FREE eBook

LEARNING
# azure-active-directory

#azure-active-directory

Table of Contents

About 1

Chapter 1: Getting started with azure-active-directory 2

Remarks 2

Examples 2

Installation or Setup 2

Azure Active Directory B2C - Setup 2

Customize the Azure AD B2C user interface 2

Chapter 2: Azure Active Directory B2C 4

Introduction 4

Examples 4

Azure AD B2C - Angularjs sample (Web and Mobile) app 4

Azure AD B2C 4

Web app - Hello.js 4

Mobile app - ADAL plugin 4

Decode JWT 4

1. Project set up: 4

2. AD set up: 5

3. AD settings: 5

4. Run this sample: 5

Web App: 5

Mobile App: 6

5. Customize the Azure AD B2C user interface 6

Implementation: 6

Web App 6

Mobile App 7

Related documents: 7

Chapter 3: Azure Active Directory B2C implementation using ADAL Cordova Plugin 8

Introduction 8

Examples 8

Azure Active Directory B2C implementation using ADAL Cordova Plugin 8

Azure AD B2C 8

Mobile app - ADAL plugin 8

Chapter 4: Azure Active Directory B2C implementation using Hello.js 11

Examples 11

Angularjs- Azure Active Directory B2Cusing Hello.js 11

Credits 14

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: azure-active-directory

It is an unofficial and free azure-active-directory ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official azure-active-directory.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with azure-active-directory

Remarks

This section provides an overview of what azure-active-directory is, and why a developer might want to use it.

It should also mention any large subjects within azure-active-directory, and link out to the related topics. Since the Documentation for azure-active-directory is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Detailed instructions on getting azure-active-directory set up or installed.

Azure Active Directory B2C - Setup

1. Create an Azure AD B2C Directory

   Note the ***Domain name***, it'll be used as the *tenantName*.

2. Register your application Follow the instructions to create an application and enable both Web App and Native client. Refer Register a web application and Register a mobile/native application

   Enter the *Reply URL* as http://localhost:8100 or any port from wher you'll be serving your app.

   In *Application Claims*, select Email Addresses too.

   Note the ***Application ID*** . It'll be used as the *clientId*.

3. Create a sign-up or sign-in policy

   Note the name of the policy. It'll be used as *policy*.

4. Create a password reset policy

   Note the name of the policy. It'll be used as the *password-reset-policy*

Customize the Azure AD B2C user interface

The Azure AD B2C login screen can be customized to suit our branding. Refer Customizing the UI

Refer https://github.com/NewtonJoshua/Azure-ADB2C-Angularjs-sample

In this sample we have two customized UI screens,

- AD B2C Sign in ansd Sign up page: adCustomPages/unified.html
- AD B2C Password reset page: adCustomPages/resetpassword.html

In adCustomPages/unified.html, at line 442 and 445, enter your tenantName, password-reset-policy and clientId

The pages should be uploaded in a blob and their url should be referred in the Azure AD B2C policies.

- *Create a storage account* as mentioned in Upload the sample content to Azure Blob Storage
- Upload the sample AD Pages in the container and note down their url.
- For the created Blob service Storage account create a CORS rule with '*' as ALLOWED ORIGINS. Select all in ALLOWED METHODS. Enter * for ALLOWED HEADERS and EXPOSED HEADERS as well.
- Customize your policy

Now in your application you can see the customized UI.

Read Getting started with azure-active-directory online: https://riptutorial.com/azure-active-directory/topic/5465/getting-started-with-azure-active-directory

Chapter 2: Azure Active Directory B2C

Introduction

Azure AD B2C is a cloud identity management solution for your web and mobile applications. It is a highly available global service that scales to hundreds of millions of identities.

Examples

Azure AD B2C - Angularjs sample (Web and Mobile) app

This sample demonstrates the use of AD B2C for securing an AngularJS based web and mobile app.

Refer https://github.com/NewtonJoshua/Azure-ADB2C-Angularjs-sample

Azure AD B2C

Azure AD B2C is a cloud identity management solution for your web and mobile applications. It is a highly available global service that scales to hundreds of millions of identities.

Web app - Hello.js

Web app implementation uses Hello.js that performs identity management with Azure AD B2C . Hello.js is a client-side JavaScript SDK for authenticating with OAuth2 web services and querying REST APIs.

