FranceConnect Façade (FCF)

Zzz

Technical-functional specifications intended for Microsoft Dynamics 365 BizApps Portals, Power Apps Portals/Power Pages applications, and Azure AD B2C support

Version Beta - October 2022

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/data-in-use-protection-workshop/blob/master/LICENSE) file, and grant you a license to any code in the repository under the MIT License, see the [LICENSE-CODE](https://github.com/microsoft/data-in-use-protection-workshop/blob/master/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

[The problem’s statement 6](#_Toc117156912)

[A brief recap of OIDC provider support from D365 Biz Apps’ portals 6](#_Toc117156913)

[A brief recap of services’ support from the FranceConnect platform 11](#_Toc117156914)

[Issues at the OIDC protocol level 13](#_Toc117156915)

[Issues at UX level 14](#_Toc117156916)

[The proposed solution 14](#_Toc117156917)

[In scope 15](#_Toc117156918)

[Out of scope 15](#_Toc117156919)

[Risks and assumptions 15](#_Toc117156920)

[Dependencies 16](#_Toc117156921)

[The proposed changes 17](#_Toc117156922)

[Changes for the login experience 17](#_Toc117156923)

[Changes for the logout experience 19](#_Toc117156924)

[Environments supported 20](#_Toc117156925)

[Targeted environment 20](#_Toc117156926)

[Prerequisites for a development environment 20](#_Toc117156927)

[Prerequisites for a testing environment 21](#_Toc117156928)

[Prerequisites for the targeted environment 21](#_Toc117156929)

[Reason to choose an inferior solution (optional) 21](#_Toc117156930)

[Code base for the proposed solution 21](#_Toc117156931)

[Motivations 22](#_Toc117156932)

[Solution endpoints 22](#_Toc117156933)

[Metrics 23](#_Toc117156934)

[Telemetry 23](#_Toc117156935)

[Error reporting 23](#_Toc117156936)

[Additional references 25](#_Toc117156937)

# The problem’s statement

*This section will be used to provide a brief overview of the solution purpose and/or problem statement. Summarize desired business outcomes. Give details about the exact problem being addressed by the custom solution. Include data points and/or customer testimonials to elucidate the point. It should also include an estimate of the scale of the problem, e.g., “How many customers are being affected?”, etc.*

In 2019, the **French government** created DINUM (Inter-Ministerial Digital Directorate) to support the digital transformation of the state and all communities. DINUM is an entity very influential attached to the Prime Minister.

DINUM supports ministries, towns, departments and regions. But it is also developing shared services and resources such as **FranceConnect**.

**FranceConnect** (FC) is a solution offered by DINUM which makes it possible to secure and simplify the connection of all French citizen with e-administration, and the related online services. See <https://franceconnect.gouv.fr>, <https://partenaires.franceconnect.gouv.fr/> Interestingly, a demo environment is available [here](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Ffournisseur-de-service.dev-franceconnect.fr%2F&data=04%7C01%7CPhilippe.Beraud%40microsoft.com%7C14cddd7aaa2c4a1145e308d9d9a352d3%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637780117765655732%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=JBjuPYdUB%2BS7YnJEaPhnJS01ZI5Xbj%2BOUZ9UDuDQn1c%3D&reserved=0).

FranceConnect figures:

* 40+ million users.
* 18 million monthly connections.
* Access to over 1400 online services.

In terms of supported identity providers (IdP), this platform makes it possible for citizens to use an existing account @ an administrations or an e-gov agency’s website, e.g., IRS and social security.

Dynamics 365 Biz Apps portals, the Power Apps Portals/Power Pages websites[[1]](#footnote-2),[[2]](#footnote-3), as well as Azure AD B2C, cannot integrate with the FranceConnect platform (FCP), while both these offerings and FCP are based on industry standard protocol, namely OpenID Connect (OIDC) w/ the authorization code flow. As always, the devil resides in detail.

**To better understand the issues let’s start with a brief recap of the OpenID Connect-based authentication support in the D365 Biz Apps portals.** (Dynamics 365 Biz Apps portals, and the Power Apps Portals/Power Pages websites are all referred to as D365 Biz Apps portals in the rest of this document.)

## A brief recap of OIDC provider support from D365 Biz Apps’ portals

Generally speaking, an external identity provider that supports OpenID Connect can be integrated with 365 Biz Apps’ portals and Power Apps portals. Two paths are provided: i) direct integration or ii) via Azure AD B2C (user flows).

The next two sections briefly depict in order each of these two paths for the user sign-in experience.

### **Direct support w/ authorization code flow**

The direct support is suitable to integrate with an identity provider that implements OpenID Connect in the “most common options”, starting with the availability of an **auto-discovery document exposed through a discovery endpoint, a.k.a. /.well-known/openid-configuration**.

The configuration is covered in the documentation here: [Configure an OpenID Connect provider for portals - Power Apps | Microsoft Docs](https://docs.microsoft.com/en-us/powerapps/maker/portals/configure/configure-openid-provider).

#### Login experience

**Une image contenant texte

Description générée automatiquement**

**The flow is as follows:**

1. **To integrate with portals, and automatically configure them, the above-mentioned auto-discovery document is required here. Information present in this document is used by portals to create authorization requests and validate the authentication tokens. This metadata file is requested from the provider and must contain at least:**
2. **URL of the Authorize page.**
3. **URL of the Token page.**
4. **URL of the JKWS page (to typically retrieve the RS256 public signature key of the provider).**
5. **When the user signs in, the portal redirects to the OIDC provider's Authorize page.**

* **The provider proceeds with the authentication the user with its own mechanisms.**
* **Once authenticated, the provider sends the authentication response to the Callback URL configured in the provider and set as a Redirect URI parameter in the Authorize page call.**
* **The provider includes in the authentication response the code to retrieve the user's identity.**

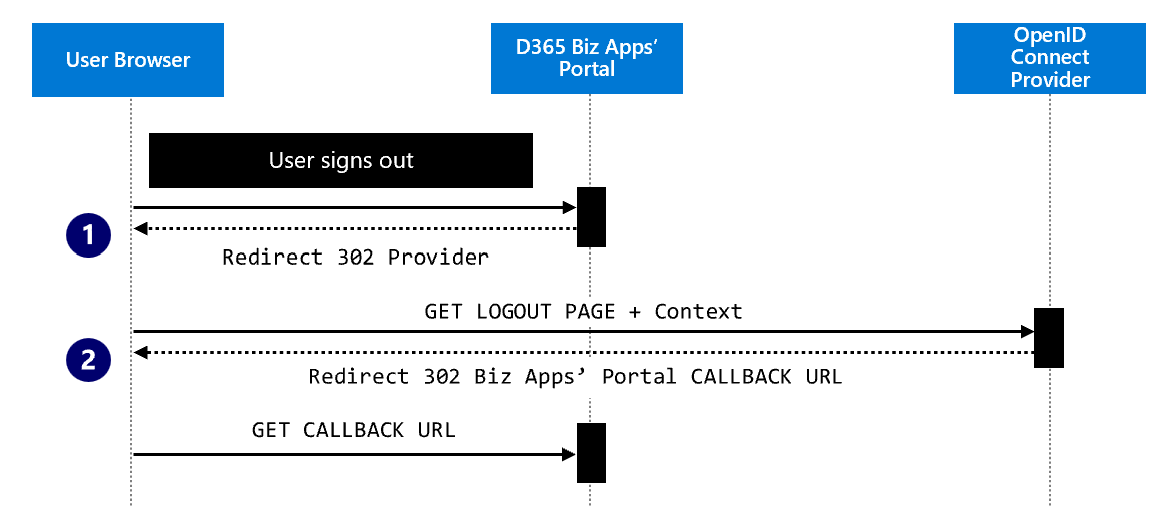
1. **When the portal retrieves the code, it queries the provider by passing the retrieved code to the provider’s Token page.**

* **The URL of the Token page is the one retrieved in the above metadata.**
* **The provider issues in turn both an access token and a JSON Web Token (JWT), i.e., an id\_token, which , which is a security token that allows the client to verify the identity of the user. As such, the id\_token contains attributes about the authenticated user, a.k.a. claims.**
* **The id\_token is signed by the provider using a private key, typically via the RS256 algorithm (RSA Signature with SHA-256).**

1. **In order to validate the signature, the portal downloads the public key of the provider from the JKWS page above to validate the signature of the id\_token.**
2. **A session is eventually established on the portal and the requested content, i.e., an authenticated page, is provided to the user.**

#### **Logout experience**

The portal optionally supports external account sign-out. When this capability is enabled, users are redirected to the external sign-out user experience when they sign out from the portal. When disabled, users are rather only signed out from the portal.



To externally sign-out, the user **initiates the external logout flow by clicking the logout link presented by the portal. This triggers the following flow:**

1. **The portal redirects the user to the Logout page of the provider:**
2. In the portal authentication configuration, the Post logout redirect URL setting must indicate the location where the provider will redirect a user after external sign-out. This location should be set appropriately in the provider configuration.
3. **The portal specifies this URL via the post\_logout\_redirect\_uri parameter.**
4. **The user is redirected to the indicated Callback URL.**

### **Support via Azure AD B2C**

The support via Azure AD B2C can be seen as an extension of the previous direct support that enables external customers to sign in through local credentials and federation with various identity providers, a.k.a. claims providers.

