Getting Started with Azure AD B2C

{IEF - custom policies}

**Lab**

Terms of Use

© 2018 Microsoft Corporation. All rights reserved.

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see **Microsoft Copyright Permissions** at <http://www.microsoft.com/permission>

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The Microsoft company name and Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

**This document reflects current views and assumptions as of the date of development and is subject to change.  Actual and future results and trends may differ materially from any forward-looking statements.  Microsoft assumes no responsibility for errors or omissions in the materials.**

**THIS DOCUMENT IS FOR INFORMATIONAL AND TRAINING PURPOSES ONLY AND IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT.**

# Lab Overview

###### Abstract

This is a very short lab to introduce you to Azure AD B2C IEF custom policies. This lab demonstrates how to get started with an Azure AD B2C tenant, create a B2C application, and make your first Sign up or Sign in policy. You will also be able to provide a connection to a RESTful API call to retrieve attributes during a login flow.

## Prerequisites

In order to complete this lab, you will need to have access to an active Azure subscription. Additionally, access to [Visual Studio Code](https://code.visualstudio.com/download) and the [Azure AD B2C extension](https://marketplace.visualstudio.com/items?itemName=AzureADB2CTools.aadb2c).

# Exercise 1: Create an Azure AD B2C

Skip this exercise if you already have an Azure AD B2C tenant. Simply switch into this tenant from the top-right corner.

Each exercise consists of a scenario and learning objectives, the scenario describes the purpose of the exercices, while the objectives are listed and have bullet points.

1. Open the portal <https://portal.azure.com>
2. **Click** on “Create a resource” in the left hand navigation
3. Search for “B2C” and select “Azure AD B2C”. **Click** on “Create”.
4. Follow the steps to finish creating a tenant. You will have to wait for a few minutes for the tenant to be created

# Exercise 2: Link a subscription

Skip this exercise if you already have an Azure AD B2C tenant with a subscription linked to it.

Each exercise consists of a scenario and learning objectives, the scenario describes the purpose of the exercices, while the objectives are listed and have bullet points.

### Scenario

1. In this exercise, you will link a subscription to the newly created tenant. Click on your identity in the upper right and make sure you are in the same tenant that you used to create your Azure AD B2C tenant (you should NOT be in your Azure AD B2C tenant)
2. **Click** on “Create a resource” in the left-hand navigation
3. Search for “B2C” and select “Azure AD B2C”. Click on “Create”.
4. Click on “Link an existing Azure AD B2C Tenant to my Azure subscription”
5. Complete the page. Make sure to select the tenant you created above. For the resource group, create a new one (name it “B2C Resources”) or use an existing one.

# Exercise 3: Create an application

Each exercise consists of a scenario and learning objectives, the scenario describes the purpose of the exercices, while the objectives are listed and have bullet points.

### Scenario

1. In this exercise, you will create a B2C application that will represent your real-world application when the applications make a request to Azure AD B2C. Click your identity in the upper right and switch into the Azure AD B2C tenant.
2. Open the **Azure AD B2C** menu by searching for it under **More Services**
3. Select **Applications** and then select **Add**
4. Add a **Name** for the application, for example Training App.
5. Select **Yes** for **Include web app/web api**.
6. In the **reply URL**, type in https://jwt.ms
7. Leave the rest of the settings to their default values and select **Create**.

# Exercise 4: Add signing and encryption keys

### Scenario

In this exercise, you will need to create the signing, encryption keys that will be required for the Azure AD B2C custom policy in the Technical Profiles that are used during sign in or sign-up.

1. Under Azure AD B2C resource tab, Select **Identity Experience Framework – PREVIEW**

##### Step 1: Create the signing Key

1. Select **Policy Keys** and then select **Add**.
2. For **Options**, choose Generate.
3. In **Name**, enter TokenSigningKeyContainer. The prefix B2C\_1A\_ will be added automatically.
4. For **Key type**, select **RSA**.
5. For **Key usage**, select **Signature**.
6. Click **Create**.

##### Step 2: Create the encryption Key

1. Select **Policy Keys** and then select **Add**.
2. For **Options**, choose Generate.
3. In **Name**, enter TokenEncryptionKeyContainer. The prefix B2C\_1A\_ will be added automatically.
4. For **Key type**, select **RSA**.
5. For **Key usage**, select **Encryption**.
6. Click **Create**.