Mobile app - ADAL plugin

Mobile app implementation uses ADAL Cordova Plugin Patch For B2C. This is a chopped version of Active Directory Authentication Library (ADAL) plugin for Apache Cordova apps, cordova-plugin-ms-adal that works with Azure AD B2C. The original cordova-plugin-ms-adal plugin provides easy to use authentication functionality for your Apache Cordova apps by taking advantage of Active Directory.

Decode JWT

jwtHelper of angular-jwt will take care of helping you decode the token (JWT) and check its expiration date. JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.

1. Project set up:

1. Clone or download this repository

```
git clone https://github.com/NewtonJoshua/Azure-ADB2C-Angularjs-sample.git
```

2. Install dependencies

   ```
   npm install
   ```

   ```
   bower install
   ```

2. AD set up:

ADAL-B2C configuration

1. Create an Azure AD B2C Directory

   Note the ***Domain name***, it'll be used as the *tenantName*.

2. Register your application Follow the instructions to create an application and enable both Web App and Native client. Refer Register a web application and Register a mobile/native application

   Enter the *Reply URL* as http://localhost:8100 or any port from wher you'll be serving your app.

   In *Application Claims*, select Email Addresses too.

   Note the ***Application ID*** . It'll be used as the *clientId*.

3. Create a sign-up or sign-in policy

   Note the name of the policy. It'll be used as *policy*.

4. Create a password reset policy

   Note the name of the policy. It'll be used as the *password-reset-policy*

3. AD settings:

In settings.value.js, enter the following values

- tenantName: Domain name from step 2.1
- clientId: Application ID from step 2.2
- policy: policy name from step 2.3

4. Run this sample:

Web App:

From your shell or command line run

```
ionic serve
```

Mobile App:

1. Add platforms

```
cordova platform add android
```

```
cordova platform add ios
```

2. Generate icon and splash screen resources

```
ionic cordova resources
```

3. Build the App

```
cordova build
```

For more details on building the apps refer the Cordova documentions, Android Platform Guide and iOS Platform Guide

5. Customize the Azure AD B2C user interface

The Azure AD B2C login screen can be customized to suit our branding. Refer Customizing the UI

In this sample we have two customized UI screens,

- AD B2C Sign in ansd Sign up page: adCustomPages/unified.html
- AD B2C Password reset page: adCustomPages/resetpassword.html

In adCustomPages/unified.html, at line 442 and 445, enter your tenantName, password-reset-policy and clientId

The pages should be uploaded in a blob and their url should be referred in the Azure AD B2C policies.

- *Create a storage account* as mentioned in Upload the sample content to Azure Blob Storage
- Upload the sample AD Pages in the container and note down their url.
- For the created Blob service Storage account create a CORS rule with '*' as ALLOWED ORIGINS. Select all in ALLOWED METHODS. Enter * for ALLOWED HEADERS and EXPOSED HEADERS as well.
- Customize your policy

Now in your application you can see the customized UI.

Implementation:

If you have to build an application based on this sample remember to install the required dependencies.

Web App

Dependencies:

```
bower install ng-hello --save
```

```
bower install angular-jwt --save
```

refer hello.service.js

Mobile App

Dependencies:

```
cordova plugin add https://github.com/jospete/azure-activedirectory-library-for-cordova --save
```

```
bower install angular-jwt --save
```

refer adal.service.js

Related documents:

1. Overview: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-overview
2. Azure AD - Help secure AngularJS single-page apps by using Azure AD https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-devquickstarts-angular
3. Azure AD B2C: Single-page app sign-in by using OAuth 2.0 implicit flow https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-reference-spa

Read Azure Active Directory B2C online: https://riptutorial.com/azure-active-directory/topic/10768/azure-active-directory-b2c

Chapter 3: Azure Active Directory B2C implementation using ADAL Cordova Plugin

Introduction

Examples

Azure Active Directory B2C implementation using ADAL Cordova Plugin

Refer the example here: https://github.com/NewtonJoshua/Azure-ADB2C-Angularjs-sample

Azure AD B2C

Azure AD B2C is a cloud identity management solution for your web and mobile applications. It is a highly available global service that scales to hundreds of millions of identities.

Mobile app - ADAL plugin

Mobile app implementation uses ADAL Cordova Plugin Patch For B2C. This is a chopped version of Active Directory Authentication Library (ADAL) plugin for Apache Cordova apps, cordova-plugin-ms-adal that works with Azure AD B2C. The original cordova-plugin-ms-adal plugin provides easy to use authentication functionality for your Apache Cordova apps by taking advantage of Active Directory.

