Leveraging the FranceConnect Facade (FCF) with Azure AD B2C

Zzz

A developer guide to configure and test the facade

Version Beta - April 2023

This page is intentionally left blank.

Notice

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example 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.

A drawing of a face

Description automatically generated

Attribution 4.0 International (CC BY 4.0)

Microsoft and any contributors grant you a license to this document under the [Creative Commons Attribution 4.0 International Public License](https://creativecommons.org/licenses/by/4.0/legalcode), see the [LICENSE](https://github.com/microsoft/franceconnect-facade-dotnet-webapp-aspnetcore/blob/main/LICENSE-CODE) file, and grant you a license to any code in the repository under the MIT License, see the [LICENSE-CODE](https://github.com/microsoft/franceconnect-facade-dotnet-webapp-aspnetcore/blob/main/LICENSE-CODE) file.

Microsoft, Windows, Microsoft Azure and/or other Microsoft products and services referenced in the document may be either trademarks or registered trademarks of Microsoft in the United States and/or other countries. The license for this document does not grant you rights to use any Microsoft names, logos, or trademarks. Microsoft's general trademark guidelines can be found at <http://go.microsoft.com/fwlink/?LinkID=254653>.

Privacy information can be found at <https://privacy.microsoft.com/en-us/>

Microsoft and any contributors reserve all other rights, whether under their respective copyrights, patents, or trademarks, whether by implication, estoppel or otherwise.

Content

[About this guide 5](#_Toc133586358)

[You said Azure AD B2C, Quès aco!? 5](#_Toc133586359)

[Guide elements 9](#_Toc133586360)

[Guide assumptions 9](#_Toc133586361)

[Guide prerequisites 9](#_Toc133586362)

[Fulfill the prerequisites for your development environment 11](#_Toc133586363)

[Fulfill the prerequisites for your Azure testing environment 11](#_Toc133586364)

[Fulfill the prerequisites for your local environment 14](#_Toc133586365)

[Configure the FranceConnect Façade (FCF) for Azure AD B2C 16](#_Toc133586366)

[Update the configuration of the FranceConnect Façade 16](#_Toc133586367)

[Execute the FranceConnect Façade 17](#_Toc133586368)

[Redeploy the façade on the Internet 18](#_Toc133586369)

[Leverage B2C user flows for FranceConnect 19](#_Toc133586370)

[Define B2C user flows 19](#_Toc133586371)

[Test B2C user flows 32](#_Toc133586372)

[Troubleshoot B2C user flows 38](#_Toc133586373)

[Integrate with Microsoft Power Pages (Optional) 38](#_Toc133586374)

[Leverage B2C custom policies for FranceConnect 48](#_Toc133586375)

[Configure and upload the B2C custom policies 48](#_Toc133586376)

[Register the jwt.ms web application 52](#_Toc133586377)

[Test the B2C custom policies 52](#_Toc133586378)

[Troubleshoot your B2C custom policies 57](#_Toc133586379)

[As a conclusion 63](#_Toc133586380)

# About this guide

As the title of the guide (implicitly) suggests, this document takes you and guides you through all the steps required to leverage an already deployed instance of the FranceConnect, and integrate it with Azure Active Directory B2C (Azure AD B2C), i.e., the white-label Microsoft customer identity & access management (CIAM) solutions to define various identity user experiences (UX) : sign-up, sign-in, profile update, etc.

With this knowledge, you should be able to further adapt the provided resources if needed to accommodate your own specific situation and/or target a specific deployment in your production environment whatever it is.

*But wait a minute*

## You said Azure AD B2C, Quès aco!?

See [What is Azure Active Directory B2C?](https://docs.microsoft.com/en-us/azure/active-directory-b2c/overview) and [Technical and feature overview](https://docs.microsoft.com/en-us/azure/active-directory-b2c/technical-overview) to learn more.

Azure AD B2C provides business-to-customer (B2C) identity as a service (IDaaS). Your customers use their preferred social, enterprise, or local account identities to get single sign-on (SSO) access and single logout (SLO) capabilities to your web applications, mobile applications, and REST APIs, as well as for authorization such as access to REST API resources by authenticated users.

As such, Azure AD B2C is a white label, custom branded CIAM solution capable of supporting millions of users and billions of authentications per day. It takes care of the scaling and safety of the authentication platform, the capture of detailed analytics about sign-in behavior and sign-up conversion, the centralization/orchestration of the collection of user profile & preference information, the monitoring, and the automatically handling threats, such as denial-of-service (DoS), password spray, or brute force attacks.

Azure AD B2C uses standards-based authentication protocols notably including OpenID Connect (OIDC) by more than 95% of applications on the Internet. This allows Azure AD B2C to integrate with most modern applications and commercial off-the-shelf software.

The FranceConnect platform (FCP) makes no exceptions here, but the devil resides in the details, and this is the reason why the FranceConnect Façade is required for such a support, and the related integration path.

### Defining user journeys

See [User flows and custom policies in Azure Active Directory B2C](https://docs.microsoft.com/en-us/azure/active-directory-b2c/user-flow-overview).

In Azure AD B2C, you can model your user journeys and define the business logic behind that users follow to gain access to your applications and REST APIs. For example, you can determine the sequence of steps users follow when they sign up, sign in, edit a profile and/or preference information, reset a password, etc. After completing the sequence, the user acquires a security token and gains access to your application.

These identity user experiences can be provided in two ways, i.e., with:

1. **User flows** that are predefined, built-in, configurable policies for the most common identity tasks, and thus enable to create sign-up, sign-in, and policy editing experiences in minutes.
2. **Custom policies** that enable you to create your own specific user journeys for (more) complex identity experience scenarios with the full spectrum of capabilities provided by the so-called **identity experience framework (IEF) orchestration platform** of Azure AD B2C.

**The rest of this guide focuses on both the user flows and the custom policies and how to leverage them in accordance with the FranceConnect Façade (FCF).**

So let’s first consider what they are.

### A first look at the user flows

To set up the most common identity tasks in Azure AD B2C, the Azure portal includes several predefined and configurable policies called *user flows*.

You can configure user flow settings like these to control identity experience behaviors in your applications:

* Account types used for sign-in, such as social accounts like a Facebook, or local accounts that use an email address and password for sign-in
* Attributes to be collected from the user, such as first name, postal code, or country/region of residency.
* Azure AD multi-factor authentication (MFA).
* Customization of the user interface.
* Set of claims in a token that your application receives after the user completes the user flow
* Session management.
* ...and even more.

Most of the common identity scenarios for applications can be defined and implemented effectively with user flows. We recommend that you use the built-in user flows, unless you have complex user journey scenarios that require the full flexibility of custom policies. See next section.

### A first look at custom policies

See [Azure Active Directory B2C custom policy overview](https://docs.microsoft.com/en-us/azure/active-directory-b2c/custom-policy-overview).

By contrast, custom policies in Azure AD B2C are fully configurable and policy-driven. They orchestrate trust between entities in standard protocols such as the above-mentioned OpenID Connect (OIDC) protocol in the context of this guide, and a few non-standard ones, for example REST API-based system-to-system claims exchanges. The complete framework creates user-friendly, white-labeled experiences.

As such, custom policies are configuration files that define the behavior of your Azure AD B2C tenant user experience. Custom policies can be fully edited by an identity developer to complete many different crafted tasks.

#### User journeys

Custom policies give you the ability to construct user journeys: each user journey is defined by a policy. You can build as many or as few policies as you need to enable the best user experience for your organization.

User journeys allows you to define the business logic with path through which user will follow to gain access to your applications or REST APIs, a.k.a. relying parties (RPs). The user is taken through the user journey to retrieve the claims that are to be presented to your application or REST API.

A user journey is built with any combination of steps that constitute a sequence of **orchestration steps** (See below) to interact with internal and external parties.

For example:

* Federate with other identity providers (IdP), a.k.a. claims providers,
* First-and third-party multifactor authentication (MFA) challenges,
* Collect and/or validate any user input,
* Integrate with external systems using REST API communication.

This orchestration sequence must be followed through for a successful transaction. If any step fails, the transaction fails.

A relying party application or REST API calls a custom policy to execute such a specific user journey.

See [User journeys in Azure Active Directory B2C](https://docs.microsoft.com/en-us/azure/active-directory-b2c/userjourneys) and [Sub journeys in Azure Active Directory B2C](https://docs.microsoft.com/en-us/azure/active-directory-b2c/subjourneys).

#### Building blocks

A custom policy is defined by several XML-formatted configuration files, which fer to each other in a hierarchical chain through an inheritance model, where the child policy at any level can inherit from the parent policy and extend it by adding new elements.

In terms of building blocks, the XML elements define the claims schema, claims transformations, content definitions, claims providers, technical profiles, validation technical profiles, user journey orchestration steps, and other aspects of the identity experience:

* **Claims schema**. A claim provides temporary storage of data during a policy execution. It can store information about the user, such as first name, last name, or any other claim obtained from the user or other systems (claims exchanges).

The claim schema defines the claim types that can be referenced as part of the policy. The claims schema is the place where you declare your claim types. A claim type is similar to a variable in many programmatic languages. You can use the claim type to collect data from the user of your application, receive claims from social identity providers, send and receive data from a custom REST API, or store any internal data used by your custom policy. See [ClaimsSchema: Azure Active Directory B2C](https://docs.microsoft.com/en-us/azure/active-directory-b2c/claimsschema).

* **Claims transformations**. A claims transformation converts a given claim into another one. In the claims transformation, you specify the transform method, for example adding an item to a string collection or changing the case of a string. See [ClaimsTransformations - Azure Active Directory B2C](https://docs.microsoft.com/en-us/azure/active-directory-b2c/claimstransformations).
* **Content definitions**. A content definition contains URLs to HTML5 templates that can be used in a user journey. The HTML5 page URI is used for a specified user interface step. For example, the sign-in or sign-up, password reset, or error pages.

You can modify the look and feel and create new content definitions according to your needs. The identity experience framework orchestration platform of Azure AD B2C runs the related code in your customer's browser and uses a modern approach called Cross-Origin Resource Sharing (CORS). See [ContentDefinitions - Azure AD B2C](https://docs.microsoft.com/en-us/azure/active-directory-b2c/contentdefinitions).

* **Claims providers**. A claims provider is an interface to communicate with different types of parties via its technical profiles (see below). Every claims provider must have one or more technical profiles that determine the (metadata) endpoints, exact claims exchange definitions, secrets, keys, and certificates as needed, and the protocols needed to communicate with the claims provider.

Multiple technical profiles may be defined because the claims provider supports multiple protocols, various endpoints with different capabilities, or releases different claims at different assurance levels. It may be acceptable to release sensitive claims in one user journey, but not in another. See [ClaimsProviders - Azure Active Directory B2C](https://docs.microsoft.com/en-us/azure/active-directory-b2c/claimsproviders).

* **Technical profiles**. A technical profile provides an interface to communicate with different types of parties. A user journey combines calling technical profiles via orchestration steps (see below) to define your business logic.

All types of technical profiles share the same concept. You send input claims, run claims transformation, and communicate with the configured party. After the process is completed, the technical profile returns the output claims to claims bag. See [Technical profiles - Azure AD B2C](https://docs.microsoft.com/en-us/azure/active-directory-b2c/technicalprofiles).

* **Validation technical profiles**. A validation technical profile is an ordinary technical profile from any protocol, such as Azure AD or a REST API. The validation technical profile returns output claims, or returns 4xx HTTP status code, with an error message. See [Define a validation technical profile in a custom policy - Azure AD B2C](https://docs.microsoft.com/en-us/azure/active-directory-b2c/validation-technical-profile).
* **User journey orchestration steps**. An orchestration step references to a method that implements its intended purpose or functionality. This method is called a technical profile. When your user journey needs branching to better represent the business logic, the orchestration step references to **sub journey**. A sub journey contains its own set of orchestration steps.

A user must reach the last step to acquire a security token, but users may not need to travel through all of the orchestration steps. Orchestration steps can be conditionally executed based on **preconditions** defined in the orchestration step.

After an orchestration step completes, the IEF orchestration platform of Azure AD B2C stores the outputted claims in the **claims bag.** The claims in the claims bag can be utilized by any further orchestration steps in the user journey.

As such, When running through the orchestration steps, the IEF orchestration platform of Azure AD B2C sends and receives claims to and from internal and external parties and then sends a subset of these claims to your relying party (RP) application or REST API as part of the token.