**This path supposes to leverage user flows, a.k.a. predefined, built-in, configurable policies, in the Azure AD B2C tenant.**

User flows notably allow to create sign-in experiences in minutes. The configuration is covered in the documentation here:

* [Configure the Azure Active Directory B2C provider (using interface in preview) - Power Apps | Microsoft Docs](https://docs.microsoft.com/en-us/power-apps/maker/portals/configure/configure-azure-ad-b2c-provider)
* [Configure an OpenID Connect provider for portals - Power Apps | Microsoft Docs](https://docs.microsoft.com/en-us/power-apps/maker/portals/configure/configure-openid-provider)

In addition to user flows, Azure AD B2C also support another way to provide identity user experiences with the custom policies. Custom policies enable to create your own identity experience scenarios where technical profiles can be defined for claims providers to accommodate specific situations.

#### Login experience

The endpoints exposed by the Azure AD B2C tenant are specific to the user flow, a.k.a. policy, that the portal wishes to use to sign-in the user.

* **Discovery endpoint URL:**

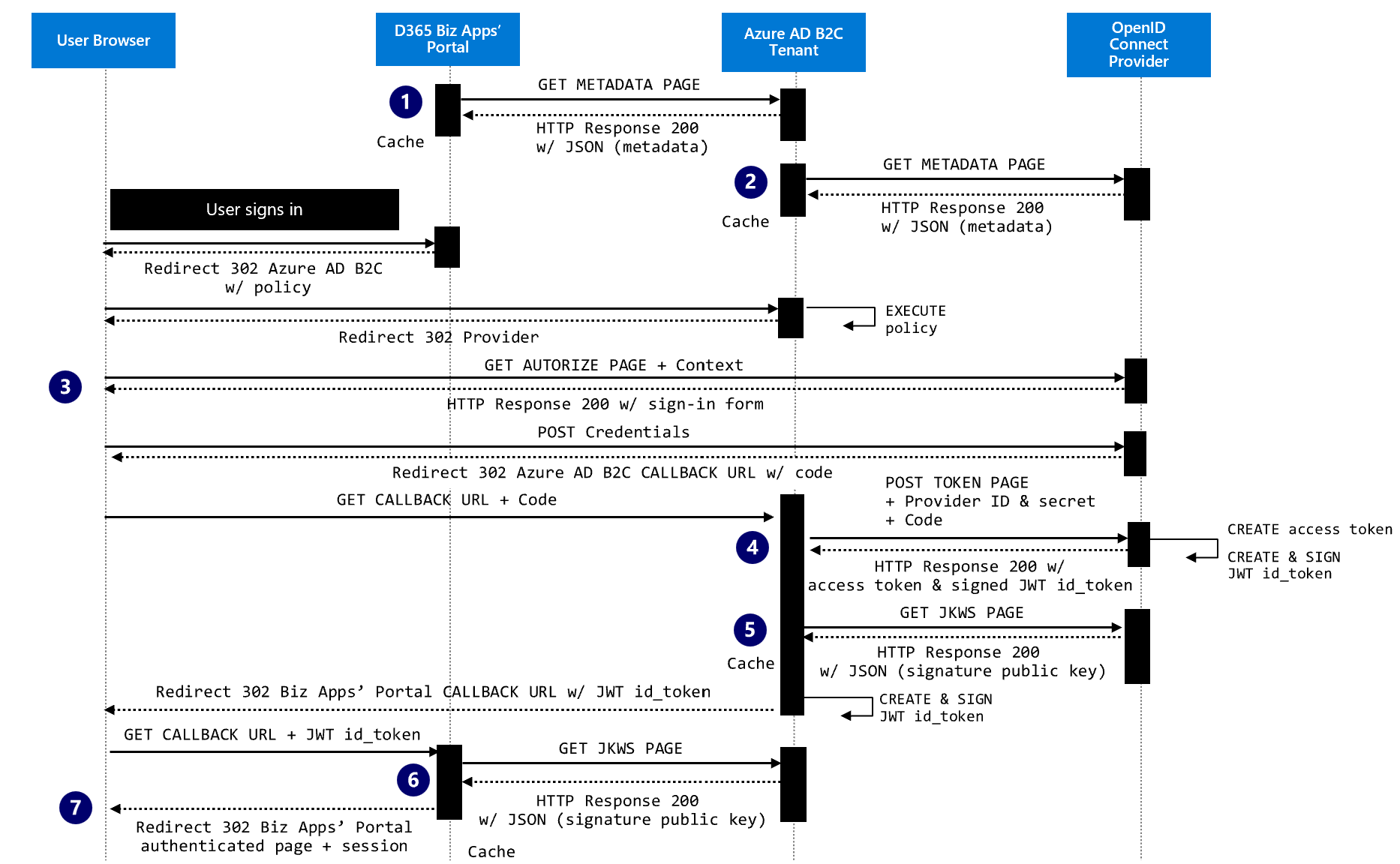
https://{*tenant*}.b2clogin.com/{*tenant*}.onmicrosoft.com/{*policy*}/oauth2/v2.0/**.well-known/openid-configuration**

* Authorization page:

https://{*tenant*}.b2clogin.com/{*tenant*}.onmicrosoft.com/{*policy*}/oauth2/v2.0/authorize

* Token page:

https://{*tenant*}.b2clogin.com/{*tenant*}.onmicrosoft.com/{*policy*}/oauth2/v2.0/token

****

**The flow is as follows:**

* **An auto-discovery document is also required here from this time the Azure AD B2C tenant, which is per policy.**
* **In turn, at the Azure AD B2C tenant level, yet another auto-discovery document is also required here is as part of the definition of a generic OpenID Connect identity provider for the considered user flow. The provider discovery endpoint is specified in the** Metadata url **setting in the UI for the provider definition.**

**Note: With custom policies, the required METADATA attribute of a technical profile definition of the provider must point to the provider discovery endpoint, see** [Define an OpenID Connect technical profile in a custom policy - Azure AD B2C | Microsoft Docs](https://docs.microsoft.com/en-us/azure/active-directory-b2c/openid-connect-technical-profile).

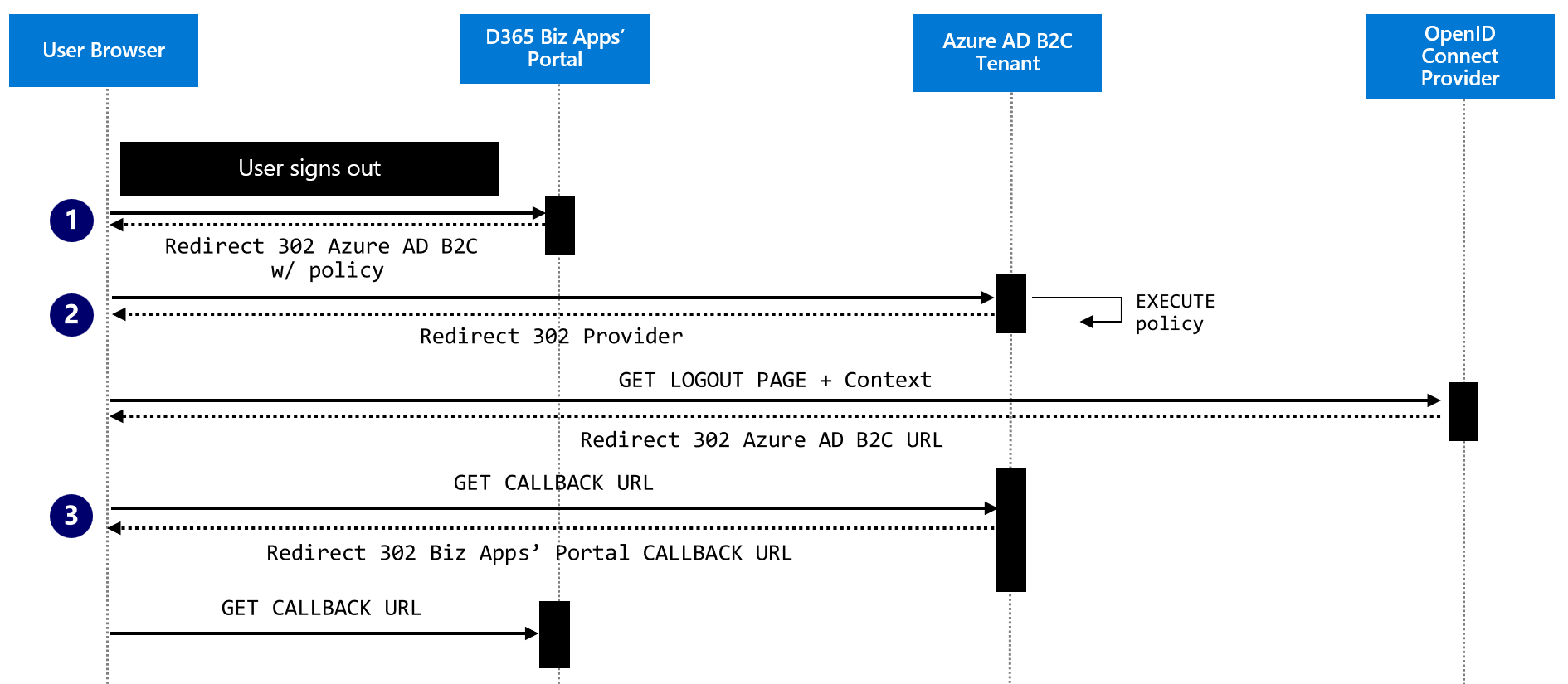
**For the configuration of the related provider as part of the pointed policy, this metadata file is requested from the provider and contain the same information as before. It is used later with the related state machine of the Identity Experience Framework (IEF) orchestration platform of Azure AD B2C.**

