Onboarding to the KBC Sandbox

Onboarding to the KBC sandbox kbc.openbankingapittesting.com

This document describes what will be required for Third Party Provider (TTP) development personnel to start using KBC's Open Banking Implementation Entity (OBIE) compliant PSD2 APIs. There is extensive and detailed documentation on https://www.openbanking.org.uk/ which should be read in association with this document.

Before you Start

There are a number of things you will need before you start.

1. You must first sign up to the KBC developer portal at https://kbcireland.dev.ca.com/admin/app/registration
2. Once you have verified your email address, you will then need to register your application on the portal. Once the application has been approved, the following will be made available to you to start interacting with the sandbox:
    a. An Oauth Client ID
    b. An Oauth Client Secret
    c. An Oauth userid which will be the same as the Client ID.
    d. An Oauth password for the user.
3. You will need the following certificates which you will find in appendix A of this document:
    a. The root and intermediate certificates to connect to kbc.openbankingapitesting.com
    b. The public certificate to check signatures on signed messages from the sandbox. This is supplied in .pem format and as a JSON Web Key (JWK)
4. Finally, you will need to create transport and signing certificates as documented in the Environment Setup document.

Please refer to the Environment Setup document if you wish to use Postman to view and interact with the APIs. Note that due to security requirements, it is difficult to use these APIs without having any experience with Oauth tokens. This document is intended to enable you to programmatically work with the APIs and the Oauth server.

Oauth and Openid principles

For good reason, the security profile defined by the Open Banking Implementation Entity (OBIE) is critical in terms of securing the access to banking information and the processing of payments. There are many good resources available on the Internet on the details of these specifications, however, here we just want to give you a basic feeling for what is required of a TPP when interacting with the OBIE defined APIs and security profile.

Access Tokens

This is the key element used to control access to the APIs and accounts. This identifies who you are and determines whether the system will allow you as an application to do what you want to do. Access tokens are retrieved from the Oauth/Openid server and are generally 'short lived'. This means that they are will expire in seconds or minutes after they have been retrieved.

Refresh Tokens

These are tokens generally issued with an Access Token and are 'long lived' in that they can remain active for 90 days or more depending on the policies of the Oauth server you are using. The general idea is that you can easily get an access token based on an existing refresh token with a simple call to the Oauth server. As such, it is essential that you maintain a record of any refresh or access tokens issued for a given user and for the TPP itself.

Roles

There are primarily two roles that are in play when using these APIs.

1. The TPP is identified by their application Client ID and Client Secret. The TPP will use what is called a 'client_credentials' grant request to get a token to represent requests from the TPP. This is a relatively simply HTTP POST request which will return an access token and a refresh token for the TPP to use in subsequent calls.
2. The second role identifies a PSU (essentially a user of the TPP's service). Ultimately a key part of the security profile is that a PSU explicitly consents to activity on their accounts. This requires that the PSU logs on to the bank's Oauth server implementation using their banking credentials to approve or deny access to consent requests set up by TPPs. The userid and password set up as part of your application is what is required in this case. This is managed using an Oauth Authorize request which redirects to the Oauth server to enable you to enter in the userid and password and consent to the request. Once you have consented, you will be redirected back to your application based on the redirect provided on the authorise request. This will include a short lived 'code' which can then be exchanged for an access token and refresh token to represent that user. This is then provided on the API request to identify the user.

In both cases, the token is provided to the API using the HTTP 'Authorisation Header set to the following:

Bearer <access token>

While it's possible to manage the life time of an access token, it is always possible that even when you believe it is still active, by the time it gets to the API it may have expired so each call needs to:

1. Tolerate an error indicating that the access token is invalid/has expired.
2. If a refresh token is available, get a new access token based on that and reissue the request.
3. If no refresh token is available or the refresh token has expired or is invalid, you must go through the redirect process again to have the PSU login and consent to the TPP accessing your data.

Scopes

When requesting a token from the Oauth server, you need to tell it what you want to token for. The following scopes are required when accessing the relevant APIs:

- 'accounts' scope is required to access the AIS endpoints.
- 'payments' scope is required to access the PIS endpoints.
- 'fundsconfirmations' scope is required to access the CBPII endpoints.

If an incorrect scope is provided in an access token for a call to an endpoint, it will be rejected.