# Exercise 5: Register applications

### Scenario

Azure AD B2C requires you to register two applications within Azure AD (not B2C) that are used to sign up and sign in users: **IdentityExperienceFramework** (a web app), and **ProxyIdentityExperienceFramework** (a native app) with delegated permission from the **IdentityExperienceFramework** app. Local accounts exist only in your tenant. Your users sign up with a unique email address/password combination to access your tenant-registered applications.

##### Step 1: Register the IdentityExperienceFramework application

1. Choose **All services** in the top-left corner of the Azure portal, search for and select **Azure Active Directory**, and then select **App registrations**.
2. Select **New application registration**.
3. For **Name**, enter IdentityExperienceFramework.
4. For **Application type**, choose **Web app/API**.
5. For **Sign-on URL**, enter https://your-tenant-name.b2clogin.com/your-tenant-name.onmicrosoft.com, where your-tenant-name is your Azure AD B2C tenant domain name.
6. Click **Create**.
7. After it's created, **copy the application ID and save it in Notepad for later use**

##### Step 2: Register the ProxyIdentityExperienceFramework application

1. Select **App registrations**, and then select **New application registration**.
2. For **Name**, enter ProxyIdentityExperienceFramework.
3. For **Application type**, choose **Native**.
4. For **Redirect URI**, enter https://your-tenant-name.b2clogin.com/your-tenant-name.onmicrosoft.com, where your-tenant-name is your Azure AD B2C tenant.
5. Click **Create**.
6. After it's created, **copy the application ID and save it in Notepad for later use**
7. On the Settings page, select **Required permissions**, and then select **Add**.
8. Select **Select an API**.
9. Search for and select **IdentityExperienceFramework**, and then click **Select**.
10. Select the check box next to **Access IdentityExperienceFramework**, click **Select**, and then click **Done**.
11. Select **Grant Permissions**, and then confirm by selecting **Yes**.

# Exercise 6: Download starter pack and modify policies

### Scenario

In this scenario you will download the .zip file and extract the required files for the Azure AD custom policies. While not required, we recommend using an XML editor, try [**Visual Studio Code**](https://code.visualstudio.com/download) to make it easier to read and navigate through the policies.

1. [Download the .zip file](https://github.com/Azure-Samples/active-directory-b2c-custom-policy-starterpack/archive/master.zip) and extract, or run:

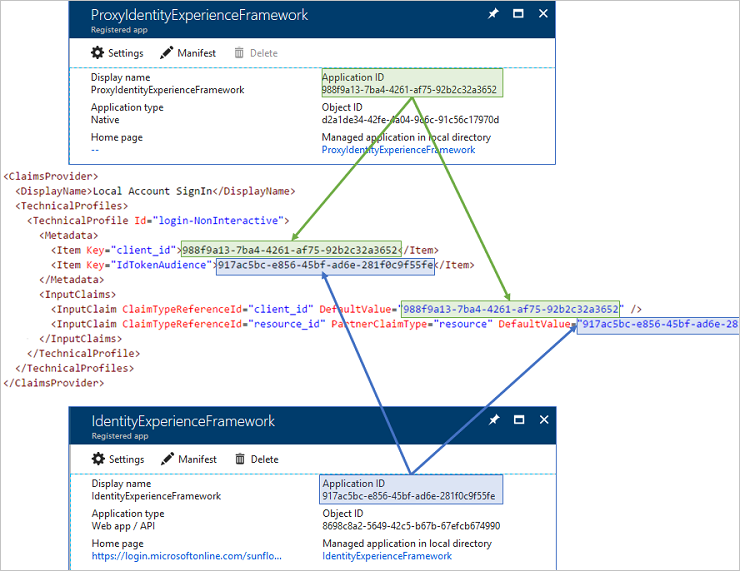
git clone https://github.com/Azure-Samples/active-directory-b2c-custom-policy-starterpack

1. In the **SocialAndLocalAccounts** folder, edit all five files replacing with the name for your tenant. For example, contosoTenant.onmicrosoft.com.