Find the angularjs/ionicframework example below

Install the dependencies:

```
cordova plugin add https://github.com/jospete/azure-activedirectory-library-for-cordova --save
```

```
bower install angular-jwt --save
```

Let us have a LoginController

```
.controller('LoginController', function($scope, $state, $ionicPopup, jwtHelper, AdalService) {

    $scope.login = function(){
        AdalService.login().then(function(authResponse) {
        displayUserDetails(getUserData(authResponse));
    });

    $scope.logout = AdalService.logout;

    // Decode decode the token and diaplay the user details
    function getUserData(response) {
        var user = {};
        user.token = response.access_token || response.token;
        var data = jwtHelper.decodeToken(user.token);
        user.expires_in = new Date(response.expires * 1000) || response.expiresOn;
        user.name = data.name;
        user.email = data.emails ? data.emails[0] : '';
        user.id = data.oid;
        return user;
    };

    function displayUserDetails(user) {
        $scope.user = user;
        $ionicPopup.alert({
            title: user.name,
            template: '<b>Email:</b> ' + user.email + '<br> <b>Id:</b> <code>' + user.id +
'</code>'
        });
    }

});
```

Enter the Azure AD B2C settings here

```
.value('settings', {
    // ADAL-B2C configuration
    adalB2C: {
        tenantName: 'Enter your tenant name',
        clientId: 'Enter your client id',
        policy: 'Enter your policy name'
    }
});
```

And here is the adal.service that implements Azure AD B2C using ADAL plugin

angular .module('azureADB2C') .service('AdalService', function($q, $http, settings) {

```
    var extraQueryParams = 'nux=1';
    var userId = null;
    var redirectUri = 'https://login.microsoftonline.com/tfp/oauth2/nativeclient';
    var authority = 'https://login.microsoftonline.com/' + settings.adalB2C.tenantName;
    var resourceUri = 'https://graph.windows.net';

    this.login = function() {
        var deferredLoginResponse = $q.defer();
        var authContext = new Microsoft.ADAL.AuthenticationContext(authority);
        // Attempt to authorize user silently
        authContext.acquireTokenSilentAsync(resourceUri, settings.adalB2C.clientId, userId,
redirectUri, settings.adalB2C.policy)
            .then(function(authResponse) {
                deferredLoginResponse.resolve(authResponse);
            }, function() {
                // We require user credentials so triggers authentication dialog
                authContext.acquireTokenAsync(resourceUri, settings.adalB2C.clientId,
redirectUri, userId, extraQueryParams, settings.adalB2C.policy)
                    .then(function(authResponse) {
                        deferredLoginResponse.resolve(authResponse);
                    }, function(err) {
                        deferredLoginResponse.reject(err);
                    });
            });
        return deferredLoginResponse.promise;
    };

    this.logout = function() {
        // Step1: clear cache
        var authContext = new Microsoft.ADAL.AuthenticationContext(authority);
        authContext.tokenCache.clear();

        // Step2: make XmlHttpRequest pointing to the sign out url
        return $http.post(authority + '/oauth2/logout?post_logout_redirect_uri=' +
redirectUri);
    };

});
```

Read Azure Active Directory B2C implementation using ADAL Cordova Plugin online: https://riptutorial.com/azure-active-directory/topic/10770/azure-active-directory-b2c-implementation-using-adal-cordova-plugin

Chapter 4: Azure Active Directory B2C implementation using Hello.js

Examples

Angularjs- Azure Active Directory B2Cusing Hello.js

Refer the example in https://github.com/NewtonJoshua/Azure-ADB2C-Angularjs-sample

Web app implementation uses Hello.js that performs identity management with Azure AD B2C . Hello.js is a client-side JavaScript SDK for authenticating with OAuth2 web services and querying REST APIs.

jwtHelper of angular-jwt will take care of helping you decode the token (JWT) and check its expiration date. JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.