* **When the user signs in, the portal redirects this time to the Azure AD B2C tenant to execute the specified policy. As part of the orchestration steps, the user is redirect to the OIDC provider's Authorize page. Once authenticated, the user is redirected to the Azure AD B2C Tenant’s Callback URL configured in the provider and passed in parameter during the Authorize page call.**
* **When the IEF orchestration platform as part of the execution of the policy retrieves the code, it queries the provider by passing the retrieved code to the provider’s Token page. As before, the provider issues both an access token and an id\_token.**
* **In order to validate the signature, the IEF orchestration platform downloads the public key of the provider from the JKWS page above to validate the signature of the id\_token. After completing the orchestration steps specified in the policy, the IEF orchestration platform issues, signs, and returns yet another id\_ token to the portal.**
* **To validate the signature of the received id\_token, the portal downloads the Azure AD B2C Tenant’s public key. The user information is extracted, and a user context is created for the user.**
* **A session is eventually established on the portal and the requested content, i.e., an authenticated page, is provided to the user.**

#### **Logout experience**

When you want to sign the user out of the portal, it isn't enough to clear the application's cookies or otherwise end the session with the user. You must redirect the user to Azure AD B2C to sign out. Otherwise, the user might be able to re-authenticate to your portals without entering their credentials again.

**See sections “Sign-out” & “Single sign-out ” in** [Configure session behavior - Azure Active Directory B2C | Microsoft Docs](https://docs.microsoft.com/en-us/azure/active-directory-b2c/session-behavior?pivots=b2c-custom-policy#single-sign-out).



To externally sign-out, the user **initiates the external logout flow by clicking the logout link presented by the portal. This triggers the following flow:**

1. The portal redirects the user to the Azure AD B2C tenant’s Logout page.
   1. The Azure AD B2C tenant’s Logout page is specified in the end\_session\_endpoint location that is listed in the Azure AD tenant’s a**uto-discovery document**:

https://{*tenant*}.b2clogin.com/{*tenant*}.onmicrosoft.com/{*custompolicy*}/oauth2/v2.0/logout

1. **Upon** a sign-out request, **the IEF orchestration platform** attempts to sign out from an external provider if the provider **auto-discovery document exposed through a discovery endpoint, a.k.a. /.well-known/openid-configuration**, specifies an end\_session\_endpoint location.

Note: With custom policies, if in the provider technical profile metadata, the SingleLogoutEnabled setting is set to true, **the IEF orchestration platform** will also attempt to sign out from the external provider in this case.

The sign-out request doesn't pass by default the id\_token\_hint parameter. To enforce the presence of a previously issued id\_token passed to the Logout page, in the properties of the policy in the UI, the Require ID Token in logout requests setting in Session behavior must be turned on.

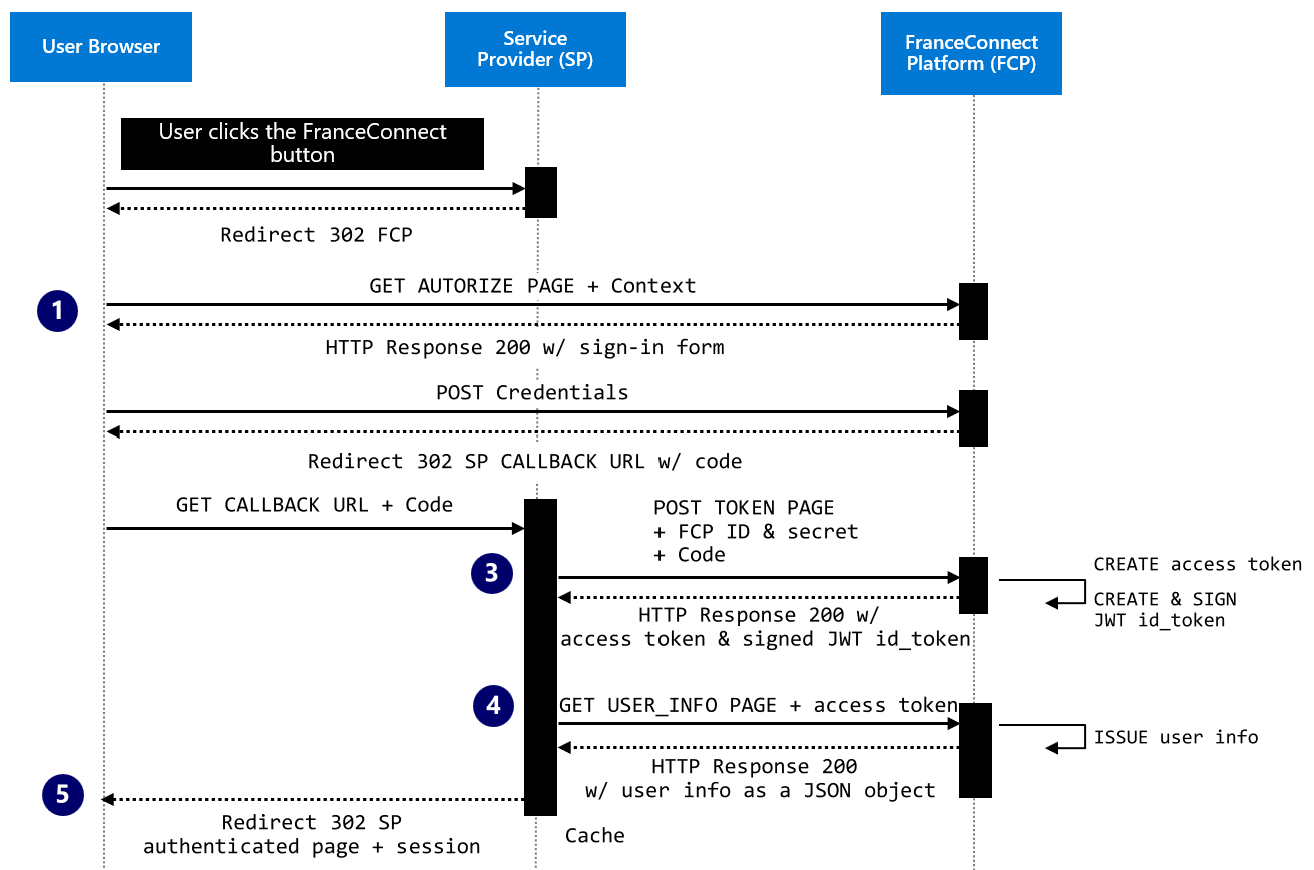
Note: With custom policies, the sign-out request doesn't pass the id\_token\_hint parameter. If the provider requires this parameter, the sign-out request will fail.

1. **After logout, the user is redirected to the URI specified in the post\_logout\_redirect\_uri parameter, regardless of the Redirect URIs that have been specified for the application defined in the Azure AD B2C tenant for the portal.** 
   1. **In the application defined in the Azure AD B2C tenant for the portal, the Front-channel logout URL setting in Authentication must be set to the Callback URL** to call to have the application clear the user's session data. **his parameter** that is required for single sign-out to work correctly.
   2. **If a valid id\_token\_hint is passed and the above Require ID Token in logout requests setting is turned on, the IEF orchestration platform verifies that the value of post\_logout\_redirect\_uri matches one of the application's configured Redirect URIs before performing the redirect. If no matching Redirect URI was configured for the application, an error message is displayed, and the user is not redirected.**

## A brief recap of services’ support from the FranceConnect platform

### **Login experience**

The FC Service Provider specification fully describes how a service provider integrates with FCP, see <https://partenaires.franceconnect.gouv.fr/fcp/fournisseur-service>.