\*\*Make sure you edit all files below or you will receive an error during upload\*\*

* 1. *PasswordReset.xml*
  2. *ProfileEdit.xml*
  3. *SignUpOrSignin.xml*
  4. *TrustFrameworkBase.xml*
  5. *TrustFrameworkExtension.xml*

1. Open the **TrustFrameworkExtensions.xml file** and find the element <TechnicalProfile Id="login-NonInteractive">.
2. **Replace** both instances of IdentityExperienceFrameworkAppId **with the application ID** of the Identity Experience Framework application that you created earlier. Replace both instances of ProxyIdentityExperienceFrameworkAppId **with the application ID** of the Proxy Identity Experience Framework application that you created earlier. The following example shows the **login-NonInteractive** technical profile after the changes:



1. Save your extension file.

# Exercise 7: Facebook - Set up sign-up and sign-in using Azure Active Directory B2C

### Scenario

To use a Facebook account as an identity provider in Azure Active Directory (Azure AD) B2C, you need to create an application in your tenant that represents it. If you don’t already have a Facebook account, you can get it at <https://www.facebook.com/>.

#### Step 1: Create a Facebook application

1. Sign in to [Facebook for developers](https://developers.facebook.com/) with your Facebook account credentials.
2. If you have not already done so, you need to register as a Facebook developer. To do this, select **Register** on the upper-right corner of the page, accept Facebook's policies, and complete the registration steps.
3. Select **My Apps** and then click **Add a New App**.
4. Enter a **Display Name** and a valid **Contact Email**.
5. Click **Create App ID**. This may require you to accept Facebook platform policies and complete an online security check.
6. Select **Settings** > **Basic**.
7. Choose a **Category**, for example Business and Pages. This value is required by Facebook, but not used for Azure AD B2C.
8. At the bottom of the page, select **Add Platform**, and then select **Website**.
9. In **Site URL**, enter https://your-tenant-name.b2clogin.com/ replacing your-tenant-name with the name of your tenant. Enter a URL for the **Privacy Policy URL**, for example http://www.contoso.com. The policy URL is a page you maintain to provide privacy information for your application.
10. Select **Save Changes**.
11. At the top of the page, copy the value of **App ID**.
12. Click **Show** and **copy** the value of **App Secret**. You use both of them to configure Facebook as an identity provider in your tenant. ***App Secret*** *is an important security credential*. **SAVE IT THIS FOR LATER**
13. Select **Products**, and then select **Set up** under **Facebook Login**.
14. Select **Settings** under **Facebook Login**.
15. In **Valid OAuth redirect URIs**, enter https://your-tenant-name.b2clogin.com/your-tenant-name.onmicrosoft.com/oauth2/authresp. Replace your-tenant-name with the name of your tenant. Click **Save Changes** at the bottom of the page.
16. To make your Facebook application available to Azure AD B2C, click the Status selector at the top right of the page and turn it **On** to make the Application public, and then click **Confirm**. At this point the Status should change from **Development** to **Live**.

#### Step 2: Create a policy key

You need to store the client secret that you previously recorded in your Azure AD B2C tenant.

1. Sign in to the [Azure portal](https://portal.azure.com/).
2. Make sure you're using the directory that contains your Azure AD B2C tenant by clicking the **Directory and subscription filter** in the top menu and choosing the directory that contains your tenant.
3. Choose **All services** in the top-left corner of the Azure portal, and then search for and select **Azure AD B2C**.
4. On the Overview page, select **Identity Experience Framework - PREVIEW**.
5. Select **Policy Keys** and then select **Add**.
6. For **Options**, choose **Manual**.
7. Enter a **Name** for the policy key as **FacebookSecret**. The prefix B2C\_1A\_ is added automatically to the name of your key.
8. In **Secret**, enter your client secret that you previously recorded in [**Ste****p 1.12**](#_Step_1:_Create).
9. For **Key usage**, select **Signature**.
10. Click **Create**.

#### Step 3: Add a claim provider

If you want users to sign in by using a Google account, you need to define the account as a ‘*claims provider*’ that Azure AD B2C can communicate with through an endpoint. The endpoint provides a set of claims that are used by Azure AD B2C to verify that a specific user has authenticated.

You can define a Facebook account as a claims provider by adding it to the **ClaimsProviders** element in the extension file of your policy.

In the *TrustFrameworkExtensions.xml* file, replace the value of **facebook\_clientid** with the Facebook application ID from [**Ste****p 1.11**](#_Step_1:_Create)

1. <ClaimsProvider>
2. <DisplayName>Facebook</DisplayName>
3. <TechnicalProfiles>
4. <TechnicalProfile Id="Facebook-OAUTH">
5. <Metadata>
6. <Item Key="client\_id">facebook\_clientid</Item>
7. <Item Key="scope">email public\_profile</Item>
8. <Item Key="ClaimsEndpoint">https://graph.facebook.com/me?fields=id,first\_name,last\_name,name,email</Item>
9. </Metadata>
10. </TechnicalProfile>
11. </TechnicalProfiles>
12. </ClaimsProvider>
13. Save the file

# Exercise 8: Upload the policies

### Scenario

In this exercise, you will be uploading the custom policies (XML files) to Azure AD B2C admin console. You achieve this by leveraging the Identity Experience Framework UX.