Find the angularjs example below

Let us have a LoginController

```
.controller('LoginController', function($scope, $state, $ionicPopup, jwtHelper,  HelloService)
{

    // Initialize
    (function initialize() {
            HelloService.initialize().then(function(authResponse) {
            displayUserDetails(getUserData(authResponse));
        });
    })();

    $scope.login = HelloService.login;
    $scope.logout = HelloService.logout;


    // Decode decode the token and display the user details
    function getUserData(response) {
        var user = {};
        user.token = response.access_token || response.token;
        var data = jwtHelper.decodeToken(user.token);
        user.expires_in = new Date(response.expires * 1000) || response.expiresOn;
        user.name = data.name;
        user.email = data.emails ? data.emails[0] : '';
        user.id = data.oid;
        return user;
    };

    function displayUserDetails(user) {
        $scope.user = user;
        $ionicPopup.alert({
            title: user.name,
            template: '<b>Email:</b> ' + user.email + '<br> <b>Id:</b> <code>' + user.id +
'</code>'
        });
    }
});
```

Enter the Azure AD B2C settings here

```
.value('settings', {
    // ADAL-B2C configuration
    adalB2C: {
        tenantName: 'Enter your tenant name',
        clientId: 'Enter your client id',
        policy: 'Enter your policy name'
    }
});
```

And here is the hello.service that implements Azure AD B2C using Hello.js

```
.service('HelloService', function(hello, $q, settings) {

    var network = 'adB2CSignInSignUp';

    this.initialize = function() {
        //initiate all policies
        hello.init({
            adB2CSignIn: settings.adalB2C.clientId,
            adB2CSignInSignUp: settings.adalB2C.clientId,
            adB2CEditProfile: settings.adalB2C.clientId
        }, {
            redirect_uri: '../',
            scope: 'openid ' + settings.adalB2C.clientId,
            response_type: 'token id_token'
        });
        var adB2CSignInSignUpPolicy = getPolicyConfiguredData();
        hello.init(adB2CSignInSignUpPolicy);
        var authResponse = hello(network).getAuthResponse();
        if (authResponse && !authResponse.error) {
            return $q.when(authResponse);
        } else {
            var error = authResponse && authResponse.error ? authResponse.error : '';
            return $q.reject(error);
        }
    };

    this.login = function() {
        hello(network).login({
            display: 'page',
            force: true
        });
    };

    this.logout = function() {
        hello(network).logout({
            force: true
        });
    };

    function getPolicyConfiguredData() {
        var adB2CSignInSignUpPolicy = {};
        adB2CSignInSignUpPolicy[network] = {
            name: 'Azure Active Directory B2C',
            oauth: {
                version: 2,
                auth: 'https://login.microsoftonline.com/tfp/' + settings.adalB2C.tenantName +
'/' + settings.adalB2C.policy + '/oauth2/v2.0/authorize',
                grant: 'https://login.microsoftonline.com/tfp/' + settings.adalB2C.tenantName
+ '/' + settings.adalB2C.policy + '/oauth2/v2.0/token'
            },
            refresh: true,
            scope_delim: ' ',
            // Don't even try submitting via form.
            // This means no POST operations in <=IE9
            form: false
        };
        adB2CSignInSignUpPolicy[network].xhr = function(p) {
            if (p.method === 'post' || p.method === 'put') {
                //toJSON(p);
                if (typeof(p.data) === 'object') {
                    // Convert the POST into a javascript object
                    try {
                        p.data = JSON.stringify(p.data);
                        p.headers['content-type'] = 'application/json';
                    } catch (e) {}
                }
            } else if (p.method === 'patch') {
                hello.utils.extend(p.query, p.data);
                p.data = null;
            }
            return true;
        };
        adB2CSignInSignUpPolicy[network].logout = function() {
            //get id_token from auth response
            var id_token = hello(network).getAuthResponse().id_token;
            //clearing local storage session
            hello.utils.store(network, null);

            //redirecting to Azure B2C logout URI
            window.location = ('https://login.microsoftonline.com/' +
settings.adalB2C.tenantName + '/oauth2/v2.0/logout?p=' + settings.adalB2C.policy +
'&id_token_hint=' +
                id_token + '&post_logout_redirect_uri=https://login.microsoftonline.com/' +
settings.adalB2C.tenantName + '/oauth2/logout');
        };
        return adB2CSignInSignUpPolicy;
    }

});
```

Read Azure Active Directory B2C implementation using Hello.js online: https://riptutorial.com/azure-active-directory/topic/10771/azure-active-directory-b2c-implementation-using-hello-js

Credits

S. No	Chapters	Contributors
1	Getting started with azure-active-directory	Community, Newton Joshua
2	Azure Active Directory B2C	Newton Joshua
3	Azure Active Directory B2C implementation using ADAL Cordova Plugin	Newton Joshua
4	Azure Active Directory B2C implementation using Hello.js	Newton Joshua