Azure AD B2C is responsible for handling user account sign-up, sign-in, profile edit and password reset functionalities, AAD B2C has its own login portal management, It is an authentication service for publicly facing applications. In addition to provide authentication service for local email accounts, it also integrates with other third-party identity providers such as Google, Facebook, LinkedIn to provide a one-stop shop for authentication. Azure AD B2C provides a directory that can hold 100 custom attributes per user.  
  
User journey option includes progressive profiling. Progressive profiling allows our customers to quickly complete their first transaction by collecting a minimal amount of information. Then, gradually collect more profile data from the customer on future sign-ins. In Azure Active Directory B2C (Azure AD B2C), a tenant represents our organization and is a directory of users. Each Azure AD B2C tenant is distinct and separate from other Azure AD B2C tenants. The primary resources we work with in an Azure AD B2C tenant are:  
  
An Azure AD B2C tenant is the first resource we need to create to get started with Azure AD B2C  
  
Directory - The directory is where Azure AD B2C stores our users' credentials, profile data, and our application registrations.  
Application registrations - Register our web applications with Azure AD B2C to enable identity management.  
User flows and custom policies - Create identity experiences for our applications with built-in user flows and fully configurable custom policies:  
User flows help us quickly enable common identity tasks like sign-up, sign-in, and profile editing.  
Custom policies let us build complex identity workflows unique to our organization, customers, employees and citizens.  
Sign-in options - Azure AD B2C offers various sign-up and sign-in options for users of our applications:  
Username, email, and phone sign-in - Configure our Azure AD B2C local accounts to allow sign-up and sign-in with a username, email address, phone number, or a combination of methods.  
Social identity providers - Federate with social providers like Facebook, LinkedIn, or Twitter.  
External identity providers - Federate with standard identity protocols like OAuth 2.0, OpenID Connect, and more.  
Keys - Add and manage encryption keys for signing and validating tokens, client secrets, certificates, and passwords.  
  
Azure AD B2C lets us manage common attributes of consumer account profiles. For example display name, surname, given name, city, and others.  
  
We can also extend the Azure AD schema to store additional information about our users. For example, their country/region of residency, preferred language, and preferences like whether they want to subscribe to a newsletter or enable multifactor authentication.  
  
In Azure AD B2C, there are two ways to provide identity user experiences:  
  
User flows are predefined, built-in, configurable policies that we provide so user can create sign-up, sign-in, and policy editing experiences in minutes.  
  
Custom policies enable to create own user journeys for complex identity experience scenarios.  
  
We can customize our Azure AD B2C domain in the redirect URLs for Azure AD B2C. Custom domain allows us to create a seamless experience so that the pages that are shown blend seamlessly with the domain name of our application.  
  
Language customization in Azure AD B2C allows us to accommodate different languages to suit our customer needs. Microsoft provides the translations for 36 languages, but can also provide our own translations for any language. Even if experience is provided for only a single language, we can customize any text on the pages.  
  
Azure AD B2C ensures valid email addresses by requiring customers to verify them during the sign-up, and password reset flows. It also prevents malicious actors from using automated processes to generate fraudulent accounts in our applications.  
  
For applications, Azure AD B2C supports the OAuth 2.0, OpenID Connect, and SAML protocols for user journeys. Our application starts the user journey by issuing authentication requests to Azure AD B2C. The result of a request to Azure AD B2C is a security token, such as an ID token, access token, or SAML token. This security token defines the user's identity within the application.  
  
For external identities, Azure AD B2C supports federation with any OAuth 1.0, OAuth 2.0, OpenID Connect, and SAML identity providers.  
  
During sign up or password reset, our users must supply a password that meets complexity rules. By default, Azure AD B2C enforces a strong password policy. Azure AD B2C also provides configuration options for specifying the complexity requirements of the passwords our customers use.  
  
Azure AD B2C allows you to discover when people sign up or sign in to our app, where the users are located, and what browsers and operating systems they use.  
  
By integrating Azure Application Insights into Azure AD B2C custom policies, we can gain insight into how people sign up, sign in, reset their password or edit their profile. With such knowledge, we can make data-driven decisions for our upcoming development cycles.  
  
