

# Azure Active Directory B2C: Advance user experience customization

By using the page UI customization feature, you can customize the look and feel of any policy. You can also maintain brand and visual consistency between your application and Azure AD B2C. In this article we will show you how to:

1. Set the page title and icon
2. Render dynamic HTML content
3. Change the order of policy sign-up attribute
4. Change the order of social accounts identity providers in the sign-up or sign-in page.
5. Render HTML elements inside the `<div id="api">` HTML element.

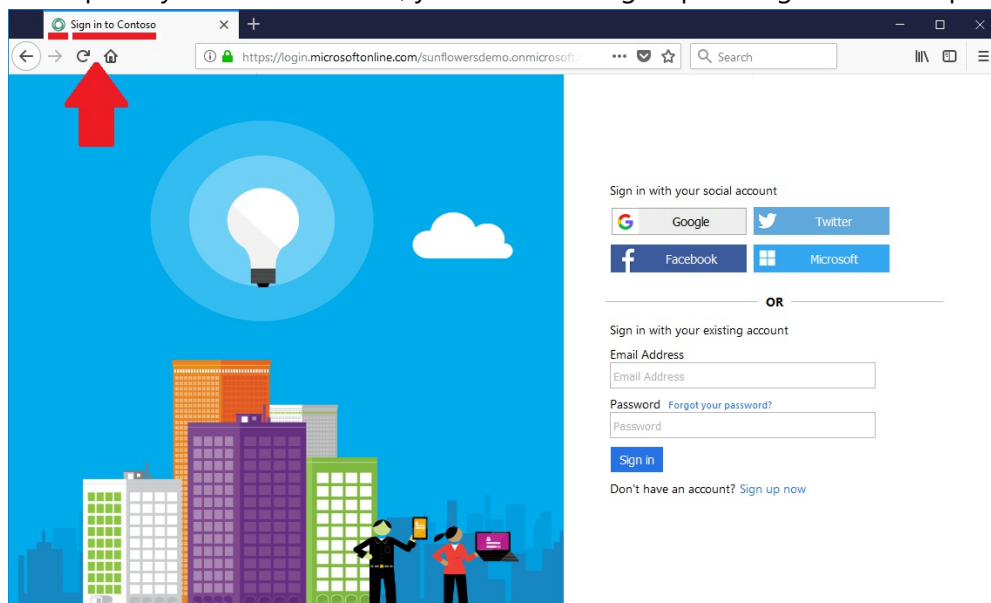
## 1. Change page title and Icon

Azure AD B2C merges UI elements with the HTML content that's loaded from your URL and then displays the page to the customer. Azure AD B2C also reads the page title and Favicon from your HTML content, and shows them in the user browser.

Following example shows how to set the page tile and Favicon:

```
<title>My custom title</title>
<link rel="icon" href="https://sunflowersdemo.blob.core.windows.net/azure-ad-b2c/CustomFavicon.gif" type="image/gif" sizes="16x16">
```

After you change and upload your HTML content, you will have a sign-up and sign-in custom policy with your



title and Favicon.

## 2. Render dynamic HTML content

With Azure AD B2C policy, you can localize your HTML content by turning on 'Language customization'. Enabling this feature allows you to send the current language used by the end user to your HTML page. You set up a folder structure to host content for specific languages and Azure AD B2C calls the right one if you put the wildcard value, `{Culture:RFC5646}`. For example, create following structure:

- **English** `https://server-name/en/unified.html`
- **Spanish** `https://server-name/es/unified.html`
- **German** `https://server-name/de/unified.html`

In Azure AD B2C policy, you configure the url to include Azure AD B2C `{Culture:RFC5646}` variable. For example: `https://account-name.blob.core.windows.net/{Culture:RFC5646}/unified.html`

## 2.1 Dynamic HTML content using custom polices

With custom policies, you have more control. You can send more variables to your HTML endpoint. Then you can dynamically change the page content accordingly. For example, you can render different content based on: localization, application Id, custom query string parameter.

Here the list of Azure AD B2C variables you can send to your HTML content endpoint.