**To sign-in, the user initiates the authentication flow by clicking the so-called FranceConnect button.**

**This triggers the following flow:**

1. **The service provider (SP) redirects the user to the Authorization page of the FranceConnect platform (FCP):**

* **The URL of the Authorization page is provided as per specification: integration platform, a.k.a. sandbox environment, vs. production environment.**
* **FCP delegates in turn the authentication to a supported OIDC provider – not illustrated here for the sake of brevity -. The user is invited to select one of the listed OIDC provider and redirected to it to authenticated**
* **Once authenticated, the user is redirected back to FCP, and after his/her consent, the user is finally redirected to the Callback URL configured in FCP and passed in parameter during the Authorize page call.**
* **FCP includes the code to retrieve the user's identity.**

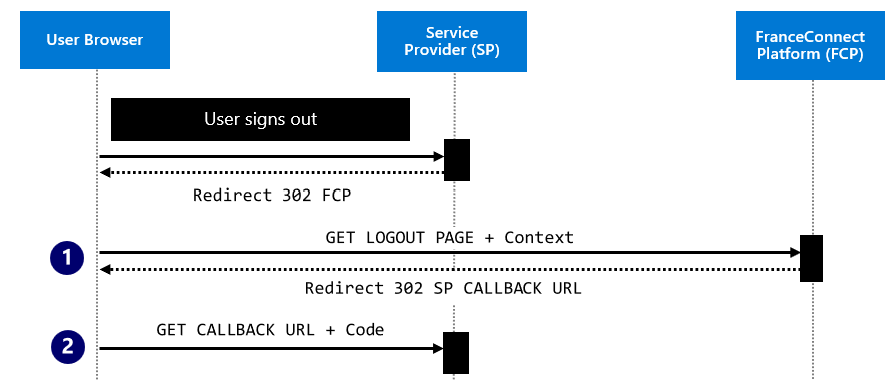
1. **When the portal retrieves the code, it queries FCP by passing the retrieved code to the Token page:**

* **The URL of the Token page is provided as per specification: integration platform, a.k.a. sandbox environment, vs. production environment.**
* **FCP issues in turn both an access token and a JSON Web Token (JWT), i.e., an id\_token, which contains some of the attributes of the authenticated user. The id\_token only contains the so-called technical identifier SUB.**
* **The id\_token is signed by FCP using a symmetric key via the HS256 algorithm (HMAC with SHA-256).**

1. **In order to validate the signature, the service provider uses the same secret.**
2. **The access token is then used to get the remaining user attributes. The whole user information, a.k.a. the pivot identity or a subset of it depending on the provided scope, is used to create a user context for the user.**
3. **The session on the portal is created and the requested content is provided to the user.**

### **Logout experience**

As per above-mentioned specification, FCP allows the user to disconnect from his/her FranceConnect session.



To sign-out, the user **initiates the external logout flow by clicking the logout link presented by the SP. The SP must disconnect both from its own session, if any, and the FC session.**

**This triggers the following flow:**

1. **The service provider (SP) redirects the user to the Logout page of the FranceConnect platform (FCP):**
2. **The SP must specify the Callback URL to which the user should be redirected once he has chosen to log out or not from FranceConnect via the post\_logout\_redirect\_uri parameter**
3. **The id\_token retrieved during user authentication is also passed via the id\_token\_hint parameter.**
4. **The user is redirected to the Callback URL indicated by the user.**

**The comparison between the above flow and the above-described ones from the Biz App’s perspective, along with the related specifications, and documentations that further depict them raises a number of issues: at the OIDC protocol level vs. at the UX level. The next two sections provide additional details on the identified issues.**

## **Issues at the OIDC protocol level**

*Complete the table below to describe the identified issues.*

|  |  |  |
| --- | --- | --- |
| # | Issue | Notes |
| 1 | No OIDC discovery endpoint | [Portals require to set a OIDC discovery endpoint](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fdocs.microsoft.com%2Fen-us%2Fpowerapps%2Fmaker%2Fportals%2Fconfigure%2Fconfigure-openid-faqs%23do-i-require-an-openid-connect-auto-discovery-document-to-integrate-with-portals&data=04%7C01%7CPhilippe.Beraud%40microsoft.com%7C14cddd7aaa2c4a1145e308d9d9a352d3%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637780117765655732%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=lVrK1ZhI%2BtAno4lGuOge%2FGM4n9to%2Fqah22LmuSK%2FlFM%3D&reserved=0) for the identity provider (IdP). However, the FCP doesn’t expose any OIDC Discovery Endpoint, and thus no associated document is provided. This point can be circumvented, as it is possible to create these elements from the FranceConnect spec and host them on our own. The various endpoints to interact with and the callback to expose are indeed fully described along w/ the related flow. |
| 2 | Support of specific OIDC scopes | The FC spec extends the OIDC scopes’ mechanism to be somehow “more modular”. While only opendid is mandatory, following values are also supported and possibly expected: given\_name, family\_name, birthdate, idp\_birthdate, gender, female, male, birthplace, birthcountry, email, preferred\_username, profile, birth, and identite\_pivot. Some of these scopes are custom ones like identite\_pivot.  A given SP is not expected to support all the possible values. However, the one(s) requested should allow the mapping w/ an Azure AD account based on the content of the tokens being returned: access token and id token. In other words, it is important to ensure an exact match between the data received from FranceConnect in these tokens and the corresponding account name for acct linking/impersonation. Thus, the (statically) specified scope(s) should allow to return the expected data. Applying specific transformation(s) may also be necessary. |
| 3 | Support of the acr\_values parameter | An acr\_values parameter allows to specify the expected eIDAS level (eidas1: standard, eidas2: substantial, and eidas3: high) for the AuthN context. Without this parameter being specified, only accredited IdP that support the maximum level eIDAS (eidas3) can be used, which is very few, to not say none.  As per [FAQs for using OpenID Connect in portals - Power Apps | Microsoft Docs](https://docs.microsoft.com/en-us/powerapps/maker/portals/configure/configure-openid-faqs): portals don't support acr\_values request parameters in authorization requests. However, the portals feature does support all the required—and recommended—request parameters defined in the [OpenID Connect specification](https://openid.net/specs/openid-connect-core-1_0.html#AuthRequest). |
| 4 | Presence of telemetry parameters | FranceConnect refuses requests with unknown parameters. Telemetry parameters, e.g., "x-client-SKU" and "x-client-ver", part of the default behavior of ASP.NET (Core) since version 2, are examples of such unknown parameters. |
| 5 | Support of the HS256 token signing algorithm | The token signature algorithm is HS256, a symmetric algorithm that is neither supported at the Biz Apps’ portals level nor in the Azure AD B2C custom policies |
| 6 | Nonce with a dot | Whenever the "nonce" sent during the connection contains a character such as a dot, an erratic disconnection behavior can be encountered.  The default behavior of ASP.NET Core includes a dot in this nonce, so it is likely to be the same here with the Biz Apps’ portals. We opened a ticket to change such a behavior, and this was also refused by FranceConnect: “We will not be able to modify the restrictions on the nonce”. |
| 7 | Call of the USER\_INFO endpoint as part of the Identity dance | The SP must call the USER\_INFO endpoint to complete the identity dance and retrieve all the user information. The id\_token retrieved from the token endpoint at the end of the client credential flow isn’t sufficient.  This is not part of the flow for the Biz Apps’ portals. This also not supported in Azure AD B2C user flows. This can be addressed in an Azure AD B2C custom policy with the following XML element:  <Item Key="ClaimsEndpoint">https://foo/oauth2/v1/userinfo</Item> |
| 8 | Absence of id\_token\_hint with custom policies | With custom policies, the sign-out request doesn't pass the id\_token\_hint parameter required by FranceConnect. |
| 9 | [insert issue title here] | [insert notes here] |

## **Issues at UX level**

*Complete the table below to describe the identified issues.*

|  |  |  |
| --- | --- | --- |
| # | Issue | Notes |
| 1 | So-called FranceConnect button | From a UX perspective, any conformant Service Provider must implement the FranceConnect button to interact w/ FranceConnect. |
| 2 | [insert issue title here] | [insert notes here] |

# The proposed solution

*This section gives a summary of the proposed solution. The proposed scenario needs to be explained and how it solves the problem from the previous section.*

**The proposed solution aims at specifying and developing a FranceConnect Façade (FCF) as a lightweight adaptation layer to handle all the above identified discrepancies and cope with the related issues.**

This façade must support in terms of instantiation various environments and platforms from the public cloud to the on-premises to accommodate specific customer requirements.

## In scope

*This section gives a summary of what is in scope. What needs to be implemented for this technical vs. functional spec to be deemed as done.*

*Complete the table below to describe the features/the capabilities that are in scope and set their priority.*

|  |  |  |
| --- | --- | --- |
| # | Title | PRI |
| 1 | Complete support of the FC OIDC-based authorization code flow for a service provider (SP), as per spec FC Service Provider specification <https://partenaires.franceconnect.gouv.fr/fcp/fournisseur-service> - The various endpoints to interact with and the callbacks to expose are fully described along w/ the related flows – This encompasses both the sign-ins and the logout capabilities -. | PRI1 |
| 2 | Direct support from the BizApps portals via identity-based native capabilities, See [Overview of authentication in Power Apps portals - Power Apps | Microsoft Docs](https://docs.microsoft.com/en-us/powerapps/maker/portals/configure/configure-portal-authentication). Further refer as to Scenario #1 | PRI1 |
| 3 | Complete support from with Azure AD B2C user flows. See [Ibid](https://docs.microsoft.com/en-us/powerapps/maker/portals/configure/configure-portal-authentication). Further refer as to Scenario #2. Also applies for any web application registered in the Azure AD tenant. | PRI1 |
| 4 | Complete support from with Azure AD B2C custom policies for any web application registered in the Azure AD tenant. | PRI2 |
| 5 | [insert feature/capability title here] | [insert priority here] |

## Out of scope

*This section gives a summary of what is out of scope. This can be a placeholder for upcoming work or for strictly defining what is not getting done as part of this spec.*