Consent Processing

A key requirement of the security specification is that a PSU explicitly consents to a specific request. The process is generally as follows:

1. The TPP gets a 'client credential' based on the Client ID and Client secret.

2. The TPP uses a consent API to create a consent object for that PSU.
3. The TPP will be returned a unique consent id for the consent object which will have a status of ‘Awaiting Authorisation’.
4. The TPP must then redirect the PSU to the authorization endpoint with a signed request object which will include the consent id to be managed.
5. The Oauth implementation must then force the PSU to logon using their credentials.
6. Once the PSU has successfully logged on, the Oauth implementation will present the specific consent that is being requested.
7. The PSU can reject it in which case the consent will by marked as rejected.
8. If the PSU accepts the consent, the consent will be marked as ‘Authorized’ and the PSU will be redirected back to the TPP’s application.
9. The TPP will receive an authorization code which can then be exchanged for an access token and refresh token.
10. The TPP can either use the consent to:
    a. Enable access to the AIS APIs if the consent object was related to account access.
    b. Can issue funds confirmation checks using the appropriate API if the consent was a funds confirmation consent object.
    c. Can initiate the payment when the consent was for a Domestic Payment.

It is incumbent upon the TPP to remember any consents that have been created by a given PSU as there may be cases where the PSU will only authorize the consent at some future time.