Variable	Notes	Example
<code>{Policy:TrustFrameworkTenantId}</code>	The trust framework tenant id	contosob2c.onmicrosoft.com
<code>{Policy:RelyingPartyTenantId}</code>	The tenant id of the relying party	contosob2c.onmicrosoft.com
<code>{Policy:TenantObjectId}</code>	The tenant id of the policy	de678512-5915-46fd-ae83-a5df4ec052be
<code>{Policy:PolicyId}</code>	The policy id that is executing now	B2C_1A_signup_signin
<code>{Culture:LanguageName}</code>	The two letter ISO code for the language	en

{Culture:RegionName}	The two letter ISO code for the region	US
{Culture:RFC5646}	The RFC5646 language code	en-US
{Culture:LCID}	The LCID of language code	1033
{OIDC:ClientId}	The Open Id Connect client_id parameter. This is the application ID that the Azure portal assigned to your app.	e766b17d-50b7-433f-a0dd-308b55a57189
{OIDC:LoginHint}	The Open Id Connect <b>login_hint</b> parameter	mike@contoso.com
{OIDC:DomainHint}	The Open Id Connect <b>domain_hint</b> parameter	Facebook.com
{OAUTH-KV:query string}	Any content parameter in a query string	<a href="#">Read more</a>

To pass those variables to your HTML content endpoint, edit your relying party policy (for example, SignUpOrSignIn.xml). If it doesn't exist already, add a child node `<UserJourneyBehaviors>` to the `<RelyingParty>` node. It must be located immediately after the `<DefaultUserJourney />` element. Add the following node as a child of the `<UserJourneyBehaviors>` element.

```
<UserJourneyBehaviors>
  <ContentDefinitionParameters>
    <Parameter Name="RFC5646">{Culture:RFC5646}</Parameter>
    <Parameter Name="ClientId">{OIDC:ClientId}</Parameter>
  </ContentDefinitionParameters>
</UserJourneyBehaviors>
```

After the changes, your policy should look similar to following one:

```
<RelyingParty>
  <DefaultUserJourney ReferenceId="SignUpOrSignIn" />
  <UserJourneyBehaviors>
    <ContentDefinitionParameters>
      <Parameter Name="LCID">{Culture:LCID}</Parameter>
      <Parameter Name="RFC5646">{Culture:RFC5646}</Parameter>
      <Parameter Name="ClientId">{OIDC:ClientId}</Parameter>
    </ContentDefinitionParameters>
  </UserJourneyBehaviors>
```

Azure AD B2C calls your HTML content with the query string parameters you specified. For example: `https://your-web-app/unified.aspx?RFC5646=en-US&ClientId=0239a9cc-309c-4d41-87f1-31288feb2e82`

### 3. Change the order of policy sign-up attribute

To change the appearance of the attributes Azure AD B2C presents to the user, during sign-up or self asserted:

1. Open your policy properties

2. Click on **Edit**
3. On the B2C features blade, click **Page UI customization**.
4. And then click on **Local account sign-up page**
5. Drag and drop the attribute to the right position.

