UK Open Banking Configuration Guide
Oracle Banking Digital Experience
Patchset Release 22.2.5.0.0

Part No. F72987-01

October 2024

ORACLE

UK Open Banking Configuration Guide

October 2024

Oracle Financial Services Software Limited

Oracle Park

Off Western Express Highway

Goregaon (East)

Mumbai, Maharashtra 400 063

India

Worldwide Inquiries:

Phone: +91 22 6718 3000

Fax:+91 22 6718 3001

www.oracle.com/financialservices/

Table of Contents

1. Preface

1.1 Purpose

Welcome to the User Guide for Oracle Banking Digital Experience. This guide explains the operations that the user will follow while using the application.

1.2 Audience

This manual is intended for Customers and Partners who setup and use Oracle Banking Digital Experience.

1.3 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit, http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1.4 Critical Patches

Oracle advises customers to get all their security vulnerability information from the Oracle Critical Patch Update Advisory, which is available at Critical Patches, Security Alerts and Bulletins. All critical patches should be applied in a timely manner to ensure effective security, as strongly recommended by Oracle Software Security Assurance.

1.5 Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1.6 Conventions

The following text conventions are used in this document:

Convention	Meaning

boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
Italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1.7 Screenshot Disclaimer

The images of screens used in this user manual are for illustrative purpose only, to provide improved understanding of the functionality; actual screens that appear in the application may vary based on selected browser, theme, and mobile devices.

1.8 Acronyms and Abbreviations

The list of the acronyms and abbreviations that you are likely to find in the manual are as follows:

Abbreviation	Description
OBDX	Oracle Banking Digital Experience

2. Objective and Scope

Background

Open Banking Configuration Document provides the various configurations required to enable UK Open Banking in OBAPI

Scope

- Headers Configuration
- Properties
- OAuth Configuration
- Code Convention and Extensibility

3. Technology Stack

Software	Version
Java	Java JDK or JRE version 11
OBDX/OBAPI	22.2.5.0.0
OAuth	OBDX Internal OAuth

Abbreviations

OOTB	Out of the Box
TPP	Third Party Providers
ASPSP	Account Servicing Payment Service Provider

4. Pre-requisites

- Java JDK or JRE version 11 or higher must be installed. For installation of Java please refer installation guide.
- OAuth Setup

5. Headers Configuration

There are two types of headers configuration available for UK Open Banking.

- System Headers (i.e. Mandatory Headers and its respective value validation)
- Configuration Headers (i.e. Mandatory Headers).

Below are the configuration steps and Out of the box header already configured in the system.

System Headers:- As of now in OOTB one header has been added as mandatory “x-fapi-financial-id” with value as “491308330388688” (This is a random value and can be changed. This value is issued by OBIE and corresponds to the Organization Id of the ASPSP in the Open Banking Directory). This value needs to be configured by Bank or ASPSP. This header needs to be sent by the TPP to the ASPSP mandatorily with the same value. Both Header name and Header value are validated for System Headers.

For configuring more system headers, below script is to be executed in the OBAPI Admin schema.

Insert into DIGX_FW_CONFIG_ALL_B (PROP_ID, CATEGORY_ID, PROP_VALUE, FACTORY_SHIPPED_FLAG, PROP_COMMENTS, SUMMARY_TEXT, CREATED_BY, CREATION_DATE, LAST_UPDATED_BY, LAST_UPDATED_DATE, OBJECT_STATUS, OBJECT_VERSION_NUMBER) values ('uk%%**HEADER NAME**%%','OpenbankingSystemHeaders','%%**HEADERVALUE**%%','N',null,'Open Banking','ofssuser',sysdate,'ofssuser',sysdate,'Y',1);

Below Query is used to check the System Headers in the system

select * from digx_fw_config_all_b where category_id = 'OpenbankingSystemHeaders';

Configuration Headers :- As of now in OOTB one header has been added as mandatory - “x-fapi-interaction-id”. This header is required to be sent by the TPP to the ASPSP mandatorily with any value.

Only header name is validated in case of Configuration Headers.

For configuring more config headers, below script is to be executed in the OBDX/OBAPI Admin schema.