*Complete the table below to describe the features that are out of scope and set their priority.*

|  |  |
| --- | --- |
| # | Title |
| 1 | Integration with the so-called [FranceConnect+](https://franceconnect.gouv.fr/france-connect-plus) platform (FC+P) and the support of the related OIDC-based authorization code flow. See FC+P Service Provider specification: [https://github.com/france-connect/Documentation-FranceConnect-Plus/blob/main/fs/docs-fs.md](https://nam06.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Ffrance-connect%2FDocumentation-FranceConnect-Plus%2Fblob%2Fmain%2Ffs%2Fdocs-fs.md&data=04%7C01%7CPhilippe.Beraud%40microsoft.com%7C14cddd7aaa2c4a1145e308d9d9a352d3%7C72f988bf86f141af91ab2d7cd011db47%7C1%7C0%7C637780117765655732%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=Ml5racb%2FyAdr9DpPtgdALIiQpDVY%2Bz7R0x%2B1z9o0XIk%3D&reserved=0) |
| 2 | Integration with the so-called [AgentConnect](https://agentconnect.gouv.fr/) platform (ACP) and the support of the related OIDC-based authorization code flow. See ACP Service Provider specification: [Documentation-AgentConnect/doc-fs.md at main · france-connect/Documentation-AgentConnect (github.com)](https://github.com/france-connect/Documentation-AgentConnect/blob/main/doc-fs.md) |
| 3 | [insert feature/capability title here] |

# Risks and assumptions

*This section identifies risks and assumptions that pertain to the custom solution. Be very explicit on what are the risks and open issues.*

*In this table, list the various risks and assumptions associated with the project.*

|  |  |
| --- | --- |
| # | Title |
| 1 | To interact with FCP, a key (client ID and secret) is required. The obtention of such a key required a prior registration to FranceConnect registration, see <https://partenaires.franceconnect.gouv.fr/monprojet/cadrage>.  Please note that an integration key for public use allows to start the FCF implementation and the testing without having to file:  ID : '211286433e39cce01db448d80181bdfd005554b19cd51b3fe7943f6b3b86ab6e'  secret : '2791a731e6a59f56b6b4dd0d08c9b1f593b5f3658b9fd731cb24248e2669af4b'   * This key is for limited use and does not benefit from any support from the FranceConnect team. * The key proposed in free access and configured with Callback URLs only in localhost can ONLY be used on integration platform for FCP, see section “Nos Endpoints” in <https://partenaires.franceconnect.gouv.fr/fcp/fournisseur-service>   It is assumed here that the envisaged façade can be developed and tested locally on a developer machine. This may require the use of tooling like [ngrok](https://ngrok.com/). |
| 2 | [insert risk or assumption here] |
| 3 | [insert risk or assumption here]] |

# Dependencies

*Use this section to identify any dependencies from both an application and a platform perspective to complete this custom solution.*

*In this table, list the various dependencies associated with the project.*

|  |  |
| --- | --- |
| # | Title |
| 1 | The proposed solution, i.e., the façade, should leverage the code base of the FC service provider’s technical canvas as part of the open-source FranceConnect starter kit, see <https://aka.ms/FranceConnect>. |
| 2 | Regardless of the targeted environment (see section Targeted environment below), [Azure Key Vault](https://www.bing.com/ck/a?!&&p=39d927b7156656b215efe68296ef913553429313403bfbd4a4445c8165341a88JmltdHM9MTY1NjQyNTc1MSZpZ3VpZD0yOTEzNjBiYi0zN2YwLTRmNzUtYjkzZC0xNjkyN2JhMmU3YzgmaW5zaWQ9NTI0Mg&ptn=3&fclid=d04bbb5b-f6ec-11ec-bcc1-37b42f7be5b4&u=a1aHR0cHM6Ly9henVyZS5taWNyb3NvZnQuY29tL2VuLXVzL3NlcnZpY2VzL2tleS12YXVsdC8&ntb=1) will be used to both generate/import the RSA signing key (RS256) of FCF and sign the JWT token to issue. |
| 3 | Depending on the targeted environment (see section Targeted environment below), the lightweight façade may leverage additional services in the Azure control plane for security, management, and governance purposes. To name a few:   * [Azure Application Gateway](https://azure.microsoft.com/en-us/services/application-gateway/#overview) to load balance the traffic to the façade, and benefit from an integrated Web Application Firewall (WAF) service that provides centralized protection of the web façade from common exploits and vulnerabilities. * [Microsoft Defender for Cloud](ttps://azure.microsoft.com/en-us/services/defender-for-cloud/#overview) to assess and strengthen the security configuration of the facade * [Azure Monitor](https://azure.microsoft.com/en-us/services/monitor/#overview) to collect, analyze, and act on telemetry data from the targeted environments, with notably:   + Application Insights for i) telemetry data to “keep an eye” on load, responsiveness, and the performance of the façade; and ii) the detection & diagnostics of failures if any, and the assessment of how users are affected.   + Azure Log Analytics for log data, and any other metrics data. |
| 4 | [insert dependency here] |

# The proposed changes

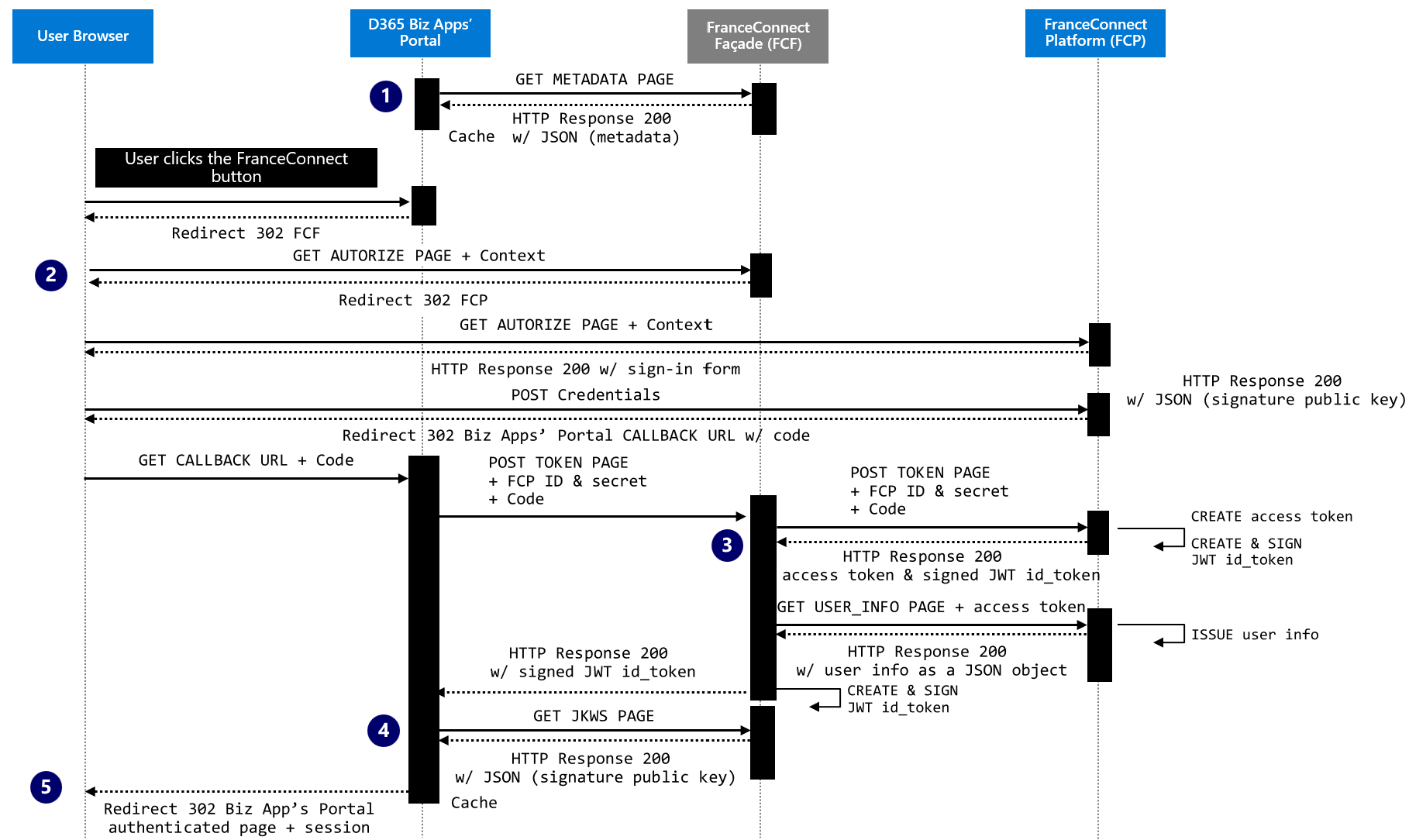
*Use this section to list the proposed changes in deeper detail to address the problem. Include screenshots, mocks and the UX flow, the protocol flow if any to list the changes and any new proposals.*

The façade must support the above-described integration paths from the Biz App’s perspective, i.e., i) direct integration vs. ii) via Azure AD B2C, which leads us to embrace the following two scenarios both from the sign-in and the logout perspectives.

## Changes for the login experience

### Scenario #1: direct support

The section describes the envisaged flow resulting from the use of FCF in between a Biz Apps’ portal and FCP to address the above-discussed issues. This proposed flow should be analyzed in the light of the one exposed in section **Direct support w/ authorization code flow**.