Appendix A – Certificates

Root and Intermediate Certificates for kbc.openbankingapitesting.com

```
subject=/C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com,
Inc./OU=http://certs.godaddy.com/repository//CN=Go Daddy Secure
Certificate Authority - G2
issuer=/C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com, Inc./CN=Go Daddy
Root Certificate Authority - G2
-----BEGIN CERTIFICATE-----
MIIE0DCCA7igAwIBAgIBBzANBgkqhkiG9w0BAQsFADCBgzELMAkGA1UEBhMCVVMx
EDAOBgNVBAgTB0FyaXpvbmExEzARBgNVBAcTClNjb3R0c2RhbGUxGjAYBgNVBAoT
EUdvRGFkZHkuY29tLCBJbmMuMTEwLwYDVQQDEyhHbyBEYWRkeSBSb290IENlcnRp
ZmljYXRlIEF1dGhvcml0eSAtIEcyMB4XDTExMDUwMzA3MDAwMFoXDTMxMDUwMzA3
MDAwMFowgbQxCzAJBgNVBAYTAlVTMRAwDgYDVQQIEwdBcml6b25hMRMwEQYDVQQH
EwpTY290dHNkYWxlMRowGAYDVQQKExFHb0RhZGR5LmNvbSwgSW5jLjEtMCsGA1UE
CxMkaHR0cDovL2NlcnRzLmdvZGFkZHkuY29tL3JlcG9zaXRvcnkvMTMwMQYDVQQD
EypHbyBEYWRkeSBTZWN1cmUgQ2VydGlmaWNhdGUgQXV0aG9yaXR5IC0gRzIwggEi
MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC54MsQ1K92vdSTYuswZLiBCGzD
BNliF44v/z5lz4/OYuY8UhzaFkVLVat4a2ODYpDOD2lsmcgaFItMzEUz6ojcnqOv
K/6AYZ15V8TPLvQ/MDxdR/yaFrzDN5ZBUY4RS1T4KL7QjL7wMDge87Am+GZHY23e
cSZHjzhHU9FGHbTj3ADqRay9vHHZqm8A29vNMDp5T19MR/gd71vCxJ1gO7GyQ5HY
pDNO6rPWJ0+tJYqlxvTV0KaudAVkV4i1RFXULSo6Pvi4vekyCgKUZMQWOlDxSq7n
eTOvDCAHf+jfBDnCaQJsY1L6d8EbyHSHyLmTGFBUNUtpTrw700kuH9zB0lL7AgMB
AAGjggEaMIIBFjAPBgNVHRMBAf8EBTADAQH/MA4GA1UdDwEB/wQEAwIBBjAdBgNV
HQ4EFgQUQMK9J47MNIMwojPX+2yz8LQsgM4wHwYDVR0jBBgwFoAUOpqFBxBnKLbv
9r0FQW4gwZTaD94wNAYIKwYBBQUHAQEEKDAmMCQGCCsGAQUFBzABhhhodHRwOi8v
b2NzcC5nb2RhZGR5LmNvbS8wNQYDVR0fBC4wLDAqoCigJoYkaHR0cDovL2NybC5n
b2RhZGR5LmNvbS9nZHJvb3QtZzIuY3JsMEYGA1UdIAQ/MD0wOwYEVR0gADAzMDEG
CCsGAQUFBwIBFiVodHRwczovL2NlcnRzLmdvZGFkZHkuY29tL3JlcG9zaXRvcnkv
MA0GCSqGSIb3DQEBCwUAA4IBAQAIfmyTEMg4uJapkEv/oV9PBO9sPpyIBslQj6Zz
91cxG7685C/b+LrTW+C05+Z5Yg4MotdqY3MxtfWoSKQ7CC2iXZDXtHwlTxFWMMS2
RJ17LJ3lXubvDGGqv+QqG+6EnriDfcFDzkSnE3ANkR/0yBOtg2DZ2HKocyQetawi
DsoXiWJYRBuriSUBAA/NxBti21G00w9RKpv0vHP8ds42pM3Z2Czqrpv1KrKQ0U11
GIo/ikGQI31bS/6kA1ibRrLDYGCD+H1QQc7CoZDDu+8CL9IVVO5EFdkKrqeKM+2x
LXY2JtwE65/3YR8V3Idv7kaWKK2hJn0KCacuBKONvPi8BDAB
-----END CERTIFICATE-----

subject=/C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com, Inc./CN=Go
Daddy Root Certificate Authority - G2
issuer=/C=US/O=The Go Daddy Group, Inc./OU=Go Daddy Class 2
Certification Authority
-----BEGIN CERTIFICATE-----
MIIEfTCCA2WgAwIBAgIDG+cVMA0GCSqGSIb3DQEBCwUAMGMxCzAJBgNVBAYTAlVT
MSEwHwYDVQQKExhUaGUgR28gRGFkZHkgR3JvdXAsIEluYy4xMTAvBgNVBAsTKEdv
IERhZGR5IENsYXNzIDIgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkwHhcNMTQwMTAx
MDcwMDAwWhcNMzEwNTMwMDcwMDAwWjCBgzELMAkGA1UEBhMCVVMxEDAOBgNVBAgT
B0FyaXpvbmExEzARBgNVBAcTClNjb3R0c2RhbGUxGjAYBgNVBAoTEUdvRGFkZHku
Y29tLCBJbmMuMTEwLwYDVQQDEyhHbyBEYWRkeSBSb290IENlcnRpZmljYXRlIEF1
dGhvcml0eSAtIEcyMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAv3Fi
CPH6WTT3G8kYo/eASVjpIoMTpsUgQwE7hPHmhUmfJ+r2hBtOoLTbcJjHMgGxBT4H
Tu70+k8vWTAi56sZVmvigAf88xZ1gDlRe+X5NbZ0TqmNghPktj+pA4P6or6KFWp/
```

```
3gvDthkUBcrqw6gElDtGfDIN8wBmIsiNaW02jBEYt9OyHGC0OPoCjM7T3UYH3go+
6118yHz7sCtTpJJiaVElBWEaRIGMLKlDliPfrDqBmg4pxRyp6V0etp6eMAo5zvGI
gPtLXcwy7IViQyU0AlYnAZG0O3AqP26x6JyIAX2f1PnbU21gnb8s51iruF9G/M7E
GwM8CetJMVxpRrPgRwIDAQABo4IBFzCCARMwDwYDVR0TAQH/BAUwAwEB/zAOBgNV
HQ8BAf8EBAMCAQYwHQYDVR0OBBYEFDqahQcQZyi27/a9BUFuIMGU2g/eMB8GA1Ud
IwQYMBaAFNLEsNKR1EwRcbNhyz2h/t2oatTjMDQGCCsGAQUFBwEBBCgwJjAkBggr
BgEFBQcwAYYYaHR0cDovL29jc3AuZ29kYWRkeS5jb20vMDIGA1UdHwQrMCkwJ6Al
oCOGIWh0dHA6Ly9jcmwuZ29kYWRkeS5jb20vZ2Ryb290LmNybDBGBgNVHSAEPzA9
MDsGBFUdIAAwMzAxBggrBgEFBQcCARYlaHR0cHM6Ly9jZXJ0cy5nb2RhZGR5LmNv
bS9yZXBvc2l0b3J5LzANBgkqhkiG9w0BAQsFAAOCAQEAWQtTvZKGEacke+1bMc8d
H2xwxbhuvk679r6XUOEwf7ooXGKUwuN+M/f7QnaF25UcjCJYdQkMiGVnOQoWCcWg
OJekxSOTP7QYpgEGRJHjp2kntFolfzq3Ms3dhP8qOCkzpN1nsoX+oYggHFCJyNwq
9kIDN0zmiN/VryTyscPfzLXs4Jlet0lUIDyUGAzHHFIYSaRt4bNYC8nY7NmuHDKO
KHAN4v6mF56ED71XcLNa6R+ghlO773z/aQvgSMO3kwvIClTErF0UZzdsyqUvMQg3
qm5vjLyb4lddJIGvl5echK1srDdMZvNhkREg5L4wn3qkKQmw4TRfZHcYQFHfjDCm
rw==
-----END CERTIFICATE-----
```