1. **Navigate** to <https://portal.azure.com>
2. Select **All Services**, and then enter **Azure AD B2C**
3. Select **Azure AD B2C**, and then select **Identity Experience Framework**
4. Click **Upload Custom Polic**y.
5. Enable **Overwrite the policy if it exists**, and click the folder icon. Browse to and select the *TrustFrameworkExtension.xml* under **SocialAndLocalAccounts** folder outlined in Exercise 6 Step 2:
   1. *TrustFrameworkExtension.xml*
6. Select **Upload Policy**

# Exercise 9: Test the custom policy

### Scenario

In this exercise, you will test signing up and logging into your JWT application with both a local account and a Facebook account

1. On the Custom Policies page, select **B2C\_1A\_signup\_signin**.
2. Select **Run now**.
3. You should be able to sign up using your **Facebook social identity**.
4. Sign in with the same account to confirm that you have the correct configuration.

# Configuring RESTful Provider

###### Abstract

In this portion of the lab you will configure a RESTful API Provider to perform an API call to a third-party service during a sign-up or sign-in flow. This scenario will retrieve a Membership Number/Loyalty Number of a customer. Once you understand how to do this, you can easily integrate this into any of your flows called by your application/service.

## Prerequisites

Have a local account or social identity that can successfully authenticate using Azure AD custom policies while authenticating to an application. If you do not have a social identity provider, you can still configure this with a local account and test this successfully.

# Exercise 1: Smart Copy OrchestrationSteps Base to Extension

### Scenario

In this exercise, you will need to copy over the OrchestrationSteps from the base file to the Extension file. The TrustFrameworkBase.xml file will inherent what is in the TrustFrameworkExtension.xml file therefore, by following best practices we will not modify the base file and instead add the additional Orchestrated Steps to the extension file.

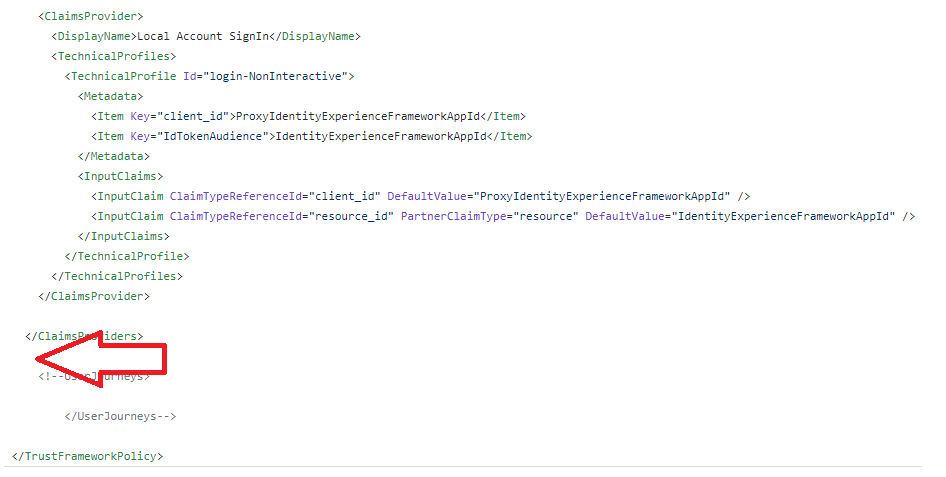
This practice should be done if you require to make any changes to the IEF such as adding additional claim schemas or additional orchestration steps to your user journey.

#### Step 1: Smart Copy Base Extension

1. Navigate to the TrustFrameworkBase.xml file within Visual Studio Code
2. Press Ctrl + “F” to pull up the find bar and type “UserJourneys”
3. Right Click **<UserJourneys>** at the beginning of the XML and Select **B2C Smart copy**
   1. This copies everything between **<UserJourneys></UserJourneys>**

#### Step 2: Smart Paste to ExtensionFile

1. Navigate to the TrustFrameworkExtensions.xml file within Visual Studio Code
2. Scroll at the bottom of the page
3. After the **ClaimsProviders**, right click and Select **B2C Smart paste**
   1. This will paste the entire **UserJourneys**



You should now have all of your UserJourneys available in your ExtensionFiles.

