Integración con PAEC desde una aplicación Java Script

PAEC viene con una biblioteca de JavaScript del lado cliente que se puede usar para proteger las aplicaciones HTML5 / JavaScript. El adaptador de JavaScript tiene compatibilidad incorporada para las aplicaciones de Cordova.

La biblioteca se puede obtener directamente del servidor PAEC en /auth/js/keycloak.js y también se distribuye como un archivo ZIP.

Una práctica recomendada es cargar el adaptador de JavaScript directamente desde el servidor de PAEC, ya que se actualizará automáticamente cuando actualice el servidor. Si copia el adaptador en su aplicación web, asegúrese de actualizar el adaptador solo después de actualizar el servidor.

Una cosa importante a tener en cuenta sobre el uso de aplicaciones del lado del cliente es que el cliente debe ser un cliente público, ya que no hay una forma segura de almacenar las credenciales del cliente en una aplicación del lado del cliente. Esto hace que sea muy importante asegurarse de que los URI de redireccionamiento que ha configurado para el cliente sean correctos y lo más específicos posible.

Para usar el adaptador de JavaScript primero debe crear un cliente para su aplicación en la Consola de administración de PAEC. Asegúrese de que el `public` esté seleccionado para `Access Type`.

También deben configurarse los URI de redirección válidos y los orígenes web válidos. Sea tan específico como sea posible, ya que si no lo hace puede provocar una vulnerabilidad de seguridad.

Una vez que se crea el cliente, haga clic en la pestaña de `Installation`, seleccione `Keycloak OIDC JSON para Format Option` y luego haga clic en Download. El archivo `keycloak.json` descargado debe estar alojado en su servidor web en la misma ubicación que sus páginas HTML.

Alternativamente, puede omitir el archivo de configuración y configurar manualmente el adaptador.

El siguiente ejemplo muestra cómo inicializar el adaptador de JavaScript:

```
<head>
      <script src="keycloak.js"></script>
      <script>
      var keycloak = Keycloak();
      keycloak.init().success(function(authenticated) {
            alert(authenticated ? 'authenticated' : 'not authenticated');
            }).error(function() {
            alert('failed to initialize');
      });

      </script>
```

```
</head>
```

Si el archivo `keycloak.json` se encuentra en otra ubicación, puede especificarla:

```
var keycloak = Keycloak('http://tst.autenticar.gob.ar/tad/keycloak.json'));
```

Alternativamente, puede pasar un objeto JavaScript con la configuración requerida:

```
var keycloak = Keycloak({ url: 'http://keycloak-server/auth', realm: 'myrealm',
clientId: 'myapp' });
```

De forma predeterminada, para autenticar, debe llamar a la función de login. Sin embargo, hay dos opciones disponibles para que el adaptador se autentique automáticamente. Puede pasar `login-required` o `check-sso` a la función init. `login-required` autenticará al cliente si el usuario ya ha iniciado sesión en PAEC o mostrará la página de inicio de sesión. `check-sso` solo autenticará al cliente si el usuario ya inició sesión, si el usuario no inició sesión, el navegador será redirigido a la aplicación y permanecerá sin autenticar.

Para habilitar login-required se debe configurar onLoad a login-required como parámetro al método init:

```
keycloak.init({ onLoad: 'login-required' })
```

Una vez autenticado el usuario, la aplicación puede realizar solicitudes a los servicios RESTful expuestos por PAEC incluyendo el token de acceso en el header `Authorization`. Por ejemplo:

```
var loadData = function () {
      document.getElementById('username').innerText = keycloak.subject;

      var url = 'http://localhost:8080/restful-service';

      var req = new XMLHttpRequest(); req.open('GET', url, true);
      req.setRequestHeader('Accept', 'application/json');
      req.setRequestHeader('Authorization', 'Bearer ' + keycloak.token);
      req.onreadystatechange = function () {
            if (req.readyState == 4) {
                  if (req.status == 200) {
                        alert('Success'); }
                  else if (req.status == 403) {
                      alert('Forbidden');
                  }
            }
      }
      req.send();
};
```

Una cosa a tener en cuenta es que el token de acceso de forma predeterminada tiene un tiempo de vida corto, por lo que es posible que tenga que actualizar el token de acceso antes de enviar la solicitud. Puede hacerlo mediante el método updateToken. El método updateToken devuelve un objeto de

promise que hace que sea fácil invocar el servicio solo si el token se actualizó correctamente y, por ejemplo, muestra un error al usuario si no lo fue. Por ejemplo:

```
keycloak.updateToken(30).success(function() {
      loadData(); }).error(function() {
          alert('Failed to refresh token');
);
```

Session Status iframe

De forma predeterminada, el adaptador de JavaScript crea un iframe oculto que se utiliza para detectar si se ha producido un Single-Sign Out. Esto no requiere ningún tráfico de red, sino que el estado se recupera al mirar una cookie de estado especial. Esta función se puede deshabilitar configurando `checkLoginIframe: false`: en las opciones pasadas al método `init`.