At this point, you should hopefully have an understanding of what Azure AD B2C is and is and what custom policies are made of.

For a deeper look at the custom policies, and beyond the above-outlined articles from the [Azure AD B2C documentation on doc.microsoft.com](https://docs.microsoft.com/en-us/azure/active-directory-b2c/custom-policy-get-started?tabs=applications), you can consider the following series of documents that provides you with an end-to-end journey with the custom policies in Azure AD B2C, presenting in-depth the most common advanced identity scenarios:

* [Custom policies introduction](https://download.microsoft.com/download/3/6/1/36187D50-A693-4547-848A-176F17AE1213/Deep%20Dive%20on%20Azure%20AD%20B2C%20Custom%20Policies/Azure%20AD%20B2C%20Custom%20Policies%20-%20Introduction.pdf).
* [Leverage custom policies in your tenant](https://download.microsoft.com/download/3/6/1/36187D50-A693-4547-848A-176F17AE1213/Deep%20Dive%20on%20Azure%20AD%20B2C%20Custom%20Policies/Azure%20AD%20B2C%20Custom%20Policies%20-%20Leveraging%20Custom%20Policies%20for%20your%20Tenant.pdf).
* [Structure policies and manage keys](https://download.microsoft.com/download/3/6/1/36187D50-A693-4547-848A-176F17AE1213/Deep%20Dive%20on%20Azure%20AD%20B2C%20Custom%20Policies/Azure%20AD%20B2C%20Custom%20Policies%20-%20Structuring%20Policies%20and%20Managing%20Keys.pdf).
* [Bring your own identity and migrate users](https://download.microsoft.com/download/3/6/1/36187D50-A693-4547-848A-176F17AE1213/Deep%20Dive%20on%20Azure%20AD%20B2C%20Custom%20Policies/Azure%20AD%20B2C%20Custom%20Policies%20-%20Bring-your-own-identity%20and%20Migrating%20Users.pdf).
* [Troubleshoot policies and audit access](https://download.microsoft.com/download/3/6/1/36187D50-A693-4547-848A-176F17AE1213/Deep%20Dive%20on%20Azure%20AD%20B2C%20Custom%20Policies/Azure%20AD%20B2C%20Custom%20Policies%20-%20Troubleshooting%20Policies%20and%20Auditing.pdf).
* [Deep dive on custom policy schema](https://download.microsoft.com/download/3/6/1/36187D50-A693-4547-848A-176F17AE1213/Deep%20Dive%20on%20Azure%20AD%20B2C%20Custom%20Policies/Azure%20AD%20B2C%20Custom%20Policies%20-%20Deep%20Dive%20on%20Custom%20Policy%20Schema.pdf).

It includes how to implement and manage custom policies for these scenarios and how to diagnose them with the available tooling. It also provides an in-depth understanding of how custom policies work and details how to fine-tune them to accommodate your own specific requirements**.**

Let’s now consider how you can leverage this CIAM foundation with the FranceConnect Facade (FCF).

## Guide elements

For the sake of this guide, and considering the above, we provide the following elements:

* Hands-on instructionscontaining the most important steps and their outputs. These are meant to show you the core elements.
* Sample Azure AD B2C content definitions and custom policiesto implement a sign-in user journey (user flow or custom policy) with the FranceConnect button to authenticated against the FranceConnect platform via the FranceConnect Façade (FCF). You can access content definitions, i.e., the related .html and .css code for your user flows and custom policies, as well as **a series of (almost) ready-to-use pre-crafted custom policies** by downloading, cloning, or forking the following GitHub repository: [microsoft/franceconnect-facade-dotnet-webapi-aspnetcore (github.com)](https://aka.ms/franceconnect).

Let’s start with the assumptions and the prerequisites.

## Guide assumptions

This guide assumes that you already have an existing FranceConnect Façade (FCF) in place.

If needed, please refer to the [Getting Started with the FranceConnect Connect Façade (FCF)](https://github.com/microsoft/franceconnect-facade-dotnet-webapp-aspnetcore/blob/main/Documentation/Getting%20Started%20with%20FranceConnect%20Facade%20(FCF).docx) guide for information on how to prepare a suitable local development environment, deploy and set all the needed resources in the Azure Cloud for a test vs. production environment, as well as a portal application, etc., and ultimately configure and test the FranceConnect Façade (FCF) code,

## Guide prerequisites

We assume here that you have a Windows 10 local environment or above. For example, Windows 11 in our illustration.

### Install Git

Git is a [free and open source](https://git-scm.com/about/free-and-open-source) distributed version control system designed to handle everything from small to very large projects with speed and efficiency

You can download and install Git:

* Download the [Git for Windows](https://github.com/git-for-windows/git/releases/download/v2.25.0.windows.1/Git-2.25.0-64-bit.exe) and run it.
* In the pop-up window, click Install. Follow the instructions. When prompted, select Use the OpenSSL library.

Git will help us to clone the sample code project and related resources to complete this guide.

### Clone the repo using Git

To clone the FCF sample code’s project on your Windows 11 local machine, open a PowerShell console, and run the following commands:

PS C:\> cd c:\

PS C:\> git clone <https://github.com/microsoft/franceconnect-facade-dotnet-webapi-aspnetcore.git>

PS C:\> cd franceconnect-facade-dotnet-webapi-aspnetcore

The Azure AD B2C resources are located under the folder *B2C*.

The sample Azure AD B2C custom policies are forked from the repo available at <https://github.com/Azure-Samples/active-directory-verifiable-credentials/tree/main/B2C>.

**Repository organization**

**The repo contains a series of content definitions, i.e. .html and .css code for the UI of the Azure AD B2C user journeys, as well as a series of custom policies for your Azure AD B2C tenant to issue resp. verify VCs as part of your signup resp. sign-in policies.**

**For the sake of this complementary walkthrough, under the folder where you cloned the repo, e.g., *c:\****>* *franceconnect-facade-dotnet-webapi-aspnetcore* in our illustration, **we will mainly be interested in the *Source\fcf.AzureADB2C* directory, and the folders underneath:**

* *HtmlTemplate****:* contains a series of content definitions, i.e. .html and .css code, for use with the user flows and the provided custom policies.**
* *CustomPolicies****:* contains a series of (almost) ready-to-use pre-crafted custom policies for your Azure AD B2C tenant.**

**So, at this stage, you’re all set! It’s high time to move to the first module of this part.**

# Fulfill the prerequisites for your development environment

**In order to complete this walkthrough, we assume that you already fulfilled the prerequisites expressed in the first and second parts of this guide**.

The below prerequisites are indeed incremental. Let’s consider them in order.

## Fulfill the prerequisites for your Azure testing environment

### Azure AD B2C tenant

In addition to your Azure AD tenant already being leveraged as part of this guide, an additional Azure AD B2C tenant is required to complete this third part. We assume at the stage that you already have an Azure subscription in place of which you're an administrator.

#### Create your Azure B2C AD tenant

See [Tutorial - Create an Azure Active Directory B2C tenant](https://docs.microsoft.com/en-us/azure/active-directory-b2c/tutorial-create-tenant) for more details.

**To get started, you first need an Azure B2C AD tenant if you don’t have any for this walkthrough.**

**Unless noticed otherwise, all the required resources will be created in the FranceCentral region. If you want to setup the environment in another region, you will need to adapt the instructions accordingly.**

Usage charges for Azure AD B2C are billed to an Azure subscription. You need to explicitly link your Azure subscription.

As priorly described, Azure AD B2C offers two methods to define how users interact with your applications: i) through predefined B2C user flows or ii) through fully configurable B2C custom policies.

#### Be ready for B2C user flows

**Your Azure AD B2C tenant is ready by default for the user flows. No other actions are expected from you side to leverage them.**

#### Be ready for B2C custom policies

See [Tutorial - Create user flows and custom policies](https://docs.microsoft.com/en-us/azure/active-directory-b2c/tutorial-create-user-flows?pivots=b2c-custom-policy&tabs=applications#custom-policy-starter-pack) for more details.

**In contrast, you then need to make the Azure AD B2C tenant ready for the custom policies.**

As already introduced, custom policies are a set of XML-formatted configuration files you upload to your Azure AD B2C tenant to define your user journeys, and thus the orchestration steps, the related technical profiles, etc.

Interestingly enough, so-called starter packs are provided with several pre-built custom policies to get you going quickly. Each of these starter packs contains the smallest number of technical profiles and user journeys needed to achieve the scenarios described:

|  |  |
| --- | --- |
| Starter pack | Description |
| LocalAccounts | Enables the use of local accounts only |
| SocialAccounts | Enables the use of social (or federated) accounts only |
| SocialAndLocalAccounts | Enables the use of both local and social accounts |
| SocialAndLocalAccountsWithMFA | Enables social, local, and multi-factor authentication (MFA) options |

Each above starter pack contains:

|  |  |
| --- | --- |
| File | Description |
| Base file | Few modifications are required to the base. Example: *TrustFrameworkBase.xml* |
| Localization file | This file is where localization changes are made. Example: *TrustFrameworkLocalization.xml* |
| Extension file | This file is where most configuration changes are made. Example: *TrustFrameworkExtensions.xml* |
| Relying party files | Task-specific files called by your application. Examples: *SignUpOrSignin.xml, ProfileEdit.xml, PasswordReset.xml* |

To automate the entire process from [Tutorial - Create user flows and custom policies](https://docs.microsoft.com/en-us/azure/active-directory-b2c/tutorial-create-user-flows?pivots=b2c-custom-policy&tabs=applications#custom-policy-starter-pack), proceed with the following steps:

1. Open a browser session, navigate to the IEF Setup App at <https://aka.ms/iefsetup>, and follow the instructions.

Une image contenant texte

Description générée automatiquement

1. Under **Begin setup**:
2. Enter the domain name of your Azure AD B2C tenant. For example, litware369b2c in our illustration: litware369b2c.onmicrosoft.com.
3. Check **Enable JavaScript**.
4. Click **Deploy custom policy starter pack**. This will automatically deploy the above-mentioned SocialAndLocalAccountsWithMFA starter pack in your Azure AD B2C tenant, which will provide Sign Up and Sign In, Password Reset and Profile Edit journeys.
5. Sign-in with an account with admin privileges in your Azure AD B2C tenant - The account that was used to create the tenant has these by default –
6. Azure AD B2C will ask you to consent to the application having the ability to create objects in your Azure AD B2C tenant (applications, keys).
7. Once you consent, the IEF Setup App will check whether your tenant has all the objects named in the above tutorial.
8. If these objects, do not exist, the IEF Setup App will create them:

* Two applications: Azure AD B2C requires you to register two applications that it uses to sign up and sign in users with local accounts: i) **IdentityExperienceFramework**, a web API, and ii) **ProxyIdentityExperienceFramework**, a native app with delegated permission to the IdentityExperienceFramework app. Your users can sign up with an email address or username and a password to access your tenant-registered applications, which creates a "local account." Local accounts exist only in your Azure AD B2C tenant.
* Two service principals for these two applications.
* Two keys: Azure AD B2C requires you to create both i) a signing key named **B2C\_1A\_TokenSigningKeyContainer**, and ii) an encryption key named **B2C\_1A\_TokenEncryptionKeyContainer**.

1. The final screen will display the relevant application IDs needed in the IEF custom policies.
2. If the application did not exist already, the final screen will provide a URL link you should use to complete admin consent for the new applications to use each other [item 9 in the above-mentioned Tutorial](https://docs.microsoft.com/en-us/azure/active-directory-b2c/custom-policy-get-started?tabs=applications#register-the-proxyidentityexperienceframework-application).
3. Use the Enterprise Apps option of the Azure portal’s AAD blade to remove the B2CIEFSetup service principal from your Azure AD B2C tenant (optional).

For your convenience, the complete code of the IEF Setup App is shared on [GitHub](https://github.com/mrochon/b2ciefsetup). A special thanks to [Marius Rochon](https://github.com/mrochon).

## Fulfill the prerequisites for your local environment

Throughout this third part of this guide, we similarly assume you have a Windows 11 machine for the purpose of the illustration.

This said, please note that the same exact above prerequisites can be installed on a macOS or a Linux machine instead, and all the provided instructions apply in a similar way on such environments as well. For the sake of brevity, we neither cover these configurations in the rest of this guide nor the use of the Windows Subsystem for Linux 2 (WSL2) with a Linux distro (e.g., Ubuntu 20.04.4 LTS).