The following covers the key functionalities of FCF for this scenario:

1. The façade will expose its own discovery endpoint so that the portal can retrieve an **auto-discovery document. The related** metadata file will be dynamically built to specify the followings:
2. **URL of the Authorize page exposed by FCF.**
3. **URL of the Token page exposed by FCF.**
4. **URL of the JKWS page exposed by FCF to retrieve the FCF RS256 public signature key.**
5. When invoked by the portal to sign in the user, the FCF **Authorize page** will filter the URL parameters, before redirecting in turn the call to the FCP Authorize page only with the authorized parameters as per FC SP specification. **Once authenticated, the user will be redirected back to the portal’s Callback URL configured in FCP and passed in parameter during both the** FCF **Authorize page** call, and in turn the **FCP Authorize page call.**
6. To retrieve the id\_token from the obtain code, the portal will call the above FCF Token page with the key (client ID and secret) obtained as part as the FranceConnect registration, or instead the integration key for public use.

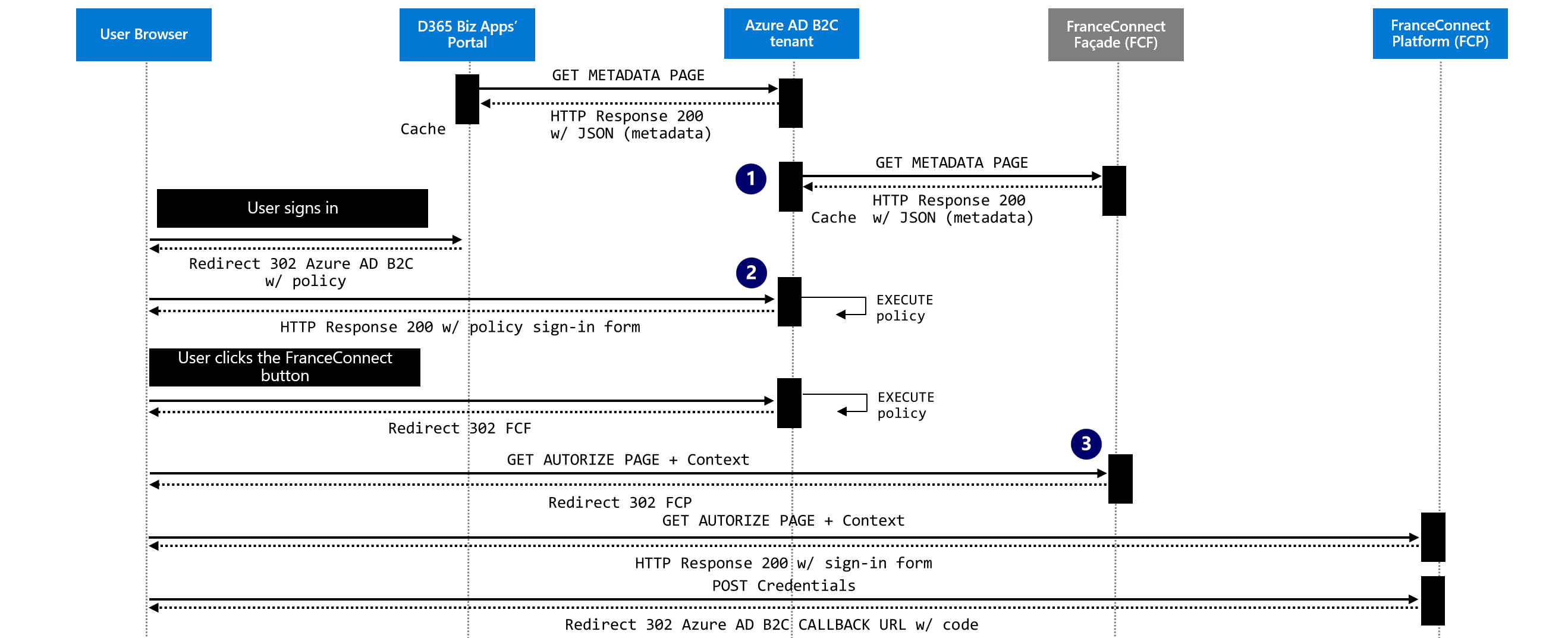
When invoked by the portal, the FCF Token page will in turn invoke the FCP Token page with the same key to get both an access token and the id\_token issued by FCP. The access token will be used in turn to query the FCP USER\_INFO endpoint, and thus retrieve the user info as a JSON object to complete the so-called ‘pivot identity’ of the user or a subset of it depending on the scopes being specified.

The information contained in the id\_token via the use the FCP HS256 key and the user info will then be merged into a JWT token created and signed with the FCF private key.

1. The FCF public key is available through a call to the FCF JKWS page. The related is generated on purpose so that the portal can validate the signature of the forged JWT token.

### Scenario #2: via Azure AD B2C

Likewise, the section describes the envisaged flow resulting from the use of FCF in between the Azure AD B2C and FCP to address the above-discussed issues. This proposed flow should be analyzed in the light of the one exposed in section **Support via Azure AD B2C**.



Une image contenant texte

Description générée automatiquement

The following covers the key functionalities of FCF for this scenario:

1. As this is the same façade being leveraged in this second scenario, it will still expose its own discovery endpoint so that this time Azure AD B2C, and more specifically **the IEF orchestration platform**, can retrieve an **auto-discovery document. The** dynamically built metadata remains the exact same**.**
2. **A custom** sign-in page will be defined as part of the user flow, a.k.a. a policy, definition. This page will include the mandatory so-called FranceConnect button to initiate the user authentication against FCP.
3. As before, but this time when invoked by **the IEF orchestration platform** as part of the defined user journey, the FCF **Authorize page** will filter the URL parameters, before redirecting in turn the call to the FCP Authorize page only with the authorized parameters as per FC SP specification. **Once authenticated, the user will be redirected back to the IEF’s Callback URL configured in FCP and passed in parameter during both the** FCF **Authorize page** call, and in turn the **FCP Authorize page call.**
4. Similarly, to retrieve the id\_token from the obtain code, **the IEF orchestration platform** will call the above FCF **Token** page with the FCP key (client ID and secret). As before, when invoked by **the IEF orchestration platform**, the FCF Token page will in turn invoke the FCP Token page with the same key to get both an access token and the id\_token issued by FCP. The access token will be used in turn by the orchestration engine to query the FCP USER\_INFO endpoint, and thus retrieve the user info as a JSON object to complete the so-called ‘pivot identity’ of the user or a subset of it depending on the scopes being specified.

The information contained in the id\_token via the use the FCP HS256 key and the user info will then be merged into a JWT token created and signed with the FCF private key and returned to IEF.

1. The FCF public key is available through a call to the FCF JKWS page. The related is generated on purpose so that **the IEF orchestration platform** can validate the signature of the forged JWT token. **After completing the orchestration steps specified in the policy, the IEF orchestration platform issues, signs, and returns yet another id\_ token to the portal.**
2. **To validate the signature of the received id\_token, the portal downloads the Azure AD B2C Tenant’s public key. The user information is extracted, and a user context is created for the user.**
3. **A session is eventually established on the portal and the requested content, i.e., an authenticated page, is provided to the user.**

**One should note that the D365 Biz Apps’ portal can be replaced by any arbitrary web application registered in the Azure AD B2C tenant. In this case, this second scenario simply addresses the integration of FCP from an Azure AD B2C perspective. This also unlock the use of custom policies in lieu of user flows.**