# Exercise 2: Add client secret and client id for REST API call

### Scenario

In this exercise, you will need to create the Client ID and Client Secret in order to perform basic authentication with the REST API when it attempts to execute a REST call to gain the **loyaltynumber** of the user that is stored in another service. The keys below will be referenced in the B2C policies in a later step.

##### Step 1: Create the Rest Client Id

1. Select **Policy Keys** and then select **Add**.
2. For **Options**, choose Manual.
3. In **Name**, enter B2cRestClientId. The prefix B2C\_1A\_ might be added automatically.
4. In **Secret**, enter ABC123. The prefix B2C\_1A\_ might be added automatically.
5. For **Key usage**, select **Signature**.
6. Click **Create**.

##### Step 2: Create the Rest Client Secret

1. Select **Policy Keys** and then select **Add**.
2. For **Options**, choose Manual.
3. In **Name**, enter B2cRestClientSecret. The prefix B2C\_1A\_ will be added automatically.
4. In **Secret**, enter XYZ456. The prefix B2C\_1A\_ will be added automatically.
5. For **Key usage**, select **Signature**.
6. Click **Create**.

# Exercise 3: Create a REST API ClaimsProvider

### Scenario

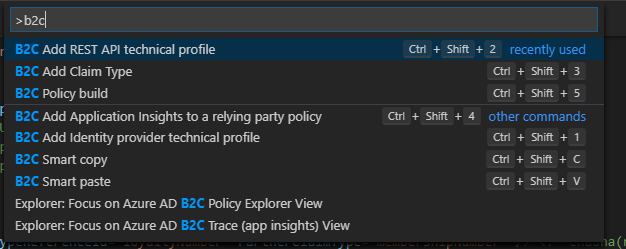
In this exercise, you will add the ClaimProvider Technical Profile for the REST API. This Technical Profile will contain the configuration for your REST API such as Client ID, Client Secret and endpoints.

1. Navigate to the TrustFrameworkExtension.xml file within Visual Studio Code
2. In the ClaimsProvider section, click above the Facebook ClaimsProvider and hit enter a couple times to create a space for your new ClaimsProvider.
3. Enter <ClaimsProvider and press Enter for autocomplete

<ClaimsProvider>

</ClaimsProvider>

1. Click within the **<ClaimsProvider></ClaimsProvider>** brackets and Press Ctrl +Shift + P on the keyboard
2. Type **B2C** and choose B2C Add REST API technical profile



1. In **Provide a name**, enter **REST-API-SignUp**
2. In **Service URL**, enter [**https://partnertrainingaadb2capi.azurewebsites.net/api/identity/membership**](https://partnertrainingaadb2capi.azurewebsites.net/api/identity/membership)
3. In **Select Authentication Type**, select Basic
4. In the OutputClaim section, replace **PartnerClaimType = ”loyaltyNumber”** to equal **membershipNumber** instead.

**EXAMPLE**

<OutputClaims>

<OutputClaim ClaimTypeReferenceId="loyaltyNumber" PartnerClaimType="membershipNumber" />

</OutputClaims>

# Exercise 4: Add the claim “loyaltyNumber” to the schema

### Scenario

In this exercise, you will be required to add the claim called loyaltyNumber to your schema inside the TrustFrameworkExtension.xml file. When calling a claim inside a technical profile, there must be a correlated claim that is instantiated within the schema. That is defined in either the base or extension file. In this example, we will be defining this within the TrustFrameworkExtension.xml file.

1. **Navigate** to the TrustFrameworkExtensions.xml file.
2. Within the **<BuildingBlocks></BuildingBlocks>** brackets, Enter the following **ClaimsSchema** value:

<ClaimsSchema>

<ClaimType Id="loyaltyNumber">

<DisplayName>Your loyalty number from your membership</DisplayName>

<DataType>string</DataType>

  <AdminHelpText>This value is taken from the REST API</AdminHelpText>

</ClaimType>

</ClaimsSchema>

**EXAMPLE**

<BuildingBlocks>

<ClaimsSchema>

<ClaimType Id="loyaltyNumber">

<DisplayName>Your loyalty number from your membership</DisplayName>

<DataType>string</DataType>

  <AdminHelpText>This value is taken from the REST API</AdminHelpText>

</ClaimType>

</ClaimsSchema>

</BuildingBlocks>

# Exercise 5: Add the Orchestration Step to call the REST API

### Scenario

In this exercise, you will add the orchestrated step to trigger the REST API after a successful sign-in or sign-up flow. This is done within the UserJourney that is contained within the TrustFrameworkExtension.xml file. This step is required in order for the REST API ClaimsProvider configured earlier to be triggered.