Let’s see how to install the additional prerequisites. Let’s consider first the edit of the XML-formatted files of your user journeys with the editor of your choice. For example, you can use Visual Studio Code, a lightweight cross-platform editor.

### Install Azure AD B2C extension for Visual Studio Code

The [Azure AD B2C extension for VS Code](https://github.com/azure-ad-b2c/vscode-extension) allows you navigate faster through you set of Azure AD B2C custom policies and create elements like technical profile, user journeys, claims definitions in your local environment and deploy them to your Azure tenant instead of using the Azure portal.

Please note that this extension is not supported by Microsoft, and is made available strictly as-is.

To install Azure AD B2C extension, proceed with the following steps:

1. Launch Visual Studio Code.
2. Navigate to **Extensions**.
3. In the Search bar, type “*Azure AD B2C*” and select the first extension.
4. Click **install**. Once completed, you should have a tab opening with the following view.

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

To start working with your custom policies, right from Visual Studio Code, you can open the folder that contains your XML-formatted files, e.g., ***c:\****>* *franceconnect-facade-dotnet-webapi-aspnetcore****Source\fcf.AzureADB2C\Policies*** in our illustration, or, open the XML-formatted files directly from any folder.

If the [XML](https://marketplace.visualstudio.com/items?itemName=redhat.vscode-xml), or the [XML Tools](https://marketplace.visualstudio.com/items?itemName=DotJoshJohnson.xml) extensions for Visual Studio Code are installed and activated, the XML extension handles the XML completion within the edited files, See [Troubleshoot custom policies and user flows in Azure Active Directory B2C](https://docs.microsoft.com/en-us/azure/active-directory-b2c/troubleshoot?pivots=b2c-custom-policy#troubleshoot-policy-validity).

In addition, you can also upload your custom policies from Visual Studio Code. This requires you to register a MS Graph delegated permissions application, see [Upload a custom polices directly from Azure AD B2C vscode extension](https://github.com/azure-ad-b2c/vscode-extension/blob/master/src/help/policy-upload.md).

### Install the IefPolicies PowerShell tools

To manage, i.e., create, extend, import, and export the XML-formatted files used for your own custom journeys in your Azure AD B2C tenant, you can leverage the so-called [IefPolicies PowerShell tools](https://github.com/mrochon/IEFPolicies).

The set of cmdlets is available in the PowerShell Gallery. Minimal PowerShell version is 7.0.

To install the IefPolicies cmdlets, proceed with the following steps:

1. Open a PowerShell 7 window with Administrative privileges.
2. Run the following command:

PS C:\> Set-PSRepository -Name "PSGallery" -InstallationPolicy Trusted

PS C:\> Install-Module -Name IefPolicies

**At this stage, you are ready to further configure your Azure testing environment to integrate The FranceConnect Façade (FCF) with Azure AD B2C.**

# Configure the FranceConnect Façade (FCF) for Azure AD B2C

**In order to complete this walkthrough, we assume that you already followed the** [Getting Started with the FranceConnect Façade (FCF)](https://github.com/microsoft/franceconnect-facade-dotnet-webapp-aspnetcore/blob/main/Documentation/Getting%20Started%20with%20FranceConnect%20Facade%20(FCF).docx) **guide, and thus deployed your own FranceConnect Façade (FCF) environment. This chapter considers such a foundation as a starting point.**

**If such a foundation isn’t yet in place, please refer to this guide to establish it in your Azure subscription.**

## Update the configuration of the FranceConnect Façade

See [Set redirect URLs to b2clogin.com for Azure Active Directory B2C](https://learn.microsoft.com/en-us/azure/active-directory-b2c/b2clogin).

**Whenever you set up an identity provider for use with your Azure AD B2C tenant, here the FranceConnect platform (FCP) through the FranceConnect Façade (FCF), you need to specify the** reply/**redirect URL of your Azure AD B2C tenant.**

**On your priorly deployed FranceConnect Façade (FCF), you should indicate this** reply/**redirect URL in lieu of the one of your Power Pages site.**

**There are two formats you can use for your b2clogin.com** reply/**redirect URLs, either:**

**https://<*your\_TenantName*>.b2clogin.com/<*your\_TenantID*>/oauth2/authresp**

**or**

**https://<*your\_TenantName*>.b2clogin.com/<*your\_TenantName*>.onmicrosoft.com/oauth2/authresp**

**where:**

* **<*your\_TenantName*> is the name of your B2C tenant. For example, in our illustration litware369b2c.**
* **<*your\_TenantID*> is the ID of your B2C tenant name. For example, in our illustration cbf80679-8d13-4ac7-a5de-a4c1d69bd5d2. This ID can be found in the exposed endpoints of your B2C tenant.**

**For example, in our illustration:**

<https://litware369b2c.b2clogin.com/cbf80679-8d13-4ac7-a5de-a4c1d69bd5d2/oauth2/authresp>

**resp.**

<https://litware369b2c.b2clogin.com/litware369b2c.onmicrosoft.com/oauth2/authresp>

**The first format provides the benefit of not having "Microsoft" appear anywhere in the URL by using the Tenant ID (a GUID) in place of your tenant domain name. Note, the *authresp* endpoint may NOT contain a user flow or a custom policy name.**

Regardless of the format you choose, the settings in the *appsettings.json* file of the FranceConnect Façade (FCF) must be modified accordingly to point to one of these redirect URLs of your B2C tenant **in lieu of the one of your Power Pages site**.

Proceed with the following steps:

1. Open the .sln solution with Visual Studio Community.
2. Open the *appsettings.json* file
3. In the FranceConnect section, specify the reply/redirect URL of your B2C tenant. For example, in our illustration:

"FranceConnect": {

…

portalredirecturi:https://litware369b2c.b2clogin.com/litware369b2c.onmicrosoft.com/oauth2/authresp"

…

}

1. Save the file.

**In a future release, the FranceConnect Façade (FCF) will be extended to simultaneously handle multiple** reply/redirect URLs to serve a series of clients, namely applications, APIs, and/or Azure AD B2C tenant. You will have to do a prior registration of the clients.

## Execute the FranceConnect Façade

At this stage, and depending on your current environment, you have to either rerun the façade locally or redeploy it for instance on Azure as covered in the [Getting Started with the FranceConnect Façade (FCF)](https://github.com/microsoft/franceconnect-facade-dotnet-webapp-aspnetcore/blob/main/Documentation/Getting%20Started%20with%20FranceConnect%20Facade%20(FCF).docx) **guide**.

### Rerun the FranceConnect Façade (FCF) locally

Please refer to the section Run the façade code locally in the [Getting Started with the FranceConnect Façade (FCF)](https://github.com/microsoft/franceconnect-facade-dotnet-webapp-aspnetcore/blob/main/Documentation/Getting%20Started%20with%20FranceConnect%20Facade%20(FCF).docx) **guide**.

#### Run ngrok for a local execution

**if you use**[ngrok](https://ngrok.com/)**, a tip to leverage here is to start it and leave it running so that your publicly available hostname for the FranceConnect Façade (FCF) will NOT change for the rest of this document.**

**This will allow you to specify in the provided custom policy XML-formatted files a publicly available fully qualified name on the Internet, and this name will remain valid after uploading these files in your B2C tenant.**

Proceed with the following steps:

1. Open a PowerShell 7 window.
2. Start a tunnel with ngrok.

PS C:\> ngrok.exe http 4242 --host-header=localhost:4242

More specifically, this command runs ngrok to set up a URL on 4242 where the FranceConnect Facade’s endpoints will be later listening on, and make it publicly available on the Internet. In other words, it will give the FranceConnect Facade and Internet address like:

<https://7516-2a01-110-8012-1013-7925-1440-7ba6-d086.ngrok.io>

1. Leave the PowerShell 7 window opened for the rest of this part.

Redeploy the FranceConnect Façade (FCF)

## Redeploy the façade on the Internet

Please refer to the section Deploy the façade on the Internet in the [Getting Started with the FranceConnect Façade (FCF)](https://github.com/microsoft/franceconnect-facade-dotnet-webapp-aspnetcore/blob/main/Documentation/Getting%20Started%20with%20FranceConnect%20Facade%20(FCF).docx) **guide**.

You are now ready to further configure your B2C tenant.

# Leverage B2C user flows for FranceConnect

**In order to complete this walkthrough, we assume that you already completed the chapters** Fulfill the prerequisites for your local environment **and** Configure the FranceConnect Façade (FCF) for Azure AD B2C **above, and thus deployed and configured your own FCF environment to operate in conjunction with Azure AD B2C.**

**This chapter considers such a foundation as a starting point.**

**With that, let’s consider the B2C user flows. As already introduced, these user flows are predefined, built-in, configurable policies that we provide so you can create sign-up, sign-in, and policy editing experiences in minutes.**

**In this chapter, you will now:**

1. Define B2C user flows**.**
2. Test B2C user flows**.**
3. Troubleshoot B2C user flows**.**
4. Integrate with Microsoft Power Pages (Optional)**.**

## Define B2C user flows

### Create the FranceConnect identity provider

#### Add a generic OpenID Connect (OIDC) identity provider

See [Set up sign-up and sign-in with OpenID Connect](https://learn.microsoft.com/en-us/azure/active-directory-b2c/identity-provider-generic-openid-connect?pivots=b2c-user-flow).

Proceed with the following steps:

1. Sign in to the [Azure portal](https://portal.azure.com) as the global administrator of your Azure AD B2C tenant.
2. Make sure you're using the directory that contains your Azure AD B2C tenant. Select the **Directories + subscriptions** icon in the portal toolbar.
3. On the **Portal settings | Directories + subscriptions** page, find your Azure AD B2C directory in the **Directory name** list, and then select **Switch**.
4. Choose **All services** in the top-left corner of the Azure portal, search for and select **Azure AD B2C**.
5. Select **Identity providers**, and then select **New OpenID Connect provider**. A Configure custom IDP blade opens up.

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

1. Fill in the blade with the following values:

See [Web sign in with OpenID Connect](https://learn.microsoft.com/en-us/azure/active-directory-b2c/openid-connect) and [Authorization code flow](https://learn.microsoft.com/en-us/azure/active-directory-b2c/authorization-code-flow).

* 1. In Name, enter a **name for the deployed FranceConnect façade (FCF)**. For example, in our illustration, enter FranceConnect.
  2. Most OIDC identity provider describes a metadata document that contains most of the information required to perform sign-in. The metadata document includes information such as the URLs to use and the location of the service's public signing keys. The OpenID Connect metadata document is always located at an endpoint that ends in  .well-known/openid-configuration. The FranceConnect façade (FCF) makes no exception to that.

In **Metadata url**, specify the URL of the OpenID Connect metadata document of the deployed FranceConnect facade (FCF).

* 1. To allow users to sign in, the FranceConnect platform (FCP) requires you to obtain a prior accreditation for your application in their service. Once obtained for the integration vs. production environment, your application has an ID that is referred to as the **client ID** and a **client secret**.

The client secret is NOT optional. You must provide a client secret as the Response type is code, which uses the secret to exchange the code for the token. See step 4 below.

For example, the FranceConnect dev account through the available so-called “integration key for public use”, has the following values:

* Client ID: 211286433e39cce01db448d80181bdfd005554b19cd51b3fe7943f6b3b86ab6e.
* Client secret: 2791a731e6a59f56b6b4dd0d08c9b1f593b5f3658b9fd731cb24248e2669af4b.

In Client ID to add the client ID resp. in Client secret to client secret, copy these values from the identity provider and enter them into the corresponding fields.

* 1. Scope defines the information and permissions you are looking to gather from the FranceConnect platform (FCP) based on your submitted accreditation, for example, in our illustration openid profile. In order to receive an ID token from the FranceConnect platform (FCP) thanks to the FranceConnect façade (FCF), the openid scope MUST be specified.

Without the ID token, users are not able to sign in to Azure AD B2C using the above FranceConnect identity provider. Other scopes can be appended separated by space. Refer to the [FranceConnect platform's documentation](https://partenaires.franceconnect.gouv.fr/fcp/fournisseur-service) (in French) to see what other scopes may be available.

In **Scope**, enter the scopes from the FranceConnect platform (FCP). For example, in our illustration:

openid profile birth email

* 1. The response type describes what kind of information is sent back in the initial call to the authorization\_endpoint of the FranceConnect platform (FCP) through the FranceConnect façade (FCF). The response type that can be used with the façade is code.

