# Env

* 74.0.3729.131, 64 bit as well as current Edge
* MSAL 1.0.0

# Code

The code i staken from the msal js samples.

<html>

<head>

 <title>Calling a Web API as a user authenticated with Msal.js app</title>

 <style>

 body {

 font-family: 'Arial Narrow', Arial, sans-serif

 }

 button {

 background-color: blue;

 color: whitesmoke;

 border-style: solid;

 border-width: 1px;

 padding: 10px;

 border-radius: 3px;

 }

 button:hover {

 background-color: #8181ee;

 }

 hr {

 margin-top: 20px;

 }

 .hidden {

 display: none;

 visibility: hidden

 }

 .visible {

 visibility: visible

 }

 .response {

 border-style: none;

 border-width: thin;

 background-color: rgb(255, 228, 198);

 padding: 0px;

 border-radius: 2px;

 }

 .blockTitle {

 font-size: 130%;

 font-weight: bold;

 margin-right: 25px;

 }

 </style>

</head>

<body style="background-color: lightblue">

 <!-- bluebird only needed if this page needs to run on Internet Explorer -->

 <script src="https://cdnjs.cloudflare.com/ajax/libs/bluebird/3.3.4/bluebird.min.js" class="pre"></script>

 <script src="https://secure.aadcdn.microsoftonline-p.com/lib/1.0.0/js/msal.js"></script>

 <script src="https://code.jquery.com/jquery-3.2.1.min.js" class="pre"></script>

 <h2>Test Authentication with Azure AD B2C</h2>

 <div>

 <div id="label">Not logged in</div>

 <br />

 <span class="blockTitle">Azure AD B2C Authentication</span>

 <button id="auth" onclick="login()">Login with azure ADB2C</button>

 <button id="callApiButton" class="hidden" onclick="getTokenSilently()">Test get token silently</button>

 <hr>

 <!-- Credentials - User/PW - Not used at the moment

 <span class="blockTitle">Authentication with username/pw</span>

 <button id="loginWithCredentialsButton" onclick="loginWithCredentials()">

 Login with username/pw (user@test.com/password)

 </button>

 <button id="callApiWithSessionButton" class="hidden" onclick="callApiWithLoggedInSession()">

 Call Web API with Session User

 </button>

 <hr>

 -->

 <span class="blockTitle">Logout</span>

 <button id="logoutButton" class="hidden" onclick="logout()">Logout</button>

 <hr>

 </div>

 <pre class="response"></pre>

 <script>

 console.info(`Start script: history.length: %s`, history.length);

 const b2bScopesAuthTest = ["https://tgwdsb2c.onmicrosoft.com/xxx/demo.read"];

 const userFlowPolicy = "B2C\_1\_Test";

 const clientIDTGWAuthTest = "the apps my client id";

 // azure B2C config.

 let applicationConfig = {

 clientID: clientIDTGWAuthTest,

 authority: `https://tgwdsb2c.b2clogin.com/tgwdsb2c.onmicrosoft.com/${userFlowPolicy}`,

 // OLD!!! authority: "https://login.microsoftonline.com/tfp/tgwdsb2c.onmicrosoft.com/B2C\_1\_Test",

 b2cScopes: b2bScopesAuthTest,

 webApi: 'http://localhost:5000/api/test',

 loginApi: 'http://localhost:5000/login',

 logoutApi: 'http://localhost:5000/api/logout',

 };

 let curAuthStrategy;

 let authenticated = false;

 let clientApplication;

 let AuthStrategy = {};

 AuthStrategy[AuthStrategy.PassportToken = 1] = "PassportToken";

 AuthStrategy[AuthStrategy.Credentials = 2] = "Credentials";

 let mockUser = {

 email: "user@test.com",

 password: "password"

 };

 let loggedInUser = undefined;

 window.onload = doIt;

 /\*\*

 \* Start test.

 \*/

 async function doIt() {

 updateUI();

 console.info(`doIt: appConfig: %O, history.length: %s`, applicationConfig, history.length);

 clientApplication = new Msal.UserAgentApplication(

 {

 auth: {

 clientId: applicationConfig.clientID,

 authority: applicationConfig.authority,

 validateAuthority: false

 },

 cache: {

 cacheLocation: "localStorage",

 storeAuthStateInCookie: true

 }

 }

 );

 /\*\*

 \* Try to get access token silently without interaction.

 \* => Try to automatically login when app starts.

 \*/

 // let account = clientApplication.getAccount();

 // if (account) {

 // console.info("\*\*\*\*\*\*\* Found account in client app.");

 // try {

 // let tokenRequest = { scopes: applicationConfig.b2cScopes };

 // let accessToken = await clientApplication.acquireTokenSilent(tokenRequest);

 // authenticated = true;

 // curAuthStrategy = AuthStrategy.PassportToken;

 // updateUI()

 // } catch (e) {

 // logMessage("Error acquiring the token silently:\n" + e);

 // }

 // }

 }

 /\* azure AD B2C.

 ----------------------------------------------------------------------------------------\*/

 /\*\*

 \* Login to azure AD.