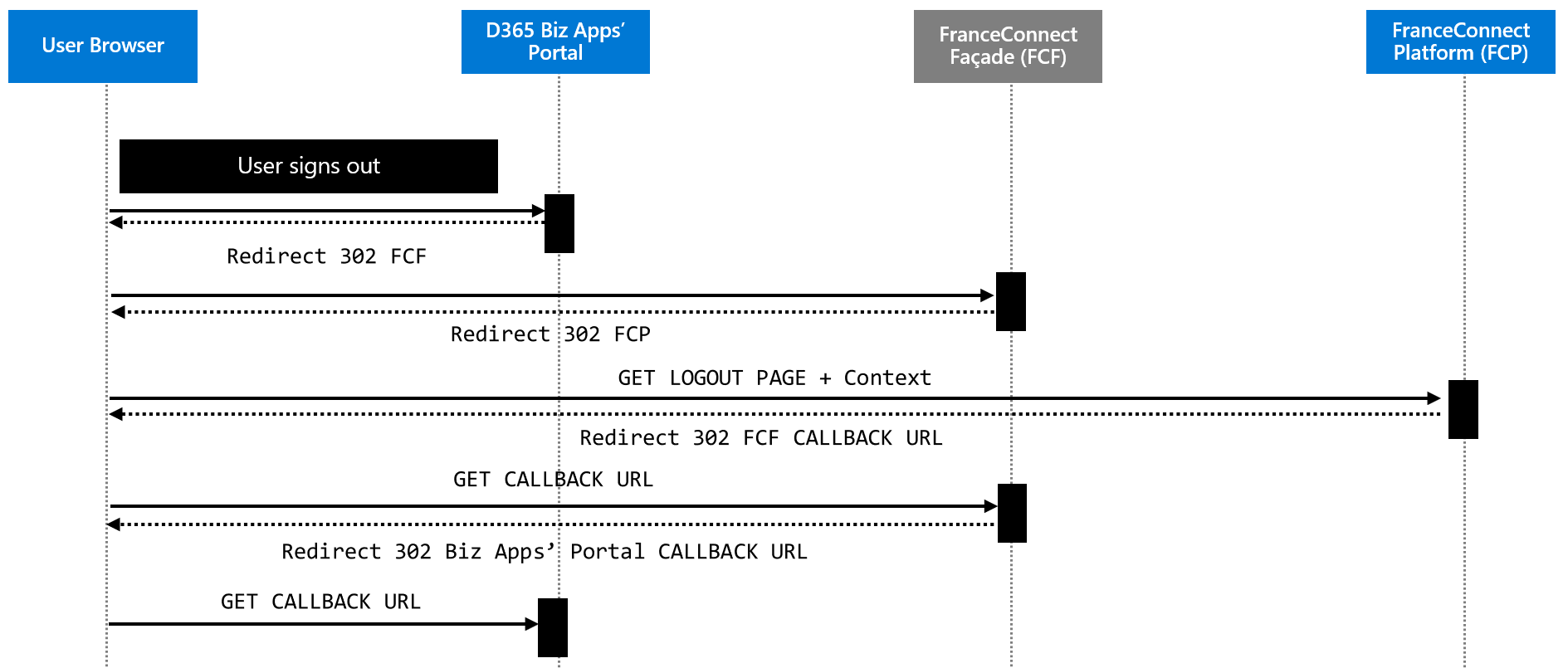
## Changes for the logout experience

Supporting the logout is also in scope for the lightweight façade as per section **In scope** above.

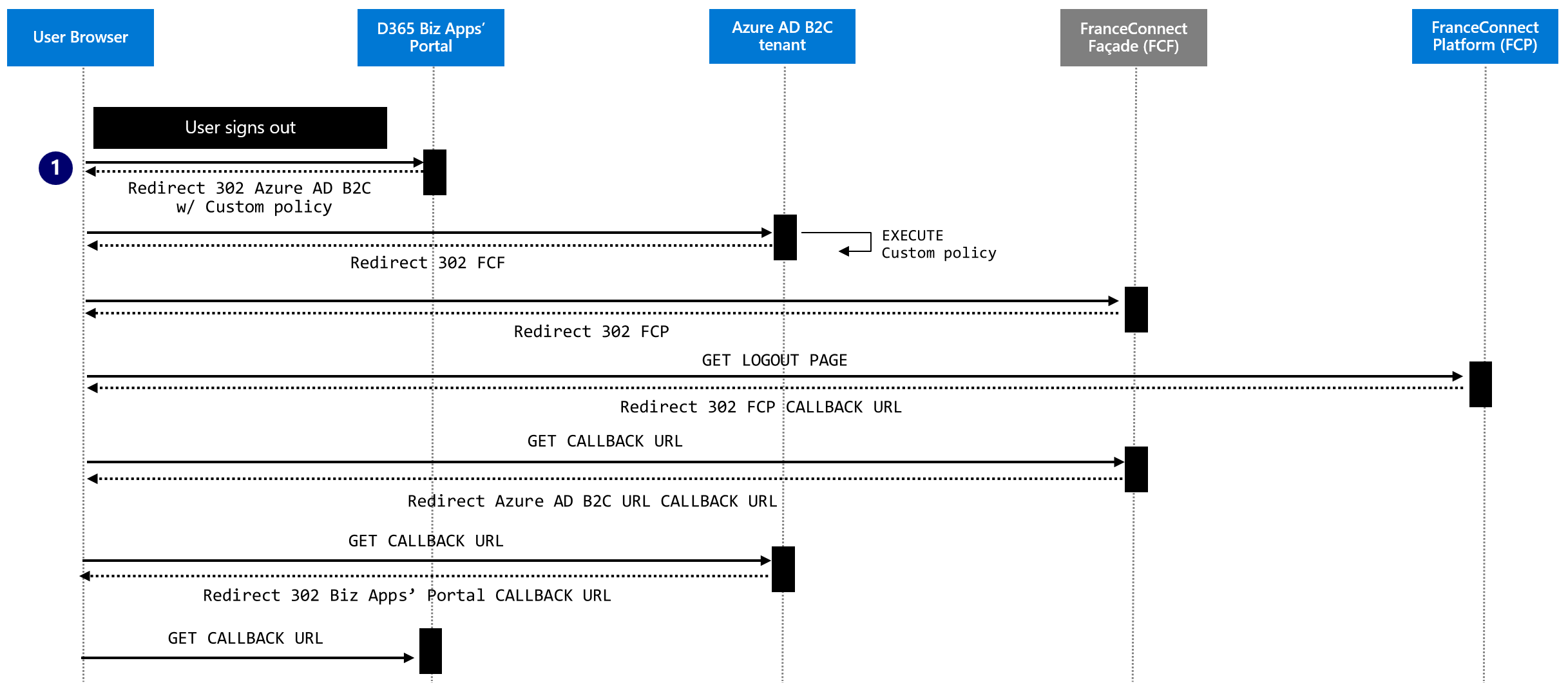
As per section A brief recap of services’ support from the FranceConnect platform above, the initial id\_token issued by FCP as part of the user login experience must be communicated back to FCP for the logout. The changes must ensure that this token is appropriately cached so that it can be retrieved later when the user signs out.

### Scenario #1: direct support

FCP do not initiate a sign-out. So, in the portal authentication configuration, The RP initiated logout setting must be disabled.



### Scenario #2: via Azure AD B2C



# Environments supported

*Use this section to document which environments the solution should work in/with.*

## Targeted environment

*Fill in the table below to describe all the environments the custom solution should support.*

|  |  |
| --- | --- |
| # | Environment |
| 1 | Azure public cloud |
| 2 | Customer on-premises environment |
| 3 | Any other suitable location through an [Azure Arc](https://docs.microsoft.com/en-us/azure/azure-arc/overview) hosting, w/ [custom locations on top of Azure Arc-enabled Kubernetes](https://docs.microsoft.com/en-US/Azure/azure-arc/kubernetes/conceptual-custom-locations). |
| 4 | [insert dependency here] |

## Prerequisites for a development environment

*Use this section to document all the prerequisites that pertains to the development environment if any.*

As per section **Risks and assumptions** above, the testing environment should be set locally to leverage as much as possible the available integration key for public use without having to file anything.

This may require the use of the [Postman API platform](https://www.postman.com/) for building this façade. Postman indeed simplifies each step of the API lifecycle and streamlines collaboration. As such, the definition of a Postman collection can adequately streamline the interaction with the façade being developed, and thus initiate the whole ‘identity dance’ regardless of the considered above-mentioned scenario, mimic the missing part, etc.

## Prerequisites for a testing environment

*Use this section to document all the prerequisites that pertains to the testing environment if any.*

## Prerequisites for the targeted environment

*Use this section to document all the prerequisites that pertains to the target environment if any as per section* Targeted environment *above.*

### Environment #1

Microsoft Azure is the first targeted environment. In fact, **the initial version of FCF will target this sole environment beyond the local development environment.**

To not only allow the façade deployment in a reproductible, predictable, and secure manner, but also the ability to in turn operate and govern it, a suitable application landing zone should be defined for the lightweight façade, see [What is an Azure landing zone? - Cloud Adoption Framework | Microsoft Docs](https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/landing-zone/).

As such, the Microsoft Cloud Adoption Framework for Azure (CAF) Migration landing zone blueprint is a set of infrastructure to help manage the cloud estate in alignment with CAF and can serve as a basis to define a suitable blueprint for the lightweight façade.

See:

* [Overview of Azure Blueprints - Azure Blueprints | Microsoft Docs](https://docs.microsoft.com/en-us/azure/governance/blueprints/overview)
* [CAF Migration landing zone blueprint sample overview - Azure Blueprints | Microsoft Docs](https://docs.microsoft.com/en-us/azure/governance/blueprints/samples/caf-migrate-landing-zone/)

This environment is composed of several Azure services used to provide a secure, fully monitored, enterprise-ready governance, and is notably composed of an Azure Key Vault instance used to host the key(s), the deployment of a Log Analytics workplace, etc., see dependencies #2 and #3 as per section Dependencies above.

A set of Bicep scripts will be provided. Bicep is a domain-specific language (DSL) that uses declarative syntax to deploy Azure resources. In a Bicep file, is defined the infrastructure to deploy to Azure, and then such a file can be (re)used throughout the development lifecycle to repeatedly deploy the intended infrastructure. The described resources are deployed in a consistent manner. Bicep provides concise syntax, reliable type safety, and support for code reuse. It offers a first-class authoring experience for your infrastructure-as-code solutions in Azure. See [Bicep language for deploying Azure resources.](https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/overview?tabs=bicep)

# Reason to choose an inferior solution (optional)

*If an inferior solution/workflow is being chosen due to any considerations (design, engineering limitations, bandwidth etc.), it needs to be listed here.*

**N/A for the initial version of FCF.**

# Code base for the proposed solution

*Use this section to describe how the custom solution should be developed in terms of languages, frameworks, and programming environment.*

**The envisaged façade for FCF will be implemented as a Web API with ASP.NET Core 6.0 to enable communication and/or interaction with all the software components outlined above with each other as part of the ‘identity dance’ to sustain the envisaged scenario.**

**One should stress that other option could fulfill the above prerequisites and requirements, such as Azure Functions to name one.**

## Motivations

Web API is the “enhanced” form of the web application technical canvas provided as part of the so-called FranceConnect Starter Kit. It will act as an HTTP based interface which accepts above-described HTTP requests from the initiating third party application, namely the Biz Apps’ portal, the Power App portals, or Azure AD B2C, and performs related operations against FCP as explained before.