As per the [authorization code flow](https://openid.net/specs/openid-connect-core-1_0.html#CodeFlowAuth), a code will be returned back to Azure AD B2C. Azure AD B2C proceeds to call the token\_endpoint of the FranceConnect façade (FCF) to in turn exchange the code for an ID token against the FranceConnect platform (FCP), query its UserInfo endpoint, and ultimately issue a suitable ID token.

In **Response type**, select code.

* 1. The response mode defines the method that should be used to send the data back from the custom identity provider to Azure AD B2C. The following response mode that can be used with the façade is query: The code or token is returned as a query parameter. form\_post: cannot be used with the current implementation of the facade.

In Response, select query.

* 1. A [domain hint](https://learn.microsoft.com/en-us/azure/active-directory-b2c/direct-signin) can be used to skip directly to the sign in page of the specified identity provider, instead of having the user make a selection among the list of available identity providers.

As in our case the so-called FranceConnect button must initiate the identity dance against the FranceConnect platform (FCP), leave Domain hint empty.

* 1. After a successful authentication on the FranceConnect platform (FCP), and when the FranceConnect façade (FCF) sends an ID token back to Azure AD B2C, Azure AD B2C needs to be able to map the claims from the received token to the claims that Azure AD B2C recognizes and uses. For each of the following mappings, refer to the detailed specifications, a.k.a. the [FranceConnect Façade (FCF) Addendum to the technical-functional specifications](https://github.com/microsoft/franceconnect-facade-dotnet-webapp-aspnetcore/blob/main/Specifications/FranceConnect%20Facade%20(FCF)%20Specifications%20-%20Addendum.docx) document, of the FranceConnect façade (FCP) to understand the claims that are returned back in the ID token:
* **In User ID**, specify the claim that provides the unique identifier for the signed-in user. Enter “*email*”.
* **In Display name**, specify the claim that provides the display name or full name for the user. Enter “*family\_name*”.
* **In Given name**, specify the claim that provides the first name of the user. Enter “*given\_name*”.
* **In Surname**, specify the claim that provides the last name of the user. Enter “*family\_name*”.
* **In Email, specify** the claim that provides the email address of the user. Enter “*email*”.

1. Click Save.

### Register a web application

See [Tutorial: Register a web application in Azure Active Directory B2C](https://learn.microsoft.com/en-us/azure/active-directory-b2c/tutorial-register-applications?tabs=app-reg-ga).

Before your applications and/or services can interact with Azure AD B2C, and in turn with the FranceConnect Façade (FCF) and eventually the FranceConnect platform (FCP), they must be registered in a tenant that you manage. This section covers how to register a web application using the Azure portal.

For the sake of our illustration, we will leverage the <http://jwt.ms> web application, a Microsoft-owned web application that displays the decoded contents of a token at the end of the identity dance - the contents of the token never leave your browser -.

#### Register the jwt.ms web application

To register this web application, proceed with the following steps:

1. Sign in to the Azure portal at <https://portal.azure.com>.
2. Make sure you're using the directory that contains your Azure AD B2C tenant. Select the **Directories + subscriptions** icon in the portal toolbar.
3. On the **Portal settings | Directories + subscriptions** page, find your Azure AD B2C directory in the **Directory name** list, and then select **Switch**.
4. In the Azure portal, search for and select **Azure AD B2C**.
5. Select **App registrations**, and then select **New registration**. A Register an application page opens up.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

1. Fill in the page with the following values:
   1. Under Name, enter a **name** for the application. For example, in our illustration “jwt.ms website”.
   2. Under **Supported account types**, select **Accounts in any identity provider or organizational directory (for authenticating users with user flows)**.
   3. Under **Redirect URI**, select **Web**, and then enter “*https://jwt.ms*” in the URL text box.

The redirect URI is the endpoint to which the user is sent by the authorization server (Azure AD B2C, in this case) after completing its interaction with the user, and to which an access token or authorization code is sent upon successful authorization. In a production application, it's typically a publicly accessible endpoint where your app is running, such as <https://www.litware369.com/auth-response>. For testing purposes here, you can set it to <https://jwt.ms>.

* 1. Under **Permissions**, select the Grant admin consent to openid and offline\_access permissions check box.

1. Click **Register**.

### Create user flows

#### Create a sign-up and sign-in user flow

The sign-up and sign-in user flow handles both sign-up and sign-in experiences with a single configuration. Users of your testing application, see sections Register a web application above and Integrate with Microsoft Power Pages (Optional) below, are led down the right path depending on the context.

Proceed with the following steps:

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. Select the **Directories + subscriptions** icon in the portal toolbar.
3. On the **Portal settings | Directories + subscriptions** page, find your Azure AD B2C directory in the **Directory name** list, and then select **Switch**.
4. In the Azure portal, search for and select **Azure AD B2C**.
5. Under **Policies**, select **User flows**, and then select + **New user flow. A Create a user flow page opens up.**

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

1. **Select Sign up and sign in user flow. A Version section shows up.**

**Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement**

1. **Under Version, select Recommended, and then select Create. See** [User flow versions in Azure Active Directory B2C](https://learn.microsoft.com/en-us/azure/active-directory-b2c/user-flow-versions) **and** [Legacy user flow versions in Azure Active Directory B2C](https://learn.microsoft.com/en-us/azure/active-directory-b2c/user-flow-versions-legacy).

**A Create page opens up.**

**Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement**

1. Fill in the page with the following values:
   1. In 1. Name, enter a name for the user flow. For example, in our illustration *FCF\_SignUpSignIn*.
   2. In 2. Identity providers, select FranceConnect. See section Create the FranceConnect identity provider above.
   3. In 3. Multifactor authentication, keep the default settings.
   4. In 4. Conditional access, keep the default settings.
   5. For 5. User attributes and token claims, choose the claims and attributes that you want to collect and send from the user during sign-up. Click Show more… A Create blade pops up.

Une image contenant texte, capture d’écran, nombre, Parallèle

Description générée automatiquement

In the Return column, check the boxes for Display Name, Email Addresses, Given Name and Surname, and click OK.

1. Select Create to add the user flow. A prefix of *B2C\_1\_* is automatically prepended to the name.

The user flow *B2C\_1\_*FCF\_*SignUpSignIn* is now listed.

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

### Customize the UI with custom html templates

See [Customize the user interface with HTML templates](https://docs.microsoft.com/en-us/azure/active-directory-b2c/customize-ui-with-html?pivots=b2c-user-flow#2-create-an-azure-blob-storage-account).

Instead of creating your custom page content from scratch, you can customize Azure AD B2C's default page content. You can download and use these files as a starting point for creating your own custom pages.

For the sake of our illustration, we will use the following pre-configured page with the addition of the so-called FranceConnect button:

|  |  |  |
| --- | --- | --- |
| Page | Description | Template |
| Unified sign-up or sign-in (unified.html) | This page handles the user sign-up and sign-in process. Users can use public/enterprise identity providers, social identity providers such as Facebook, Microsoft account, or local accounts. | [Ocean Blue](https://login.microsoftonline.com/static/tenant/templates/AzureBlue/unified.cshtml)[[1]](#footnote-2) |

See *HtmlTemplate* folder **in the *Source\fcf.AzureADB2C* directory where you cloned the repo, e.g., in our illustration:**

***c:\****>* *franceconnect-facade-dotnet-webapi-aspnetcore* ***Source\fcf.AzureADB2C\HtmlTemplate***

#### Set up an Azure Blob storage account for storing the custom .html files

##### Create a storage account

See [Create a storage account](https://docs.microsoft.com/en-us/azure/storage/common/storage-account-create?tabs=azure-powershell).

[Azure Blob Storage](https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction) is an object storage solution for the cloud. Azure AD B2C uses Azure Blob Storage to store the custom html templates, a.k.a. content definition files, when the service orchestrate a user flow.

To create a Blob Storage account, proceed with the following steps:

1. Open a PowerShell 7 session and authenticate per section Install **Azure Az PowerShell module** in the first part of the guide.
2. Create a resource group:

PS C:\> New-AzResourceGroup -Name <*yourResourceGroup*> -Location "France Central"

Replace the placeholder value in brackets with your own value. For example in our illustration:

PS C:\> New-AzResourceGroup -Name "B2C\_Configuration" -Location "France Central"

1. Next create the Blob Storage account:

PS C:\> New-AzStorageAccount -ResourceGroupName <*yourResourceGroup*> –AccountName <*yourStorageAccount*> –Location "France Central" -SkuName Standard\_LRS

Replace the placeholder values in brackets with your own values and *France Central* by the Azure region of your choice.

For example in our illustration:

PS C:\> New-AzStorageAccount -ResourceGroupName "B2C\_Configuration" –AccountName "litware369b2cstorage" –Location "France Central" -SkuName Standard\_LRS

After you've created the storage account, create a public container.

##### Create a public storage container

To create a public container in the above storage account, proceed with the following steps:

1. From the previous browser session, in the Azure portal, search for “*storage accounts”*, and then click Storage accounts.
2. Click the blob storage account you created previously.
3. In the left pane for the storage account, scroll to the **Data storage** section, and select **Containers**.
4. Select **+ Container**. A **New container** blade pops up.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

* 1. In Name, type a name for your new container. The container name must be lowercase, must start with a letter or number, and can include only letters, numbers, and the dash (-) character. For example, “*customsignin*” in our illustration.
  2. Set **Public access level** to **Blob (anonymous read access for blobs only)**.
  3. Select **Create**.

#### Upload the custom sign-in html template

Proceed with the following steps:

1. From the Azure portal, select the newly created container.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

1. Select **Upload**. An **Upload blob** blade pops up.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

* 1. Launch a File Explorer, and from the location where you cloned the repo, navigate to the *HtmlTemplate* under the *Source\****fcf.AzureADB2C*** directory, e.g., in our illustration:

*c:\franceconnect-facade-dotnet-webapi-aspnetcore\****Source\fcf.AzureADB2C\HtmlTemplate***

* 1. In the subfolder *src\images*, add the FranceConnect buttons in the .svg format available in the [button pack FranceConnect](https://partenaires.franceconnect.gouv.fr/fcp/fournisseur-service#buttons).

A screenshot of a computer

Description automatically generated with medium confidence

* 1. Select the *unified.html* file and the *src* folder, and drag and drop them.
  2. Click **Upload**.

1. Right click on the *unified.html* blob that you uploaded and click on View/edit.
2. In the text editor that appeared search for all occurrences of *https://<yourstorageaccount>.blob.core.windows.net/* and replace *<yourstorageaccount>* with the name of the Storage account you just created.

For example, in our illustration: <https://litware369b2cstorage.blob.core.windows.net/>

1. Save the file, and select the Overview tab.
2. To the right of the **URL** text box, select the **Copy to clipboard** icon to copy the URL to your clipboard. For example, in our illustration:

<https://litware369b2cstorage.blob.core.windows.net/customsignin/unified.html>

1. Open a browser session and navigate to this URL you copied to verify the blob you uploaded is accessible. If it is inaccessible, for example if you encounter a ResourceNotFound error, make sure the container access type is set to blob. The blob you uploaded is inaccessible, and you might encounter a ResourceNotFound error.

#### Configure CORS

You now need to configure your blob storage account for [Cross-Origin Resource Sharing (CORS)](https://en.wikipedia.org/wiki/Cross-origin_resource_sharing). CORS is an HTTP feature that enables a web application running under one domain to access resources in another domain. Web browsers implement a security restriction known as same-origin policy that prevents a web page from calling APIs in a different domain. CORS provides a secure way to allow one domain (the origin domain) to call APIs in another domain.

You can set CORS rules individually for each of the storage services, here the blob storage account. Once you set the CORS rules for your blob storage account, then a properly authenticated request made against the service from a different domain will be evaluated to determine whether it is allowed according to the rules you have specified.

Perform the following steps:

1. From your storage account in the Azure portal, in the left menu, under **Settings**, select **Resource sharing (CORS)**.

Une image contenant texte

Description générée automatiquement

* 1. For **Allowed origins**, enter “*https://<yourTenantName>.b2clogin.com*”. Replace <*yourTenantName*>with the name of your Azure AD B2C tenant. For example, https://litware369b2c.b2clogin.com in our illustration. Use all lowercase letters when entering your tenant name.
  2. For **Allowed Methods**, select both **GET** and **OPTIONS**.
  3. For **Allowed Headers**, enter an asterisk (\*).
  4. For **Exposed Headers**, enter an asterisk (\*).
  5. For **Max age**, enter “*200*”.

1. At the top of the page, select **Save**.

#### Test CORS

So let’s validate at this stage that you're ready.