 \*/

 function login() {

 let loginRequest = { scopes: applicationConfig.b2cScopes };

 clientApplication.loginPopup(loginRequest)

 .then(function (authResponse) {

 console.log(`After loginPopup, authResponse == %O`, authResponse);

 console.info(`login: history.length: %s`, history.length);

 if (authResponse.accessToken) {

 curAuthStrategy = AuthStrategy.PassportToken;

 authenticated = true;

 console.info(`login: history.length: %s`, history.length);

 updateUI();

 } else {

 let tokenRequest = { scopes: applicationConfig.b2cScopes };

 clientApplication.acquireTokenSilent(tokenRequest)

 .then(function (accessToken) {

 console.info("After login & acquireTokenSilent, token == %O", accessToken);

 console.info(`login: history.length: %s`, history.length);

 curAuthStrategy = AuthStrategy.PassportToken;

 authenticated = true;

 updateUI();

 }, function (error) {

 clientApplication.acquireTokenPopup(tokenRequest)

 .then(function (accessToken) {

 console.info("After login & acquireTokenPopup, token == %O", accessToken);

 console.info(`login: history.length: %s`, history.length);

 curAuthStrategy = AuthStrategy.PassportToken;

 authenticated = true;

 updateUI();

 }, function (error) {

 logMessage("Error acquiring the token silently:\n" + error);

 });

 })

 }

 }, function (error) {

 logMessage("Error during loginPopup:\n" + error);

 });

 }

 /\*\*

 \* Update the UI according to the state of login etc.

 \*/

 function updateUI() {

 console.info("Update UI");

 if (authenticated) {

 let userName = clientApplication && clientApplication.getAccount() ? clientApplication.getAccount().name : loggedInUser ? loggedInUser.email : "no logged in user";

 logMessage("User '" + userName + "' logged-in");

 let authButton = document.getElementById('auth');

 authButton.setAttribute("class", "hidden");

 // Not used: let loginWithCredButton = document.getElementById('loginWithCredentialsButton');

 // Not used: loginWithCredButton.setAttribute("class", "hidden");

 let logoutButton = document.getElementById('logoutButton');

 logoutButton.setAttribute("class", "visible");

 let label = document.getElementById('label');

 label.innerText = "Hello " + userName;

 if (curAuthStrategy === AuthStrategy.PassportToken) {

 let callWebApiButton = document.getElementById('callApiButton');

 callWebApiButton.setAttribute('class', 'visible');

 // Not used: let callApiWithSessionButton = document.getElementById('callApiWithSessionButton');

 // Not used: callApiWithSessionButton.setAttribute('class', 'hidden');

 } else {

 let callWebApiButton = document.getElementById('callApiButton');

 callWebApiButton.setAttribute('class', 'hidden');

 // Not used: let callApiWithSessionButton = document.getElementById('callApiWithSessionButton');

 // Not used: callApiWithSessionButton.setAttribute('class', 'visible');

 }

 } else {

 let logoutButton = document.getElementById('logoutButton');

 logoutButton.setAttribute("class", "hidden");

 let authButton = document.getElementById('auth');

 authButton.setAttribute("class", "visible");

 // Not used: let loginWithCredButton = document.getElementById('loginWithCredentialsButton');

 // Not used: authButton.setAttribute("class", "visible");

 }

 }

 /\*\*

 \* Test getting a token silently.

 \*/

 function getTokenSilently() {

 let tokenRequest = { scopes: applicationConfig.b2cScopes };

 clientApplication.acquireTokenSilent(tokenRequest).then(function (authResponse) {

 logMessage(`getTokenSilently: got token silently: ${authResponse.accessToken}`);

 console.info(`getTokenSilently: history.length: %s`, history.length);

 }, function (error) {

 clientApplication.acquireTokenPopup(tokenRequest).then(function (authResponse) {

 logMessage(`getTokenSilently: got token from popup: ${authResponse.accessToken}`);

 console.info(`getTokenSilently: history.length: %s`, history.length);

 }, function (error) {

 logMessage("Error acquiring the access token to call the Web api:\n" + error);

 });

 })

 }

Remainder of code not relevant…

# Register SPA in azure ADB2C portal



# Load app (index.html) in new tab



**Console log:**



# Click „Login…“



## Click „Sign in“

The login-window closes, the app (main window) gets reloaded in the JS **context of the hidden iframe**:



Console log after sign in:



# Click „Test get token silently“

Now, immediately after sign in (without any refresh/F5), click the button.

**Before the call:**



**During the call, the app reloads againin iframe:**



**After the call, we log the result:**





# Bugs/Questions

## Reload is not feasible in many situations

* The msal js samples are simple: A small web-page, which can be reloaded without any startup-cost.
* How can I prevent the app to fully initialize when ist loaded in the iframe?
	+ Reason: Our real SPAs are quite. They load ~30 webpack modules on start. So the reload in the iframe takes a lot of time. Besides, I don’t want to tamper with the state of my app. Strange things might happen if it is fully loaded in an iframe.
	I use the popups exactly for that reason: Don’t mess wit the app’s state.

## Why does „loginPopup“ cause reloads?

* Why does „loginPopup“ cause the app to reload? It opens a popup window and/or iframe which can communicate with the identity-provider (ADB2C).

## Why does „acquireTokenSilent“ cause a reload

* Why does „acquireTokenSilent“ cause a reload in an iframe, **even after login**? The login process should have cached the token. The strange thing: If I refresh (F5) the app and get the token silently once (the commented code at the beginning), **all subsequent calls to „acquireTokenSilent“ work immediately, without an iframe-reload**.
* Reloading the app can be very time-consuming in general

# Problems with our „real“ SPA

## App gets loaded in popup

We’ve used the same mechanism to authenticate in our SPAs. But even stranger things happen there: The app gets relaoded in the opopup window. Since the app takes a few seconds to load, one can watch it appear in the popup window.

## History state messed up

The history state get messed up. Our app uses history.push and pop. But the state after the login and „aquireTokenSilently“ contains 3-4 extra entries.

## Hash urls not useable

Our SPAs used hash URLs in ordert o support deep linking, even without server roundrips.

E.g.: „https://myapp.domain.com/#todos

The popup-redirect flow uses hashes, e.g. for error „interaction\_required“ or „state“.