1. **Navigate** to the TrustFrameworkExtensions.xml file.
2. Scroll at the bottom of the file until you see the OrchestrationSteps listed within the **<UserJourneys><UserJourneys/>**
3. We will be adding a new OrchestrationStep the REST-API-SignUp between the existing Order “6” and ”7”, which will leave us with a *total of 8 Orchestration Steps*.
   1. Input the following:

<OrchestrationStep Order="6" Type="ClaimsExchange">

<Preconditions>

<Precondition Type="ClaimsExist" ExecuteActionsIf="true">

<Value>objectId</Value>

<Action>SkipThisOrchestrationStep</Action>

</Precondition>

</Preconditions>

<ClaimsExchanges>

<ClaimsExchange Id="AADUserWrite" TechnicalProfileReferenceId="AAD-UserWriteUsingAlternativeSecurityId"/>

</ClaimsExchanges>

</OrchestrationStep>

<OrchestrationStep Order="7" Type="ClaimsExchange">

<ClaimsExchanges>

<ClaimsExchange Id="AnyValue" TechnicalProfileReferenceId="REST-API-SignUp" />

</ClaimsExchanges>

</OrchestrationStep>

<OrchestrationStep Order="8" Type="SendClaims" CpimIssuerTechnicalProfileReferenceId="JwtIssuer"/>

* 1. Remember to update the last OrchestrationStep to be in sequence with the ordering, which is changing “7” to “8” as highlighted above.

# Exercise 6: Add the loyaltyNumber claim to be added in the JWT token

### Scenario

In this exercise, you will be adding the outbound **loyaltyNumber** claim to be added to the JWT token. This configuration tells Azure AD B2C explicitly to send the specific claim to your application. This is done by adding an **OutputClaim** to the **SignUpOrSignIn policy**.

1. **Navigate** to the SignUpOrSignin.xml file.
2. **Add** the text <OutputClaim ClaimTypeReferenceId=“loyaltyNumber” /> within the **<OutputClaims></OutputClaims>** brackets

**EXAMPLE**

<OutputClaims>

<OutputClaim ClaimTypeReferenceId="displayName" />

<OutputClaim ClaimTypeReferenceId="givenName" />

<OutputClaim ClaimTypeReferenceId="surname" />

<OutputClaim ClaimTypeReferenceId="email" />

<OutputClaim ClaimTypeReferenceId="objectId" PartnerClaimType="sub"/>

<OutputClaim ClaimTypeReferenceId="identityProvider" />

<OutputClaim ClaimTypeReferenceId="loyaltyNumber" />

</OutputClaims>

# Exercise 7: Upload the policies

### Scenario

In this exercise, you will be uploading the custom policies (XML files) to Azure AD B2C admin console

1. On the Custom Policies page of Identity Experience Framework, select **Upload Policy.**
   1. **Navigate** to <https://portal.azure.com>
   2. Select **All Services**, and then enter **Azure AD B2C**
   3. Select **Azure AD B2C**, and then select **Identity Experience Framework**
   4. Select **Upload Policy**
2. In this order, upload *TrustFrameworkBase.xml*, *TrustFrameworkExtensions.xml*, *SignUpOrSignin.xml*, *ProfileEdit.xml*, and *PasswordReset.xml*. When a file is uploaded, the name of the policy file is prepended with B2C\_1A\_.

# Exercise 8: Test the custom policy

### Scenario

In this exercise, you will test signing up and logging into your JWT application with both a local account and a Facebook account

1. On the Custom Policies page, select **B2C\_1A\_signup\_signin**.
2. Select **Run now**.
3. You should be able to sign up using **either local or Google social identity account**.
4. Sign in with the same account to confirm that you have the correct configuration.

# Reference Documentation

* [Set up sign-up and sign-in with a Facebook account using Azure Active Directory B2C](https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-setup-fb-app)