Perform the following steps:

1. Repeat the above configure CORS steps. For **Allowed origins**, enter “*https://www.test-cors.org*”.
2. Navigate to <https://www.test-cors.org>.
3. In **Remote URL**, paste the above URL of your custom html file. For example, in our illustration:

<https://litware369b2cstorage.blob.core.windows.net/customsignin/unified.html>

Une image contenant texte

Description générée automatiquement

1. Click **Send Request**. The result should be XHR status: 200.

If you receive an error, make sure that your CORS settings are correct. You might also need to clear your browser cache or open an in-private browsing session by pressing Ctrl+Shift+P.

#### Update the user flow

Proceed with the following steps:

1. Make sure you're using the directory that contains your Azure AD B2C tenant:
   1. Select the **Directories + subscriptions** icon in the portal toolbar.
   2. On the **Portal settings | Directories + subscriptions** page, find your Azure AD B2C directory in the directory name list, and then select **Switch**.
2. In the Azure portal, search for and select ****Azure AD B2C****.
3. In the left-hand menu, select User flows, and then select the user flow *B2C\_1\_*FCF\_*SignUpSignIn* you previously created. See section Create user flows above.
4. Under Settings, select Properties.
5. Toggle on Enable JavaScript enforcing page layout.

A picture containing text, font, screenshot

Description automatically generated

1. Under Token compatibility settings, set Claim representing user flow to tfp and use an Issuer (iss) claim that includes tfp. For more details, see [Configure tokens](https://learn.microsoft.com/en-us/azure/active-directory-b2c/configure-tokens?pivots=b2c-custom-policy#token-compatibility-settings).
2. At the top of the page, select Save.
3. Under Customize, select **Page layouts**.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

1. Under **Unified sign-up or sign-in page**, select **Yes** for **Use custom page content**.
2. In **Custom page URI**, enter the URI for the unified.html file that you recorded earlier. For example, in our illustration:

<https://litware369b2cstorage.blob.core.windows.net/customsignin/unified.html>

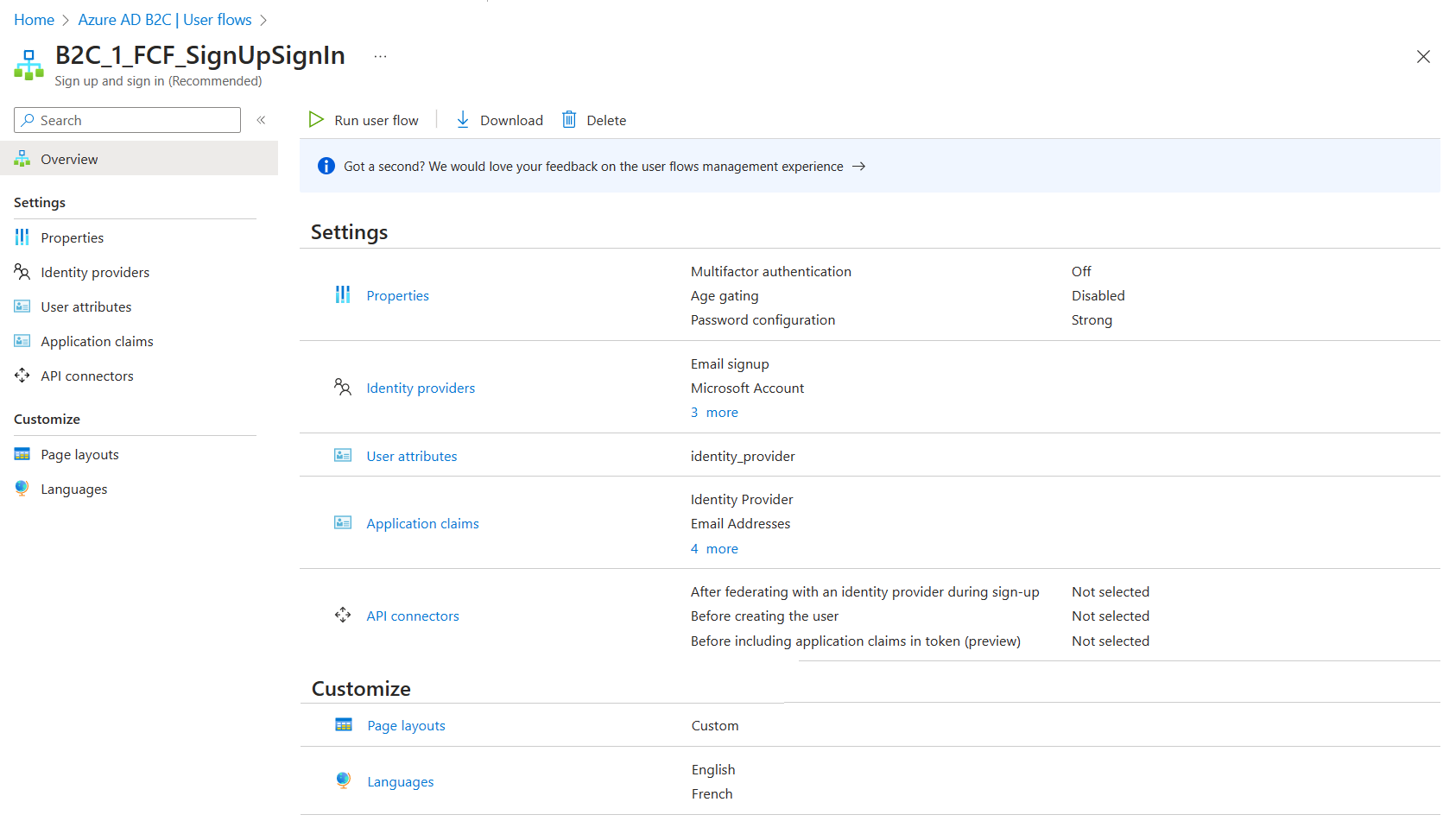
1. **Save again**.

## Test B2C user flows

Please note that Azure AD B2C has built-in support in the Azure portal for testing your B2C user flow via the **Run user flow** functionality,

Proceed with the following steps:

1. In your Azure AD B2C tenant, select User flows and select the above-defined user flow *B2C\_1\_*FCF\_*SignUpSignIn*. A B2C\_1\_FCF\_SignUpSignIn page opens up.



1. At the top of the page, select Run user flow. A Run user flow opens up.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

1. For Application, select the web application that you previously registered. See section Register a web application above. For example, in our illustration jwt.ms website. The Reply URL should show <https://jwt.ms>.
2. Click Run user flow. A sign-up or sign-in page opens up.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

For clarity, please note that we’ve left visible the FranceConnect in the HTML fragment inserted by Azure AD B2C to illustrate the design orientations outlined in the [FranceConnect Façade (FCF) Addendum to the technical-functional specifications](https://github.com/microsoft/franceconnect-facade-dotnet-webapp-aspnetcore/blob/main/Specifications/FranceConnect%20Facade%20(FCF)%20Specifications%20-%20Addendum.docx) document. This button at the bottom of the dialog can be hidden via a CSS configuration.



1. From the sign-up or sign-in page, click the so-called FranceConnect button at the top. You are redirected to the FranceConnect Façade (FCF), which in turn redirects you to the FranceConnect Platform (FCP) login page.

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

1. Select one of the test identity providers. Interestingly enough, the integration environment of the FranceConnect platform (FCP) allows you to use a "Demo" identity provider (IdP) for testing purposes only. Select Démonstration eIDAS faible. You are redirected to the selected IdP.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

1. At the selected identity provider, enter you credentials. The multiple testing user accounts available are listed here:

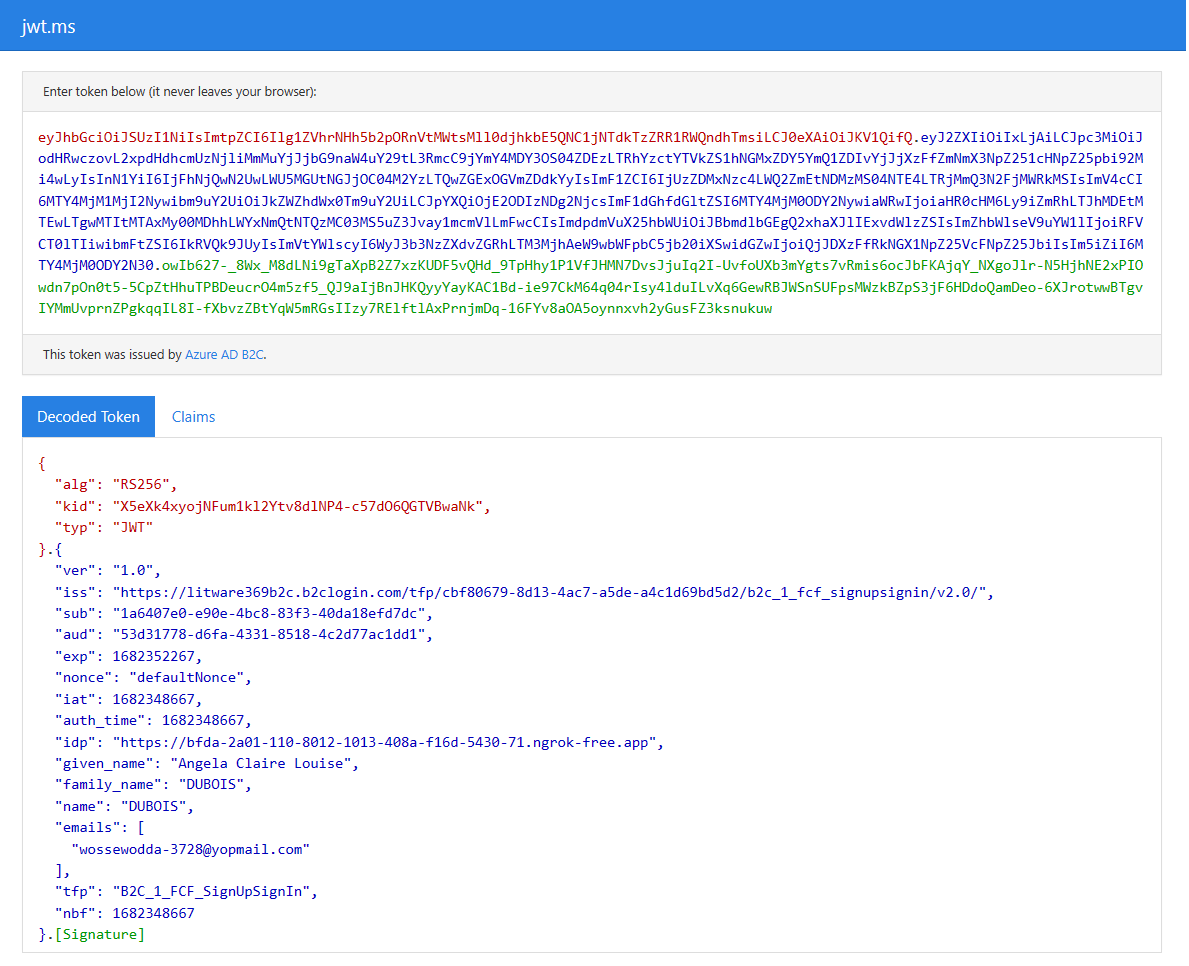
<https://github.com/france-connect/identity-provider-example/blob/master/database.csv>.

1. Click Valider. You are now redirected back to the FranceConnect platform.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

1. Give your consent by clicking Continuer sur <*Your\_Accredited\_Project*>. For example, Continuer sur Microsoft FRANCE - 49063 in our illustration. You are now redirected back to your Power Pages site through the FranceConnect Facade.
2. If the sign-in process is successful, your browser is redirected to <https://jwt.ms>, which displays the contents of the token issued by the FranceConnect façade (FCF) (based on the one received from the FranceConnect platform (FCP)), and ultimately returned by Azure AD B2C.



## Troubleshoot B2C user flows

For troubleshooting a specific B2C user flow in development environment, and as a starting point, you can turn on network traffic capturing on Microsoft Edge, or observe the traffic via a tool like the [Telerik Fiddler Classic](http://www.telerik.com/fiddler) application. Acting as a proxy server, Fiddler allows you to watch HTTP/HTTPS traffic, set breakpoints, and "fiddle" with incoming or outgoing data.

The article [Decrypt HTTPS traffic - Fiddler Classic](https://docs.telerik.com/fiddler/configure-fiddler/tasks/decrypthttps) provides you with guidance on how to configure the tool for debugging HTTPS traffic.