Public Certificate to Check the Signature on Signed Messages from the Sandbox

```
{
      "kty": "RSA",
      "e": "AQAB",
      "kid": "KBC-Mitreid-Demo",
      "n": "y-
s9C_8tzILEpyX0_eJMZLVJy3VfK3hLetAoN1v8EEciIRdQ7HVknVEsz22KI_yEme4thS
1QzPkno-hf9V6tb0J0jbmylTLrWb6ZRe4AJVllJAEiV7caOWQ-
kGhxmpEU7ceC9uErlPACK_z181Y5Gl90-
Lq3OCFhRI2dKV_lyzWJCZxa0a2Dqi1bNlRXND1le20z4GX_IpmCmjaRcbeYK9TU-
EihWYqSLDJVlED4dbGUUcw8ZJjL1pJbDH_QDLEYrOjVhE10CkUiEyzE9l18GZd1DLs70
265NNJ5IAZ0zCMVI8Ni40Wc2nNZGPtW9It53AdlJQ6HPO1QQA2oXHV2YQ"
}
```

```
-----BEGIN CERTIFICATE-----
MIIDHjCCAgYCCQDMIeEeRfDP5TANBgkqhkiG9w0BAQUFADBCMSEwHwYDVQQLExhE
b21haW4gQ29udHJvbCBWYWxpZGF0ZWQxHTAbBgNVBAMMFCoub3N0aWFzb2x1dGlv
bnMuY29tMB4XDTE4MTEyMjE2MjQ0NVoXDTIxMDgxNzE2MjQ0NVowYDELMAkGA1UE
BhMCSUUxHTAbBgNVBAoTFE9wZW4gQmFua2luZyBMaW1pdGVkMRcwFQYDVQQLEw5P
c3RpYXNvbHV0aW9uczEZMBcGA1UEAxMQS0JDLU1pdHJlaWQtRGVtbzCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAMvrPQv/LcyCxKcl9P3iTGS1Sct1Xyt4
S3rQKDdb/BBHIiEXUOx1ZJ1RLM9tiiP8hJnuLYUtUMz5J6PoX/VerW9CdI25spUy
61m+mUXuACVZZSQBIle3GjlkPpBocZqRFO3HgvbhK5TwAiv89fNWORpfdPi6tzgh
YUSNnSlf5cs1iQmcWtGtg6otWzZUVzQ9ZXttM+Bl/yKZgpo2kXG3mCvU1PhIoVmK
kiwyVZRA+HWxlFHMPGSYy9aSWwx/0AyxGKzo1YRNdApFIhMsxPZdfBmXdQy7O9Nu
uTTSeSAGdMwjFSPDYuNFnNpzWRj7VvSLedwHZSUOhzztUEANqFx1dmECAwEAATAN
BgkqhkiG9w0BAQUFAAOCAQEAArQcG/J1JOUm+G5M+qlgmpRggbNsBElF4V1D/MWt
MR84fpzdXu7dQO7OFgbLED5hYAdNwpp4CXx/OXeCbYwP29Q3RDqriW5ASvqGr/l+
CgheZbYg5xd9gMkPlDV4f0sh36VpP72I2VMQontq38+BvzmnmfMNfRyfpRjZurRX
Q/sOkS8Gwj/yvuttK94N0WgOukwXBnn3PSghqlw4QEs3pXmW2KldJk3fuxmXyxZc
AzQpJDVfKYKQYtHun7L671oKG56hAtuD3dVzH2+236asSyW53/VrnQ9JjzcVmzrg
```

VI3YYB04GA9fL8hbTP+GZTR0tBltzmUsrfG31ThxDQ1r8Q==
-----END CERTIFICATE-----

Appendix B – Various Issues

1. When generating the SSL key and Certificate Signing Request (CSR) for your requests, the transport certificate must have a CN name which is the same as the Client ID issued for your application. For example, if the Client id is “l7c91070b6b6e54134950f9e3dc21c530b”, the ‘subj’ in the certificate generation request must be as follows for the transport certificate: -subj "/C=IE/O=Open Banking Limited/OU=yourcompany/CN=l7c91070b6b6e54134950f9e3dc21c530b"
2. Signing and checking signatures can be problematic. The OBIE have mandated that the payload is removed from the signed JSON Web Token (JWT) that is used as a security feature such that the receiving code can rebuild the actual message to reinsert into the JWT. It is essentially that the exact string sent by the client (or the Sandbox) be inserted or else the signature validation will fail. This could be additional blanks, line feeds or perhaps a different date format.