Interestingly enough, the API can be easily tested on Swagger by default whenever the project is executed.

Furthermore, ASP.NET Core allows to configure the façade behavior based on the runtime environment using an environment variable to accommodate the specificities if any of the above development, testing, and production environment, see [Use multiple environments in ASP.NET Core | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/environments?view=aspnetcore-6.0).

Eventually, the API can be seamlessly instantiated in various environments from the on-premises to the cloud in Azure or cloud environment *as per section* Targeted environmentabove:

* ASP.NET Core indeed offers the platform independence. It supports the deployment on Windows, Linux and MacOS environments as well by means of a hosting bundle.
* The façade can be published as self-contained deployment that will include the .NET runtime and libraries, and the façade and its dependencies to run without any issues; be it a Windows, Linux, or Mac environment. It can run on a machine that doesn't have the .NET runtime installed. Alternatively, the façade can also be published as framework-dependent deployment that will includes only your application itself and its dependencies. The .NET runtime should be separately installed. These two different modes provide a great level of flexibility.

All of the above motivate this choice of developing a Web API with ASP.NET Core 6.0 for implementing the envisaged lightweight façade.

## Solution endpoints

The solution must implement a series of endpoints to act as a “compatible” FCP from the caller perspective and be transparent from the FCP one.

### Metadata endpoints

*Fill in the table below to describe all the endpoints the custom solution should support.*

The lightweight façade must implement the following endpoints for the OpenID Connect Discovery as per [Final: OpenID Connect Discovery 1.0 incorporating errata set 1](https://openid.net/specs/openid-connect-discovery-1_0.html) standard:

|  |  |  |  |
| --- | --- | --- | --- |
| # | Method | Endpoint | Comments |
| 1 | GET | /common/.well-know/openid-configuration | See Addendum for details |
| 2 | GET | /common/discovery/beta/keys | See Addendum for details |

### OpenID Connect endpoints

*Fill in the table below to describe all the endpoints the custom solution should support.*

The lightweight façade must implement the following endpoints for the OpenID Connect protocol support and the adaptation layer between the caller of the façade and FCP, as per [OpenID Connect Core 1.0 incorporating errata set 1](https://openid.net/specs/openid-connect-core-1_0.html) standard:

|  |  |  |  |
| --- | --- | --- | --- |
| # | Method | Endpoint | Comments |
| 1 | GET | /api/<*version*>/authorize | See Addendum for details |
| 2 | POST | /api/<*version*>/token | See Addendum for details |
| 3 | GET | /api/<*version*>/logout | See Addendum for details |

**As per this specification version, current version is beta.**

# Metrics

*List the metrics being tracked as a success criteria for this custom solution. A “Green Line” needs to be defined which would be the ship criteria for each metric of the project.*

|  |  |  |
| --- | --- | --- |
| # | Metrics | PRI |
| 1 | [insert metric type here] | [insert purpose here] |
| 2 | [insert metric type here] | [insert purpose here] |
| 3 | [insert metric type here] | [insert purpose here] |

**No specific metrics are defined at this stage for the initial version of FCF.**

# Telemetry

*Use this section to describe the telemetry the custom solution should capture and how the telemetry is used.*

*Fill in the table below to describe the telemetry that needs to be captured and the purpose it is used for.*

|  |  |  |
| --- | --- | --- |
| # | Telemetry | Purpose |
| 1 | [insert telemetry type here] | [insert purpose here] |
| 2 | [insert telemetry type here] | [insert purpose here] |
| 3 | [insert telemetry type here] | [insert purpose here] |

**Telemetry will NOT be implemented in this initial version of FCF.**

# Error reporting

*Use this section to describe how errors are handled and displayed to user throughout the custom solution.*

Acting as an OpenID Connect (OIDC) identity provider from FCF’s perspective, FCP can potentially return all kinds of errors to a client application, here FCF.

To do so, FCP relies on the errors’ return mechanism as described in the [OpenID Connect Core 1.0 incorporating errata set 1](https://openid.net/specs/openid-connect-core-1_0.html) standard for an identity provider (see below), and proposes in this context an error code system to make the detection and management of bugs easier.

|  |  |
| --- | --- |
| Error Approach | Notes |
| [3.1.2.6 Authentication Error Response](http://openid.net/specs/openid-connect-core-1_0.html#AuthError) | - |
| [3.1.3.4 Token Error Response](http://openid.net/specs/openid-connect-core-1_0.html#TokenErrorResponse) | - |
| [5.3.3 UserInfo Error Response](https://openid.net/specs/openid-connect-core-1_0.html#UserInfoError) | - |

**All the error codes supported by the above mechanism are by design passed to FCF, and in turn will be passed to the calling application. See the Addendum document for details.**

*Fill in the table below to describe how errors are displayed to users. If necessary, notate for each page if error handling is different between pages.*

|  |  |
| --- | --- |
| Error Approach | Notes |
|  |  |
|  |  |

**No additional and specific error code will be defined and implemented in this initial version of FCF.**

# Additional references

*Use this section to provide links to any supporting documentation or related content. You may also embed attachments here.*

[Overview of authentication in Power Apps portals - Power Apps | Microsoft Docs](https://docs.microsoft.com/en-us/powerapps/maker/portals/configure/configure-portal-authentication)

[Configure an OpenID Connect provider for portals - Power Apps | Microsoft Docs](https://docs.microsoft.com/en-us/powerapps/maker/portals/configure/configure-openid-provider)

[FAQs for using OpenID Connect in portals - Power Apps | Microsoft Docs](https://docs.microsoft.com/en-us/powerapps/maker/portals/configure/configure-openid-faqs)

[Configure the Azure Active Directory B2C provider (using interface in preview) - Power Apps | Microsoft Docs](https://docs.microsoft.com/en-us/powerapps/maker/portals/configure/configure-azure-ad-b2c-provider)

[Configure the Azure Active Directory B2C provider manually - Power Apps | Microsoft Docs](https://docs.microsoft.com/en-us/powerapps/maker/portals/configure/configure-azure-ad-b2c-provider-manual)

[Customize the Azure AD B2C user interface for portals - Power Apps | Microsoft Docs](https://docs.microsoft.com/en-us/powerapps/maker/portals/configure/azure-ad-b2c)

[Web sign in with OpenID Connect - Azure Active Directory B2C | Microsoft Docs](https://docs.microsoft.com/en-us/azure/active-directory-b2c/openid-connect#send-a-sign-out-request)

[Set up sign-up and sign-in with OpenID Connect - Azure AD B2C | Microsoft Docs](https://docs.microsoft.com/en-us/azure/active-directory-b2c/identity-provider-generic-openid-connect?pivots=b2c-user-flow) (user flows)

[Set up sign-up and sign-in with OpenID Connect - Azure AD B2C | Microsoft Docs](https://docs.microsoft.com/en-us/azure/active-directory-b2c/identity-provider-generic-openid-connect?pivots=b2c-custom-policy) (custom policies)

[Authorization code flow - Azure Active Directory B2C | Microsoft Docs](https://docs.microsoft.com/en-us/azure/active-directory-b2c/authorization-code-flow)

[Define an OpenID Connect technical profile in a custom policy - Azure AD B2C | Microsoft Docs](https://docs.microsoft.com/en-us/azure/active-directory-b2c/openid-connect-technical-profile)

[Configure session behavior - Azure Active Directory B2C | Microsoft Docs](https://docs.microsoft.com/en-us/azure/active-directory-b2c/session-behavior?pivots=b2c-user-flow#secure-your-logout-redirect) (user flows)

[Configure session behavior - Azure Active Directory B2C | Microsoft Docs](https://docs.microsoft.com/en-us/azure/active-directory-b2c/session-behavior?pivots=b2c-custom-policy#single-sign-out) (custom policies)

[Overview of Azure Blueprints - Azure Blueprints | Microsoft Docs](https://docs.microsoft.com/en-us/azure/governance/blueprints/overview)

[CAF Migration landing zone blueprint sample overview - Azure Blueprints | Microsoft Docs](https://docs.microsoft.com/en-us/azure/governance/blueprints/samples/caf-migrate-landing-zone/)

Bicep language for deploying Azure resources

1. After the launch of Power Apps portals on October 1, 2019, the full capabilities of Dynamics 365 Portals, previously offered only as an add-on to customer engagement apps (Dynamics 365 Sales, Dynamics 365 Customer Service, Dynamics 365 Field Service, Dynamics 365 Marketing, and Dynamics 365 Project Service Automation), are now available standalone in Power Apps. In other word, all Dynamics 365 Portals are now referred to as Power Apps portals. [↑](#footnote-ref-2)
2. Effective October 12, 2022, Power Apps portals are Power Pages. See [Microsoft Power Pages is now generally available](https://powerpages.microsoft.com/en-us/blog/microsoft-power-pages-is-now-generally-available/). [↑](#footnote-ref-3)