See also [Troubleshoot custom policies and user flows in Azure Active Directory B2C](https://learn.microsoft.com/en-us/azure/active-directory-b2c/troubleshoot?pivots=b2c-custom-policy#troubleshooting-with-application-insights).

## Integrate with Microsoft Power Pages (Optional)

For more details, see [Overview of authentication in Power Pages](https://learn.microsoft.com/en-us/power-pages/security/authentication/) and [Configure the Azure Active Directory B2C provider](https://learn.microsoft.com/en-us/power-pages/security/authentication/azure-ad-b2c-provider).

For the sake of this illustration, and the related step-by-step instructions, we will reuse the Power Pages application that was setup as part the walkthrough provided in the [Getting Started with the FranceConnect Connect Façade (FCF)](https://github.com/microsoft/franceconnect-facade-dotnet-webapp-aspnetcore/blob/main/Documentation/Getting%20Started%20with%20FranceConnect%20Facade%20(FCF).docx) guide. If needed, please refer to this guide for information on how to deploy and configure the Power Pages application.

### Register your Power Pages site

See [Tutorial: Register a web application in Azure Active Directory B2C](https://learn.microsoft.com/en-us/azure/active-directory-b2c/tutorial-register-applications?tabs=app-reg-ga).

To register the Power Pages application you created earlier, proceed with the following steps:

1. Make sure you're using the directory that contains your Azure AD B2C tenant:
   1. Select the **Directories + subscriptions** icon in the portal toolbar.
   2. On the **Portal settings | Directories + subscriptions** page, find your Azure AD B2C directory in the directory name list, and then select **Switch**.
2. In the Azure portal, search for and select **Azure AD B2C**.
3. Select **App registrations**, and then select **New registration**. A Register an application page opens up.
4. Fill in the page with the following values:
   1. Under Name, enter a **name** for the Power Pages application. For example, in our illustration “demofacade”.
   2. Under **Supported account types**, select **Accounts in any identity provider or organizational directory (for authenticating users with user flows)**.
   3. Under Redirect URI, select Web (if it isn't selected already).
   4. In Redirect URI, enter the Reply URL for your Power Pages site:

https://<*your\_AppName*>.powerappsportals.com/signin-aad-b2c\_1

where <*your\_AppName*> is the name of your app. For example, in our illustration:

<https://demofacade.powerappsportals.com/signin-aad-b2c_1>

Be sure to use this value when you configure the Redirect URL in your settings while configuring the Azure AD B2C provider. See next section below.

* 1. Under **Permissions**, select the Grant admin consent to openid and offline\_access permissions check box.

1. Select **Register**.

Take a note of the client ID. For example, in our illustration: 53d31778-d6fa-4331-8518-4c2d77ac1dd1.

You now need to enable the ID token implicit grant for this newly registered Power Pages site.

#### Enable the ID token implicit grant

With the user flow your created earlier, you need to enable the implicit grant flow in the app registration for your Power Pages site.

Proceed with the following steps:

1. In the left menu, under **Manage**, select **Authentication**.
2. Under **Implicit grant and hybrid flows**, select both the **Access tokens (used for implicit flows)** and **ID tokens (used for implicit and hybrid flows)** check boxes.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

1. Select **Save**.

### Configure Azure AD B2C on your Power Pages site

For more details, see [Overview of authentication in Power Pages](https://learn.microsoft.com/en-us/power-pages/security/authentication/) and [Configure the Azure Active Directory B2C provider](https://learn.microsoft.com/en-us/power-pages/security/authentication/azure-ad-b2c-provider).

To configure your existing Power Pages application, proceed with the following steps:

1. Open a browser session and navigate to Microsoft Power pages at <https://make.powerpages.microsoft.com/>.
2. Sign in to Power Pages with your credentials. You are redirected to the [Power Pages home page](https://make.powerpages.microsoft.com/).
3. From the right-upper corner, select the Environment drop-down to verify, or choose the Power Pages environment for your portal.
4. Select your portal, click the ellipsis (…), and select Details from the context menu.
5. If prompted with a dialog You are editing a live, public site, click Edit the site.
6. On the Set up left panel, select Identity providers under Authentication.
7. Select Configure for Azure Active Directory B2C. The Configure identity provider page opens up.
8. In Select a provider, set the following values:
   1. In Select login provider: leave Azure Actoive Directory B2C selected as the login provider. In Protocol, OpenID Connect. should be selected and grayed.
   2. In Provider name, type a name. For example, “*Azure AD B2C*”.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* 1. And click Next.

Une image contenant texte, capture d’écran, nombre, Parallèle

Description générée automatiquement

1. Under Reply URL, make a note of the Redirect URL that will be used by your Power Pages site to redirect users to the portal after the authentication succeeds with the FranceConnect Platform through the FranceConnect Facade. For example, in our illustration:

[https://demofacade.powerappsportals.com/signin-aad-b2c\_1](https://demofacade.powerappsportals.com/signin-aad-b2c_1%20)

Please note that if your Power Pages site uses a custom domain name for a production environment, you might have a different URL than the one provided here.

1. Under 2. Configuration site settings, enter the site settings for portal configuration:
   1. For Authority, enter the issuer URL defined in the metadata of the sign-up and sign-in policy user flow. To obtain this URL:
      1. Open the sign-up and sign-in user flow you created earlier, then go to the Azure AD B2C tenant on the [Azure portal](https://portal.azure.com/).
      2. Select Run user flow.
      3. Select the OpenID configuration URL to open in a new browser window or tab.
      4. Copy the URL of the Issuer from the new browser window or tab.

For example, in our illustration:

<https://litware369b2c.b2clogin.com/tfp/cbf80679-8d13-4ac7-a5de-a4c1d69bd5d2/b2c_1_fcf_signupsignin/v2.0/>

* 1. Specify the (remaining) site settings in accordance to the following table:

|  |  |
| --- | --- |
| Field | Value |
| Authority | The URL retrieved in the previous step.  For example, in our illustration:  <https://litware369b2c.b2clogin.com/tfp/cbf80679-8d13-4ac7-a5de-a4c1d69bd5d2/b2c_1_fcf_signupsignin/v2.0/> |
| Client ID | <*your\_AppID*>  For example, in our illustration: 53d31778-d6fa-4331-8518-4c2d77ac1dd1 |
| Redirect URL | https://<*your\_AppName*>.powerappsportals.com/signin-aad-b2c\_1  For example, in our illustration:  <https://demofacade.powerappsportals.com/signin-aad-b2c_1> |

1. Leave the fields empty under Password reset settings.
2. Scroll down, and expand Additional settings.
3. Do NOT configure the settings for signing users out. This capability is not (yet) available in current bits of the FranceConnect Facade. Ensure that External logout is set to Off.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

1. Click Confirm.

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

1. Click Close.

### Test the B2C user flow from your Power Pages site

To test the previous B2C user flow with this time your Power Pages site, proceed with the following steps:

1. Open a new browser session and navigate to your Power Pages site at https://<*your\_AppName*>.powerappsportals.com where <*your\_AppName*> is the name of your app. For example, in our illustration:

<https://demofacade.powerappsportals.com>

1. Click Se Connecter and then Azure AD B2C. under Connexion externe.



You are redirected to the previously defined B2C user flow *B2C\_1\_signupsignin*. See section Create user flows above.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

1. Click the so-called FranceConnect button. You are redirected to the FranceConnect Facade (FCF), which in turn redirects you to the FranceConnect Platform (FCP) login page.
2. Like before, select Démonstration eIDAS faible to use a "Demo" identity provider (IdP) for testing purposes only. You are redirected to the selected IdP.
3. At the IdP sign-in page, enter you credentials. The multiple testing user accounts available are listed here:

<https://github.com/france-connect/identity-provider-example/blob/master/database.csv>.

1. Click Valider. You are now redirected back to the FranceConnect platform.
2. Give your consent by clicking Continuer sur <*Your\_Accredited\_Project*>. For example, Continuer sur Microsoft FRANCE - 49063 in our illustration. You are now redirected back to your Power Pages site through the FranceConnect Facade.
3. Once authenticated, the user's profile page is displayed in your Power Pages site.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Compared to the UI illustrated in the [Getting Started with the FranceConnect Façade (FCF)](https://github.com/microsoft/franceconnect-facade-dotnet-webapp-aspnetcore/blob/main/Documentation/Getting%20Started%20with%20FranceConnect%20Facade%20(FCF).docx) **guide, one should note that the FranceConnect iconography is absent in the upper left corner for Angela Claire Louise DUBOIS, more specifically:**



This is due to the fact that the identity\_provider claim issued by the FranceConnect Façade (FCF), and set to “franceconnect” isn’t reissued by Azure AD B2C, transmitted and consumed by your Power Pages site.

One possible approach to fix this behavior and instantiate the above UX may consists in using an [API connector](https://learn.microsoft.com/en-us/azure/active-directory-b2c/api-connectors-overview).

Such an API connector can indeed be used to integrate the above sign-up and sign-in user flow with a REST API to be exposed by the FranceConnect Façade (FCF). This will serve the sole purpose of augmenting/enriching the token issued by Azure AD B2C with the above mentioned claim, a.k.a. attribute. For more information, see [Token enrichment](https://learn.microsoft.com/en-us/azure/active-directory-b2c/add-api-connector-token-enrichment?pivots=b2c-user-flow).

As of this writing, this REST API and related endpoint is neither specified nor implemented. This is something that can be added in the future.

This concludes this chapter devoted to user flows in Azure AD 2BC.

# Leverage B2C custom policies for FranceConnect

**In order to complete this walkthrough, we assume that you already completed the chapters** Fulfill the prerequisites for your local environment **and** Configure the FranceConnect Façade (FCF) for Azure AD B2C **above, and thus deployed and configured your own FCF environment to operate in conjunction with Azure AD B2C.**

**This chapter considers such a foundation as a starting point.**

**After illustrating the use of B2C user flows, it’s now to move to B2C custom policies. As such, these policies enable you to create your own user journeys for complex identity experience scenarios that are not supported by user flows. Azure AD B2C uses custom policies to provide extensibility.**

**In this chapter, you will now:**

* **Configure and upload the provided Azure AD B2C custom policies located in the *Policies* folder.**
* **Register the jwt.ms web application.**
* **Test it against the IEF orchestration platform, and in turn the FranceConnect Facade (FCF), and ultimately the FranceConnect Platform (FCP).**

**These steps are covered in order in the next sections. Let’s start with the registration of Microsoft Authenticator.**

## Configure and upload the B2C custom policies

The custom policies for the FranceConnect Façade (FCF) integration are B2C custom policies with custom content definition, which are needed to display the so-called FranceConnect button required to sustain the UX.

You will therefore need to first upload the .html files to your Azure Storage account and then edit and upload the .xml custom policy files. See next section.

|  |  |  |
| --- | --- | --- |
| File | Id | Description |
| TrustFrameworkBase.xml | B2C\_1A\_TrustFrameworkBase | TrustFrameworkBase from the Starter Pack (if needed) |
| TrustFrameworkExtensions.xml | B2C\_1A\_TrustFrameworkExtensions | Extensions’ replacement of the TrustFrameworkExtensions from the Starter Pack for the FranceConnect Façade (FCF) |
| SignUpSigninWithFranceConnect.xml | B2C\_1A\_signup\_signin | Standard Signup/Sign-in B2C policy but with the FranceConnect button |
| TrustFrameworkLocalization.xml | B2C\_1A\_TrustFrameworkLocalization | Localization XML-formatted file for the supported languages, namely English and French. See [Language customization in Azure Active Directory B2C?](https://learn.microsoft.com/en-us/azure/active-directory-b2c/language-customization?pivots=b2c-custom-policy) |

### Deploy the custom html templates

See [Customize the user interface with HTML templates](https://docs.microsoft.com/en-us/azure/active-directory-b2c/customize-ui-with-html?pivots=b2c-user-flow#2-create-an-azure-blob-storage-account).

#### Set up an Azure Blob storage account for storing the custom .html files

Please refer to eponym section in previous chapter.

#### Configure CORS

Please refer to eponym section in previous chapter.

#### Test CORS

Please refer to eponym section in previous chapter.

### Configure the B2C custom policies

#### Create a secret key for in the portal

The FranceConnect-OpenIdConnect technical profile defined in the provided *TrustFrameworkExtensions.xml* file is configured to use a key for security reasons, namely B2C\_1A\_FranceConnectSecret, to store the client secret of your FranceConnect account.