  
So  Open up the azure portal and login  
  
Make sure that we are using the directory that contains our subscription  
We can do this by clicking the Directory and subscription filter in the top menu and choosing the directory that contains it. This is a different directory than the one that will contain our Azure AD B2C tenant. Create the Active Directory B2C service  
  
Choose Create a resource in the top-left corner of the Azure portal.  
Search for and select Active Directory B2C  
Click create  
  
Choose Create a new Azure AD B2C Tenant, enter an organisation name and initial domain name, which is used in the tenant name, select the country (it can’t be changed later), and then click Create.  
Note: We might need to refresh our browser in order for our new B2C tenant subscription to appear in the azure portal.  
Each request that is sent to Azure AD B2C specifies a user flow. A user flow controls the behaviour of how our client application interacts with Azure AD B2C. It’s essentially the route a user takes.  
Example:  
When a user clicks the register button on a website homepage. This would be configured to trigger the sign up user flow. This user flow will then redirect the user to the sign up page provided by B2C. However if our user clicked on our login button, this would be configured to trigger a login user flow. Which then directs them to the login page  
  
Linking our subscription,we now need to carry out the following again:  
Choose Create a resource in the top-left corner of the Azure portal.  
Search for and select Active Directory B2C  
Click create  
This time however select  
‘Link an existing Azure AD B2C Tenant to my Azure Subscription’  
  
Register a web application using App Registrations:  
Select App registrations, and then select New registration.  
Enter a Name for the application. For example, webapp1.  
Under Supported account types, select Accounts in any identity provider or organizational directory (for authenticating users with user flows).  
Under Redirect URI, select Web, and then enter uri,Under Permissions, select the Grant admin consent to openid and offline\_access permissions check box.  
Select Register  
  
For a web application, we need to create an application secret. The client secret is also known as an application password. The secret will be used by our application to exchange an authorization code for an access token.  
The defining characteristic of the implicit grant is that tokens, such as ID and access tokens, are returned directly from Azure AD B2C to the application.  
  
For web apps,that request an ID token directly from the authorization endpoint, enable the implicit grant flow in the app registration.  
In the left menu, under Manage, select Authentication.  
Under Implicit grant, select both the Access tokens and ID tokens check boxes.  
Select Save.  
  
In our applications we may have user flows that enable users to sign up, sign in, or manage their profile. We can create multiple user flows of different types in our Azure Active Directory B2C (Azure AD B2C) tenant and use them in our applications as needed. User flows can be reused across applications.  
  
A user flow lets us determine how users interact with our application when they do things like sign in, sign up, edit a profile, or reset a password  
  
Azure AD B2C supports the OAuth 2.0 and OpenID Connect protocols, which makes use of tokens for authentication and secure access to resources. All tokens used in Azure AD B2C are JSON web tokens (JWTs) that contain assertions of information about the bearer and the subject of the token.  
  
The following tokens are used in communication with Azure AD B2C:  
ID token - A JWT that contains claims that we can use to identify users in our application. This token is securely sent in HTTP requests for communication between two components of the same application or service. We can use the claims in an ID token as you see fit.  
  
Access token - A JWT that contains claims that we can use to identify the granted permissions to our APIs. Access tokens are signed, but they aren't encrypted. Access tokens are used to provide access to APIs and resource servers. When our API receives an access token, it must validate the signature to prove that the token is authentic  
  
Refresh token - Refresh tokens are used to acquire new ID tokens and access tokens in an OAuth 2.0 flow. They provide our application with long-term access to resources on behalf of users without requiring interaction with those users. Refresh tokens are opaque to our application. They are issued by Azure AD B2C and can be inspected and interpreted only by Azure AD B2C  
  
A registered application receives tokens and communicates with Azure AD B2C by sending requests to these endpoints:  
  
https://<tenant-name>.[b2clogin.com/](http://b2clogin.com/)<tenant-name>.[onmicrosoft.com/](http://onmicrosoft.com/)<policy-name>/oauth2/v2.0/authorize  
https://<tenant-name>.[b2clogin.com/](http://b2clogin.com/)<tenant-name>.[onmicrosoft.com/](http://onmicrosoft.com/)<policy-name>/oauth2/v2.0/token  
Security tokens that our application receives from Azure AD B2C can come from the /authorize or /token endpoints.  
  
To validate a token, our application should check both the signature and claims of the token.A JWT contains three segments, a header, a body, and a signature. The signature segment can be used to validate the authenticity of the token so that it can be trusted by our application. Scopes provide a way to manage permissions to protected resources. When an access token is requested, the client application needs to specify the desired permissions in the scope parameter of the request.