### 3.1 Change the order of policy sign-up attribute using custom policies

With custom policy, you control the order of the attribute of sign-up or self asserted pages, by placing the `<OutputClaims>` element in the desired order. For example, in following `LocalAccountSignUpWithLogonEmail` technical profile, B2C renders the attributes if following order: email, password, displayName, givenName, and surName.

```
<TechnicalProfile Id="LocalAccountSignUpWithLogonEmail">
  <DisplayName>Email signup</DisplayName>
  <Protocol Name="Proprietary" Handler="Web.TPEngine.Providers.SelfAssertedAttributeProvider, Web.TPEngine">
    <Metadata>
      <Item Key="IpAddressClaimReferenceId">IpAddress</Item>
      <Item Key="ContentDefinitionReferenceId">api.localaccountsignup</Item>
      <Item Key="language.button_continue">Create</Item>
    </Metadata>
    <CryptographicKeys>
      <Key Id="issuer_secret" StorageReferenceId="B2C_1A_TokenSigningKeyContainer" />
    </CryptographicKeys>
    <InputClaims>
      <InputClaim ClaimTypeReferenceId="email" />
    </InputClaims>
    <OutputClaims>
      <OutputClaim ClaimTypeReferenceId="objectId" />
      <OutputClaim ClaimTypeReferenceId="email" PartnerClaimType="Verified.Email" Required="true" />
      <OutputClaim ClaimTypeReferenceId="newPassword" Required="true" />
      <OutputClaim ClaimTypeReferenceId="reenterPassword" Required="true" />
      <OutputClaim ClaimTypeReferenceId="executed-SelfAsserted-Input" DefaultValue="true" />
      <OutputClaim ClaimTypeReferenceId="authenticationSource" />
      <OutputClaim ClaimTypeReferenceId="newUser" />

      <!-- Optional claims, to be collected from the user -->
      <OutputClaim ClaimTypeReferenceId="displayName" />
      <OutputClaim ClaimTypeReferenceId="givenName" />
      <OutputClaim ClaimTypeReferenceId="surName" />
    </OutputClaims>
  </Protocol>
</TechnicalProfile>
```

### 4. Change the order of Social accounts identity providers in the sign-up or sign-in page.

With CSS, you can control the order of the identity providers buttons. For example, in the following HTML segment: The identity provider list: Google, Twitter, Facebook, and Microsoft

```

<div class="options">
  <div>
    <button class="accountButton firstButton" id="GoogleExchange" tabindex="1">Google</button>
  </div>
  <div>
    <button class="accountButton" id="TwitterExchange" tabindex="1">Twitter</button>
  </div>
  <div>
    <button class="accountButton" id="FacebookExchange" tabindex="1">Facebook</button>
  </div>
  <div>
    <button class="accountButton" id="MicrosoftAccountExchange" tabindex="1">Microsoft</button>
  </div>
</div>

```

But with following CSS directives, you can change the order

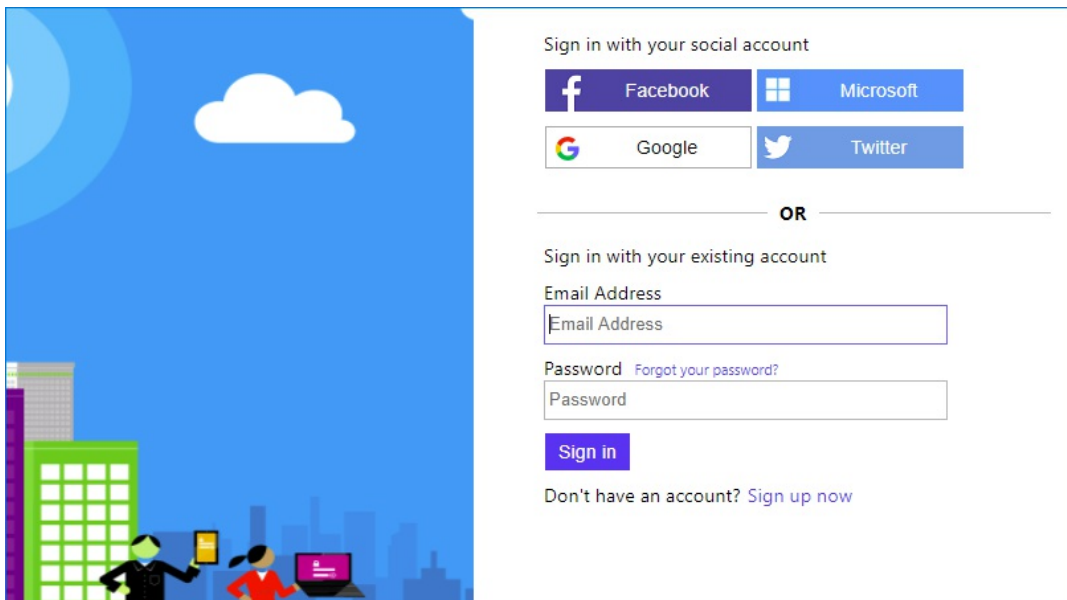
```

<style>
div.options {
  display: flex;
  flex-flow: wrap;
}

/* Standard syntax */
.options div:nth-child(1) {order: 3; }
.options div:nth-child(2) {order: 4;}
.options div:nth-child(3) {order: 1;}
.options div:nth-child(4) {order: 2;}
</style>

```

Now, first button moved to 3, second moved to 4, third moved to 1, while the fourth became 2.