For example, the FranceConnect dev account through the available so-called “integration key for public use”, has the following client secret value: 2791a731e6a59f56b6b4dd0d08c9b1f593b5f3658b9fd731cb24248e2669af4b.

Therefore, you will need at this stage to create a policy key in the Azure AD B2C portal, and (manually) set the key value to the client secret.

Proceed with the following steps:

1. Make sure you're using the directory that contains your Azure AD B2C tenant:
   1. Select the **Directories + subscriptions** icon in the portal toolbar.
   2. On the **Portal settings | Directories + subscriptions** page, find your Azure AD B2C directory in the directory name list, and then select **Switch**.
2. In the Azure portal, search for and select **Azure AD B2C**.
3. In the left pane, under **Policies**, select **Identity Experience Framework**.
4. In the left pane, under **Manage**, select **Policy keys**.
5. Create a policy key in the B2C portal with the name FranceConnectSecret and manually set the key value to the client secret of your FranceConnect account
   1. Click **Add** to add a new policy key. A **Create a key blade** opens up.

Une image contenant texte

Description générée automatiquement

* 1. For **Option**, select **Manual**.
  2. For the name, enter “*FranceConnectSecret*“.
  3. For **Secret**, specify the client secret of your FranceConnect account.
  4. For **Key usage**, select **Encryption**.
  5. Click **Create**.

#### Edit and update the B2C custom policies

Proceed with the following steps:

1. Open a PowerShell 7 window and navigate to the location where you cloned the repo, navigate to the *CustomPolicies* folder located under the *Source\****fcf.AzureADB2C*** directory, e.g., in our illustration:

*c:\franceconnect-facade-dotnet-webapi-aspnetcore\****Source\fcf.AzureADB2C\CustomPolicies***

1. Open Visual Studio Code.

PS C:\> code .

1. Search and replace with the command **Replace in Files** (Ctrl+Shift+H) in the set of all .xml custom policy files for *yourtenant*.onmicrosoft.com to the real fully qualified name of your B2C tenant. For example, litware369b2c.onmicrosoft.com in our illustration.
2. Then do the following changes to the *TrustFrameworkExtensions.xml* file:
   1. Find the place of the ClaimProvider definition for the FranceConnect platform (FCP).

<ClaimsProvider>

<Domain>FranceConnect</Domain>

   <DisplayName>Login with FranceConnect</DisplayName>

   <TechnicalProfiles>

    <TechnicalProfile Id="FranceConnect-OpenIdConnect">

      <DisplayName>FranceConnect</DisplayName>

      <Protocol Name="OpenIdConnect" />

          <Metadata>

            <!-- Action required -->

            <Item Key="METADATA">

              https://franceconnectfacade-url/common/.well-known/openid-configuration</Item>

            <!-- Action required -->

            <Item Key="client\_id">{FranceConnect Client Id}</Item>

            <Item Key="response\_types">code</Item>

            <Item Key="scope">openid profile birth email</Item>

            <Item Key="response\_mode">query</Item>

            <Item Key="HttpBinding">POST</Item>

            <Item Key="UsePolicyInRedirectUri">false</Item>

         </Metadata>

         <CryptographicKeys>

            <Key Id="client\_secret" StorageReferenceId="B2C\_1A\_FranceConnectSecret" />

         </CryptographicKeys>

         <OutputClaims>

            <OutputClaim ClaimTypeReferenceId="issuerUserId" PartnerClaimType="email" />

            <OutputClaim ClaimTypeReferenceId="givenName" PartnerClaimType="given\_name" />

            <OutputClaim ClaimTypeReferenceId="surName" PartnerClaimType="family\_name" />

            <OutputClaim ClaimTypeReferenceId="authenticationSource"

              DefaultValue="socialIdpAuthentication" />

            <OutputClaim ClaimTypeReferenceId="identityProvider" PartnerClaimType="iss" />

            <OutputClaim ClaimTypeReferenceId="email" />

         </OutputClaims>

         <OutputClaimsTransformations>

            <OutputClaimsTransformation ReferenceId="CreateRandomUPNUserName" />

            <OutputClaimsTransformation ReferenceId="CreateUserPrincipalName" />

            <OutputClaimsTransformation ReferenceId="CreateAlternativeSecurityId" />

          <OutputClaimsTransformation ReferenceId="CreateSubjectClaimFromAlternativeSecurityId" />

         </OutputClaimsTransformations>

      <UseTechnicalProfileForSessionManagement ReferenceId="SM-SocialLogin" />

    </TechnicalProfile>

</TechnicalProfiles>

</ClaimsProvider>

* 1. In the METADATA key, replace franceconnectfacade-url with the FQDN domain name of the façade. See section Execute the FranceConnect Façade above.

For a local deployment, this will be the ngrok address. For example, in our illustration: [7516-2a01-110-8012-1013-7925-1440-7ba6-d086.ngrok.io](https://7516-2a01-110-8012-1013-7925-1440-7ba6-d086.ngrok.io).

The OpenID Connect (OIDC) Discovery will then be as follows:

<https://7516-2a01-110-8012-1013-7925-1440-7ba6-d086.ngrok.io/common/.well-known/openid-configuration>

* 1. In the client\_id key, replace {FranceConnect Client Id} with the value of your FranceConnect account.

For example, the FranceConnect dev account through the available so-called “integration key for public use”, has the following Client ID value: 211286433e39cce01db448d80181bdfd005554b19cd51b3fe7943f6b3b86ab6e.

* 1. Find the place where LoadUri references your blob storage account and update the value of the URL for the .html custom template file you uploaded in the section **Deploy the custom html templates** to reflect your blob storage account.

For example, you will need to replace yourstorageaccount.blob.core.windows.net/customsignin

by litware369b2cstorage.blob.core.windows.net/customsignin in our illustration.

<ContentDefinitions>

<ContentDefinition Id="api.signuporsignin">

<!-- Action required -->

<LoadUri>https://yourstorageaccount.blob.core.windows.net/customsignin/unified.html</LoadUri>

<RecoveryUri>~/common/default\_page\_error.html</RecoveryUri>

<DataUri>urn:com:microsoft:aad:b2c:elements:contract:unifiedssp:2.1.5</DataUri>

<Metadata>

<Item Key="DisplayName">Signin and Signup</Item>

</Metadata>

</ContentDefinition>

</ContentDefinitions>

1. Save all the .xml custom policy files.

### Upload the B2C custom policies

Eventually, upload the above set of .xml custom policy files to your Azure AD B2C tenant, either via the Azure portal, the IefPolicies cmdlets or directly from Visual Studio Code, starting with *TrustFrameworkBase.xml* first if needed. Overwrite the custom policy if it already exists.

Please note that you already should have uploaded the *TrustFrameworkBase.xml* and *trustFrameworkExtensions.xml* in a previous step, see section Be ready for B2C custom policies above.

## Register the **jwt.ms web application**

The jwt.ms web application with the IEF orchestration platform. In fact, you already did it with the user flows. See section Register a web application above.

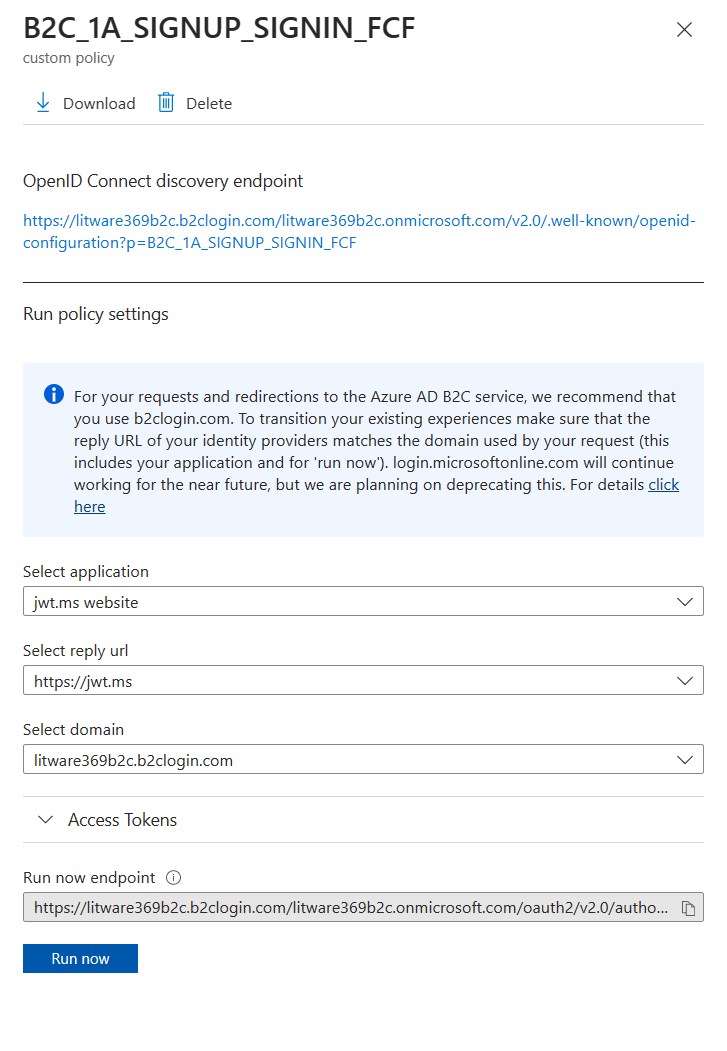
## **Test the B2C custom policies**

**At this stage,** you can verify that the FranceConnect Façade (FCF) works as expected with the IEF orchestration platform of Azure AD B2C. It will be called as part of the defined sequence of orchestration steps, as per executed custom policy.

Please note that Azure AD B2C has built-in support in the Azure portal for testing your B2C custom policies via the **Run now** functionality, similarly to what you did with user flows previously.

Proceed with the following steps:

1. Under Custom policies, select B2C\_1A\_SIGNUP\_SIGNIN\_FCF. An eponym B2C\_1A\_SIGNUP\_SIGNIN\_FCF blade opens up.



1. For Select application, select the web application that you previously registered. See section Register a web application above. For example, in our illustration jwt.ms website. In Select reply url, the Reply URL should show <https://jwt.ms>.
2. Click Run user flow. A sign-up or sign-in page opens up.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

1. From the sign-up or sign-in page, click the so-called FranceConnect button in the upper part. You are redirected to the FranceConnect Facade (FCF), which in turn redirects you to the FranceConnect Platform (FCP) login page.

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

1. Select one of the test identity providers. Like before, select Démonstration eIDAS faible. You are redirected to the selected IdP.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

1. At the selected identity provider, enter you credentials. The multiple testing user accounts available are listed here:

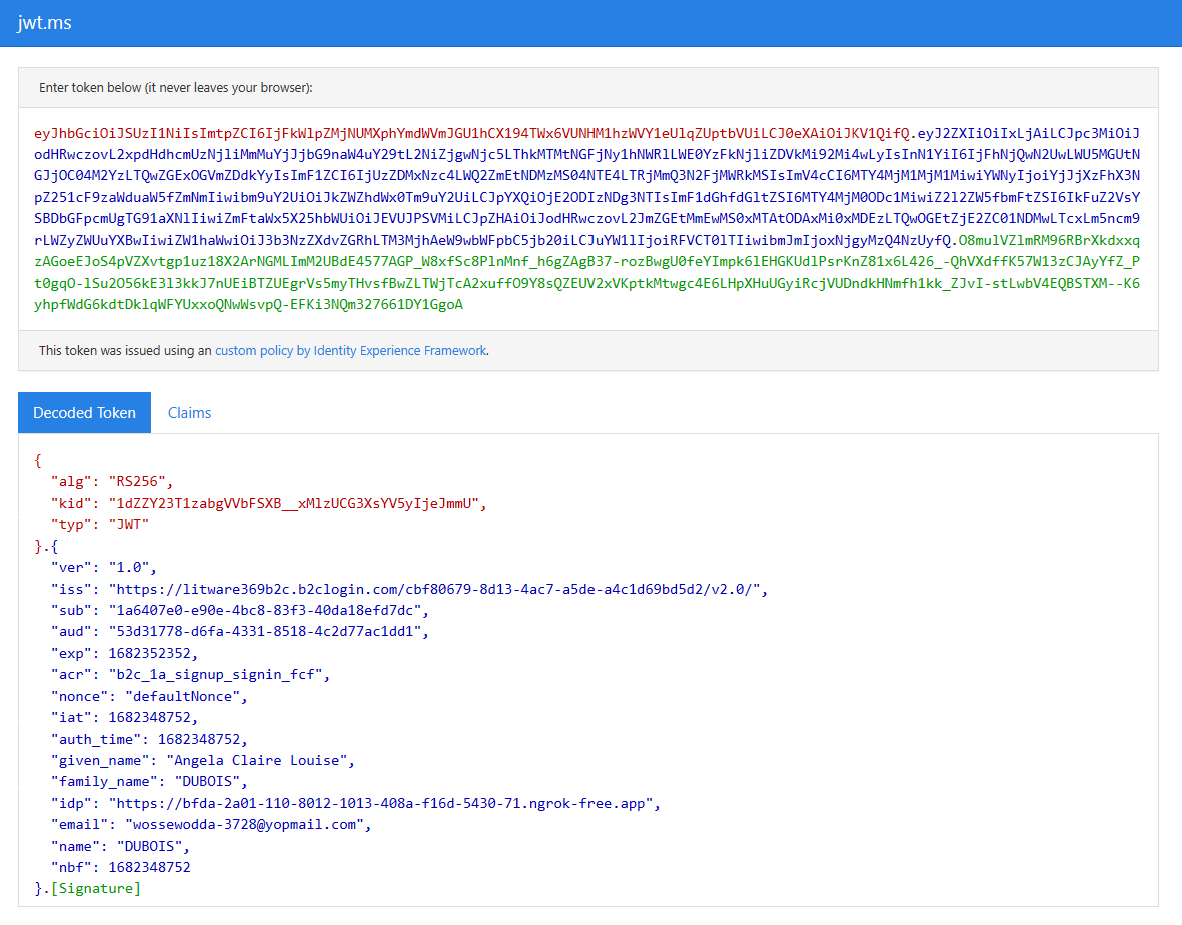
<https://github.com/france-connect/identity-provider-example/blob/master/database.csv>.

1. Click Valider. You are now redirected back to the FranceConnect platform.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

1. Give your consent by clicking Continuer sur <*Your\_Accredited\_Project*>. For example, Continuer sur Microsoft FRANCE - 49063 in our illustration. You are now redirected back to your Power Pages site through the FranceConnect Facade.
2. If the sign-in process is successful, your browser is redirected to <https://jwt.ms>, which displays the contents of the token issued by the FranceConnect façade (FCF) (based on the one received from the FranceConnect platform (FCP)), and ultimately returned by Azure AD B2C.



## Troubleshoot your B2C custom policies

See [Troubleshoot custom policies and user flows in Azure Active Directory B2C](https://learn.microsoft.com/en-us/azure/active-directory-b2c/troubleshoot?pivots=b2c-custom-policy#troubleshooting-with-application-insights) and [Troubleshoot custom policies with Application Insights](https://learn.microsoft.com/en-us/azure/active-directory-b2c/troubleshoot-with-application-insights?pivots=b2c-custom-policy).

To diagnose problems with your B2C custom policies, you can use Application Insights. [Application Insights](https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview?tabs=net) is an extension of [Azure Monitor](https://learn.microsoft.com/en-us/azure/azure-monitor/overview) and provides application performance monitoring (APM) features.

As such, and more specifically in our context, it allows you to trace the activity of your custom policy user journey. In other words, it provides a way to diagnose exceptions and observe the exchange of claims between Azure AD B2C and here the claims provider that was defined by technical profiles for the FranceConnect Façade (FCF).

### Set up Application Insights

Application Insights sends telemetry to a common Log Analytics workspace, providing full access to all the features of Log Analytics while keeping your custom policy user journeys, related infrastructure, and platform logs in a single consolidated location.

#### Create an instance of Application Insights

See [Create a new Azure Monitor Application Insights workspace-based resource](https://learn.microsoft.com/en-us/azure/azure-monitor/app/create-workspace-resource) and [Create Log Analytics workspaces](https://learn.microsoft.com/en-us/azure/azure-monitor/logs/quick-create-workspace?tabs=azure-portal).

If you don't already have one, create an instance of Application Insights in your Azure subscription.

Proceed with the following steps:

1. Open a browser session, navigate to the Azure portal, and sign-in with your administrative credentials.
2. Make sure you're using the directory that contains your Azure AD B2C tenant:
   1. Select the **Directories + subscriptions** icon in the portal toolbar.
   2. On the **Portal settings | Directories + subscriptions** page, find your Azure AD B2C directory in the directory name list, and then select **Switch**.
3. Select Create a resource in the left-hand navigation menu.
4. Search for “*Application Insights*” in the search box. As you begin typing, the list filters based on your input. Select Application Insights.
5. Select Create. An Application Insights page opens up for the instance to create.

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

1. Fill in the various settings under PROJECT DETAILS:
   1. In Subscription, select a Subscription from the dropdown list.
   2. In Resource Group, use an existing resource group or create a new one. For example, in our illustration B2C\_Configuration in the France Central region.
2. Fill in the various settings under INSTANCE DETAILS:
   1. In Name, provide a name for the new Log Analytics workspace, such as DefaultLAWorkspaceB2C. This name must be unique per resource group.
   2. In Region, select an available region. For example, in our illustration FranceCentral.

For more information, see which [regions Log Analytics is available in](https://azure.microsoft.com/regions/services/). Search for “*Azure Monitor*” in the Search for a product box.

* 1. In Resource Mode, ensure Workspace-based is selected.

1. Leave untouched the settings under WORKSPACE DETAILS.
2. Select Review + Create to review the settings. Then select Create to create the an instance of Application Insights.
3. Once the deployment has been completed, and your instance of Application Insights is created, select Go to resources.
4. Under Configure in Application Insights menu, select Properties.

Une image contenant texte, ligne, Police, capture d’écran

Description générée automatiquement

1. Take a note of INSTRUMENTATION KEY for use in a later step below. It will be referred as to <*your\_InstrumentationKey*>.

#### Use an existing instance of Application Insights

To use an existing instance of Application Insights in your subscription, proceed with the following steps:

1. If needed, open a browser session, navigate to the Azure portal, and sign-in with your administrative credentials.
2. Make sure you're using the directory that contains your Azure AD B2C tenant:
   1. Select the **Directories + subscriptions** icon in the portal toolbar.
   2. On the **Portal settings | Directories + subscriptions** page, find your Azure AD B2C directory in the directory name list, and then select **Switch**.
3. Open the instance of Application Insights that you created earlier.
4. On the Overview page, take a note of Instrumentation key for use in a later step below. It will be referred as to <*your\_InstrumentationKey*>.

### Update your B2C custom policies

Proceed with the following steps:

1. Open a PowerShell 7 window and navigate to the location where you cloned the repo, navigate to the *CustomPolicies* folder located under the *Source\****fcf.AzureADB2C*** directory, e.g., in our illustration:

*c:\franceconnect-facade-dotnet-webapi-aspnetcore\****Source\fcf.AzureADB2C\CustomPolicies***

1. Open Visual Studio Code.

PS C:\> code .

1. Do the following changes to the *TrustFrameworkExtensions.xml* file:
   1. Set the attributes DeploymentMode and UserJourneyRecorderEndpoint as follows to the [TrustFrameworkPolicy](https://learn.microsoft.com/en-us/azure/active-directory-b2c/trustframeworkpolicy) XML element.

<TrustFrameworkPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns="http://schemas.microsoft.com/online/cpim/schemas/2013/06"

PolicySchemaVersion="0.3.0.0"

TenantId=" http://litware369b2c.onmicrosoft.com"

PolicyId="B2C\_1A\_signup\_signin"

PublicPolicyUri="http://litware369b2c.onmicrosoft.com/B2C\_1A\_signup\_signin"

DeploymentMode="Development" UserJourneyRecorderEndpoint="urn:journeyrecorder:applicationinsights>

* 1. Scroll down to the following JourneyInsights node as a child of the [UserJourneyBehaviors](https://learn.microsoft.com/en-us/azure/active-directory-b2c/relyingparty#userjourneybehaviors) XML element:

<UserJourneyBehaviors>

<!-- Action required -->

<JourneyInsights TelemetryEngine="ApplicationInsights"

InstrumentationKey="{Your Application Insights Key}" DeveloperMode="true"

ClientEnabled="false" ServerEnabled="true" TelemetryVersion="1.0.0" />

<ScriptExecution>Allow</ScriptExecution>

</UserJourneyBehaviors>

* 1. Make sure to replace {Your Application Insights Key} with the Application Insights Instrumentation Key that you took a note of earlier, i.e. the above-mentioned <*your\_InstrumentationKey*> value.
  2. Ensure the other present parameters are set as follows:
     + DeveloperMode="true". This setting tells Application Insights to expedite the telemetry through the processing pipeline. This settings is appropriate for development purpose, but constrained at high volumes. In production, you should set the DeveloperMode to false instead.
     + ClientEnabled="true". This setting sends the Application Insights client-side script for tracking page view and client-side errors. You can view these in the browserTimings table in the Application Insights portal.

With the setting, you add Application Insights to your page script and you get timings of page loads and AJAX calls, counts, details of browser exceptions and AJAX failures, and user and session counts. This field is optional, and is set to false by default.

* + - ServerEnabled="true". This setting sends the existing UserJourneyRecorder JSON as a custom event to Application Insights.

1. Save the file.
2. Upload the file as per section Update your B2C custom policies above.

### Rerun your B2C custom policies

Please refer to the section **Test the B2C custom policies** above.

You can now see the related logs either in Application Insights or in VS Code - There is a short delay, typically less than five minute -.

Let’s see that before concluding.

### See the logs in Application Insights

Proceed with the following steps:

1. Open the Application Insights resource that you created in the Azure portal. See section Set up Application Insights above.
2. On the Overview page, select Logs.
3. Open a new tab in Application Insights.

Here is a list of queries you can use to see the logs:

|  |  |
| --- | --- |
| Query | Description |
| traces | Get all of the logs generated by Azure AD B2C |
| traces | where timestamp > ago(1d) | Get all of the logs generated by Azure AD B2C for the last day. |
| traces | where message contains "exception" | where timestamp > ago(2h) | Get all of the logs with errors from the last two hours. |
| traces | where customDimensions.Tenant == "litware369b2c.onmicrosoft.com" and customDimensions.UserJourney == "b2c\_1a\_signup\_signin\_fcf" | Get all of the logs generated by your B2C tenant, for example litware369b2c.onmicrosoft.com in our illustration, and where the user journey is the above B2C\_1A\_SIGNUP\_SIGNIN\_FCF. |
| traces | where customDimensions.CorrelationId == "00000000-0000-0000-0000-000000000000" | Get all of the logs generated by Azure AD B2C for a correlation ID. Replace the correlation ID with your correlation ID. |

The entries may be long. Export to CSV for a closer look. See [Log queries in Azure Monitor](https://learn.microsoft.com/en-us/azure/azure-monitor/logs/log-query-overview).

### See the logs in VS Code

With the [Azure AD B2C extension for VS Code](https://marketplace.visualstudio.com/items?itemName=AzureADB2CTools.aadb2c) you installed earlier (see section Install Azure AD B2C extension for Visual Studio Code above), the logs are organized for you by the policy name, correlation ID (the application insights presents the first digit of the correlation ID), and the log timestamp.

This feature helps you to find the relevant log based on the local timestamp and see the user journey as executed by Azure AD B2C.

This concludes this chapter devoted to custom policies in Azure AD 2BC.

# As a conclusion

This concludes this walkthrough and this guide.

We hope you have enjoyed the additional tour with Azure AD B2C, the potential and the power of the user flows custom policies, etc., and you are not too tired with all those technical details regarding the integration with FranceConnect Façade (FCF), and ultimately the ability to authenticate against the FranceConnect Platform (FCP) via the so-called FranceConnect button from client applications registered in a B2C tenant.

As part of this guided tour, we have outlined all the steps required to further enhance your own development environment in Azure with Azure AD B2C, to define and customize B2C user flows, customize and upload B2C custom policies for such an integration path starting from the “plumbing” from a protocol perspective moving towards the required UX and the use of the FranceConnect button along with some additional explanations and related rational.

|  |  |
| --- | --- |
|  |  |

1. Customization for the [Classic](https://login.microsoftonline.com/static/tenant/default/unified.cshtml), and [Slate Gray](https://login.microsoftonline.com/static/tenant/templates/MSA/unified.cshtml) templates are not provided in this guide. [↑](#footnote-ref-2)