{*

*path: "",  
 pathMatch: "full",*

*redirectTo: "/login",*

*canActivate: [OktaAuthGuard]*

*},*

*{*

*path: "authorization-code/callback",*

*component: OktaCallbackComponent,*

*pathMatch: "full",*

*},*

*{*

*... {Existing code}*

*canActivate: [OktaAuthGuard],*

*}*

## Step 6: **LoaderInterceptorService.ts**

**To send the token to the server side, changing in the interceptor service is enough.**

*import { OktaAuthService } from '@okta/okta-angular';*

*// below class name varies based on the application, look for any interceptor in your application*

*export class LoaderInterceptorService {*

*{existing code}*

*constructor(*

*{existing code},*

*,private oktaAuth: OktaAuthService) {*

*{existing code}*

*}*

*intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {*

*const accessToken = this.oktaAuth.getAccessToken();*

*// add the token in below header,*

*setHeaders: {*

*'Authorization': 'Bearer ' + accessToken*

*},*

*}*

***Use the below for server side service****.*

## Step 7: **Update Logout URL**

**It is applicable wherever logout call would happen and it can be any component.**

*import { OktaAuthService } from '@okta/okta-angular';*

*export class LogoutComponent {*

*isOktaAuthenticated : boolean;*

*{existing code}*

*constructor({existing code},*

*,private oktaAuth: OktaAuthService) {*

*{existing code}*

*this.oktaAuth.$authenticationState.subscribe(isAuthenticated => this.isOktaAuthenticated = isAuthenticated)*

*}*

***lopgout(){***

***if(this.isOktaAuthenticated){ .***

***{existing code}***

***window.location.href = "https://purpleid-stage.oktapreview.com/login/signout"; }***

***}***

## Step 8: Server side (Spring Boot)

Find out the Custom Websecurity Adapter which extends WebSecurityConfigurerAdapter  
update the below config method.

*@Configuration*

*@EnableWebSecurity*

*public class SecurityConfig extends WebSecurityConfigurerAdapter {*

*@Value("${application.okta.scope}")*

*private String scope;*

### App to App

*@Override*

*protected void configure(HttpSecurity http) throws Exception {*

*http.authorizeRequests()*

*.mvcMatchers("/something/something/\*\*").hasAuthority(scope)*

*.and()*

*.oauth2ResourceServer()*

*.jwt();*

*}*

### UI

*@Override*

*protected void configure(HttpSecurity http) throws Exception {*

*httpSecurity.csrf().disable().antMatcher("/something/\*\*").permitAll()*

*.and()*

*.authorizeRequests()*

*.antMatchers("/something/\*\*").authenticated() //open to authenticated users*

*http.cors();*

*Okta.configureResourceServer401ResponseBody(http);*

*}*

*}*

## Step 9: Update Spring Boot Okta Dependencies

### build.gradle

*dependencies {  
 implementation 'com.okta.spring:okta-spring-boot-starter'  
 implementation 'org.springframework.security:spring-security-oauth2-resource-server'  
 implementation 'org.springframework.security:spring-security-oauth2-jose'  
}*

### pom.xml

*<dependency>*

*<groupId>com.okta.spring</groupId>*

*<artifactId>okta-spring-boot-starter</artifactId>*

*<version>1.3.0</version>*

*</dependency>*

## Step 10: Application.properties

To Communicate Spring Boot with Okta, Configure Okta oauth2 properties

### App to App

*spring:*

*security:*

*oauth2:*

*resourceserver:*

*jwt:*

*issuer-uri: https://purpleid-stage.oktapreview.com/oauth2/default*

### UI

*okta:*

*oauth2:*

*issuer: https://purpleid-stage.oktapreview.com/oauth2/default*

*client-id: {yourClientId}*

*client-secret: {yourClientSecret}*

*audience: api://default*

## References

1. <https://medium.com/@raghavendra.pes/securing-angular-spring-boot-application-with-okta-671e983e5b6>
2. <https://www.prishusoft.com/blog/implement-okta-authentication-in-angular.html>
3. [GitHub - okta/okta-angular: Angular SDK for Okta's OIDC flow](https://github.com/okta/okta-angular#:~:text=the%20config%20object.-,OktaCallbackComponent,them%2C%20then%20redirects%20to%20%2F%20.)
4. <https://gitlab.prod.fedex.com/440947/Spring-Boot-Resource-Server-Okta-Demo>
5. <https://gitlab.prod.fedex.com/440947/Angular-Okta-PKCE-Demo>