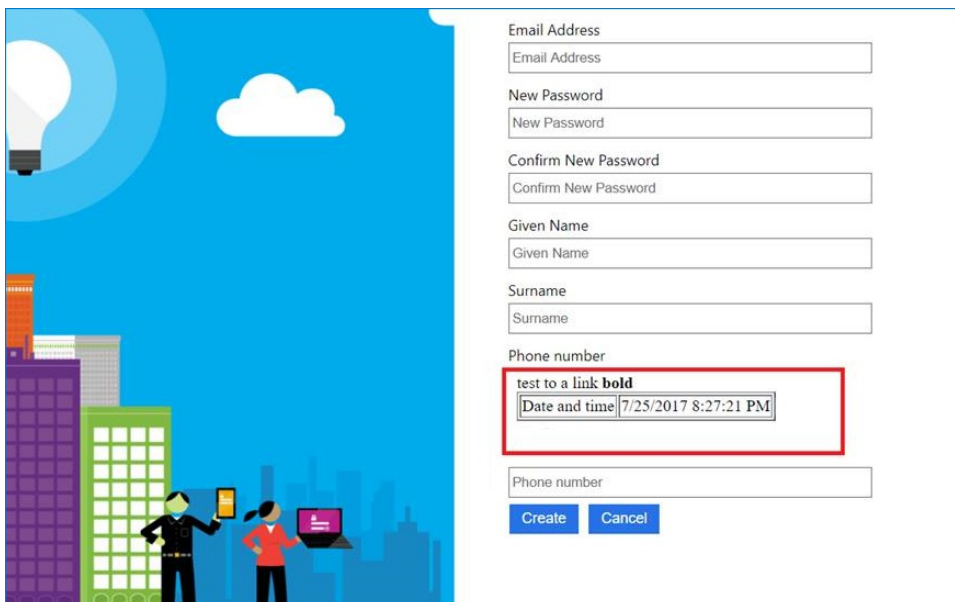


**Note:** With custom policy, place the elements inside the `<ClaimsProviderSelections>` in the correct order.

5. Render HTML elements inside the `<div id="api">` HTML element.

Azure AD B2C renders the content of the `<div id="api">` element. But using CSS content directive with svg data, it is possible to add your HTML elements, such as, Tables, Bold, Italic, DIV etc. to the page without custom domain. Following CSS adds the HTML after the label for phoneNumber. [Read more here](#)

```
label[for=phoneNumber]::after {
    content:
url('data:image/svg+xml;%20charset=utf8,%20%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20height%3D%22160%22%20width%3D%22300%22%3E%0A%0A%20%20%3CforeignObject%20y%3D%220%22%20x%3D%220%22%20height%3D%22100%25%22%20width%3D%22100%25%22%3E%0A%3Cbody%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F1999%2Fhtml%22%3E%0A%20%20%20%20%20%20%20%20%20%3Ca%20href%3D%22http%3A%2F%2Fwww.google.com%22%3Etest%20to%20a%20link%3C%2Fa%3E%20%3Cb%3Ebold%3C%2Fb%3E%0A%3Ctable%20border%3D%221%22%3E%3Ctr%3E%3Ctd%3EDate%20and%20time%3C%2Ftd%3E%3Ctd%3E%20%40DateTime.Now.ToString()%20%3C%2Ftd%3E%3C%2Ftr%3E%3C%2Ftable%3E%0A%20%20%20%20%20%20%20%3C%2Fbody%3E%0A%20%20%3C%2FforeignObject%3E%0A%3C%2Fsvg%3E');
}
```



Email Address  
Email Address

New Password  
New Password

Confirm New Password  
Confirm New Password

Given Name  
Given Name

Surname  
Surname

Phone number  
test to a link **bold**  
Date and time 7/25/2017 8:27:21 PM

Phone number

Create Cancel