No debe confiar en mirar esta cookie directamente. Su formato puede cambiar y también está asociado con la URL del servidor PAEC, no con su aplicación.

Flujos Implicit e Hybrid

De forma predeterminada, el adaptador de JavaScript usa el flujo Authorization Code.

Con este flujo, el servidor PAEC devuelve un código de autorización, no un token de autenticación, a la aplicación. El adaptador de JavaScript intercambia el code para un token de acceso y un token de actualización después de que el navegador se redirige nuevamente a la aplicación.

PAEC también soporta el flujo Implicit en el que se envía un token de acceso inmediatamente después de la autenticación correcta con PAEC. Esto puede tener un mejor rendimiento que el flujo estándar, ya que no hay una solicitud adicional para intercambiar el código por tokens, pero tiene implicaciones cuando expira el token de acceso.

Sin embargo, enviar el token de acceso en el fragmento de URL puede ser una vulnerabilidad de seguridad. Por ejemplo, el token podría filtrarse a través de los registros del servidor web o del historial del navegador.

Para habilitar el flujo implicit, debe habilitar el indicador `Implicit Flow` Enabled para el cliente en la consola de administración de PAE. También debe pasar el flujo de parámetros con el valor implícito al método init:

```
keycloak.init({ flow: 'implicit' })
```

Una cosa a tener en cuenta es que solo se proporciona un token de acceso y no hay token de actualización. Esto significa que una vez que el token de acceso ha caducado, la aplicación tiene que volver a redireccionar a PAEC para obtener un nuevo token de acceso.

PAEC también soporta el flow Hybrid.

Esto requiere que el cliente tenga activados los indicadores `Standard Flow Enabled` e `Implicit Flow` en la consola de administración. El servidor PAEC enviará tanto el código como los tokens a su aplicación. El token de acceso se puede usar inmediatamente mientras se pueda intercambiar el código para acceder y actualizar tokens. Similar al flujo implícito, el flujo híbrido es bueno para el rendimiento porque el token de acceso está disponible de inmediato. Pero, el token todavía se envía en la URL, y la vulnerabilidad de seguridad mencionada anteriormente aún puede aplicarse.

Una ventaja en el flujo híbrido es que el token de actualización está disponible para la aplicación.
Para el flujo híbrido, se debe pasar el parámetro flow con el valor hybrid al método init:

```
keycloak.init({ flow: 'hybrid' })
```

Navegadores antiguos

El adaptador de JavaScript depende de Base64 (window.btoa y window.atob) y HTML5 History API. Si se necesita soportar navegadores que no tienen estos disponibles (por ejemplo, IE9), necesita agregar polyfillers.
Ejemplos de librerías polyfill:

https://github.com/davidchambers/Base64.js
https://github.com/devote/HTML5-History-API

Referencia del adaptador JavaScript

Constructor

```
new Keycloak();
new Keycloak('http://localhost/keycloak.json');
new Keycloak({ url: 'http://localhost/auth', realm: 'mmodern', clientId: 'tad' });
```

Propiedades

authenticated	Es `true` si el usuario está autenticado, `false` si no.
token	El token codificado en base 64 que puede ser enviado en el header `Authorization` en requests a servicios.
tokenParsed	El token parseado a un objeto JavaScript.
subject	El Id del usuario.
idToken	El ID Token codificado en base 64.
idTokenParsed	El ID Token parseado a un objeto JavaScript.
realmAccess	Los roles de realm asociados al token.
resourceAccess	Los roles de resource asociados al token.
refreshToken	El token de refresco codificado en base 64, que puede usarse para obtener un nuevo token de acceso.
refreshTokenParsed	El token de refresco parseado a un objeto JavaScript.
timeSkew	La diferencia de tiempo estimada entre la hora del browser y el servidor PAEC en segundos. Este valor es solamente un estimado, pero lo suficientemente preciso para determinar si el token está expirado o no.
responseMode	Modo de respuesta hecho al init (el valor por defecto es "fragment").
flow	Flujo pasado en init.
responseType	Tipo de respuesta enviada a PAEC con los requests de login. Esto es determinado por el valor del flujo usado durante la inicialización, pero se puede reemplazar seteando este valor.

Librerías OAuth 2.0 y OpenID Connect

Adicionalmente a los adaptadores nativos de PAEC, se pueden integrar y utilizar librería de terceros que cumpla con los estándares OAuth 2.0 y OpenID Connect, un ejemplo de esto es la librería https://appauth.io

AppAuth que es un SDK de cliente para aplicaciones nativas que permite autenticar y autorizar a los usuarios finales que utilizan OAuth 2.0 y OpenID Connect, está disponible para entornos JavaScript Nativo, Android, iOS y MacOS, implementa prácticas de seguridad y usabilidad modernas para la autenticación y la autorización de aplicaciones, mapeando las solicitudes y respuestas de las especificaciones utilizando el lenguaje nativo de las implementaciones y mejores prácticas establecidas en RFC 8252: OAuth 2.0 para aplicaciones nativas.