Insert into DIGX_FW_CONFIG_ALL_B (PROP_ID, CATEGORY_ID, PROP_VALUE, FACTORY_SHIPPED_FLAG, PROP_COMMENTS, SUMMARY_TEXT, CREATED_BY, CREATION_DATE, LAST_UPDATED_BY, LAST_UPDATED_DATE, OBJECT_STATUS, OBJECT_VERSION_NUMBER) values ('uk%%**HEADER NAME**%%',' OpenbankingConfigHeaders',null,'N',null,'Open Banking','ofssuser',sysdate,'ofssuser',sysdate,'Y',1);

Below Query is used to check the System Headers in the system

```
select * from digx_fw_config_all_b where category_id = 'OpenbankingConfigHeaders';
```

6. Properties

Below are the properties required to be updated in the UK Open Banking. Please find the below properties, its purpose and OOTB values.

Table:- DIGX_FW_CONFIG_ALL_B

Category-Id :- OpenBankingConfig

Property Id	Property Value (Out of the Box)	Purpose
CONSENT_EXPIRY DAYS	90	This value is used to check if expiry date send by TPP for the Account Access Consent is not more than 90 days and if it is more than 90 days then ASPSP will reject this value

Token Settings

Table:- DIGX_FW_CONFIG_ALL_B

Category-Id :- OAuthTokenConfig

Note: Prior to changing the value of OAUTH_TOKEN_SIGNER to X509RS256 or X509PS256, make sure to generate Public and Private Key Pair in Security Keys Section by logging in as admin.

Property Id	Property Value	Purpose
OAUTH_TOKEN_SIGNER	X509RS256 – x509 signed token with RS256 algorithm X509PS256 - x509 signed token with PS256 algorithm	The algorithm used to generate JWT token.
refreshTokenExpiry	86400	Default expiry for Refresh Token.
tokenExpiry	3600	Default expiry for Access Token.
ISSUER	OBDX-OAUTH	Issuer of the access/refresh token.
AUDIENCE	OBDXTestResServer	Audience of the access/refresh token.
OPAQUE_ACCESS_TOKEN_FLAG	Values can be Y or N.	Flag to enable/disable opaque access token.
CODE_CHALLENGE_FLAG	Values can be Y or N.	Flag to enable/disable code challenge verification as per the FAPI requirement.

Common Settings

Table:- DIGX_FW_CONFIG_ALL_B

Category-Id :- OAuthCommonConfig

Property Id	Property Value	Purpose
OAUTH_REDIRECT_HOST_PORT	http://{{HOST}}:{{PORT}}	'HOST' refers to the hostname/IP of the application 'PORT' refers to the application's port

DCR(Dynamic Client Registration) Configs

Table:- DIGX_FW_CONFIG_ALL_B

Category-Id :- OAuthDCRConfig

Property Id	Property Value	Purpose
PUBLIC_KEY_FETCH_URL	e.g. https://keystore.openbankingtest.org.uk/keystore/openbanking.jwks	Open Banking Directory URL to fetch the public key for payload jwt verification.
SCIM_PRIVATE_KEY	This should be value of obseal_dec.key without any space or enter character	This is ASPSP's private key for signing the jwt payload while communicating to the Open Banking Directory.
SSA_REQUEST_PAYLOAD_PUBLIC_KEY	e.g. software_jwks_endpoint	This is TPP's SSA claimset value, which is used to fetch the public key to verify SSA.
DCR_REQUEST_PAYLOAD_PUBLIC_KEY	e.g. software_jwks_endpoint	This is TPP's claimset value, which is used to fetch the public key to verify DCR payload.
ID_TOKEN_PRIVATE_KEY	This should be value of obseal_dec.key without any space or enter character	This is ASPSP's private key for signing the jwt payload.
ID_TOKEN_PRIVATE_KEYID	This should be value of key id generated when the ASPSP's certificate is uploaded in the Open Banking Directory.	This is ASPSP's key id to fetch the ASPSP's public key from the Open Banking Directory by the TPP.
OBIE_CLAIM	iss	To identify the issuer claimset in the DCR payload.
OBIE_CLAIM_VALUE	OpenBanking Ltd	To identify the value of the issuer claimset in the DCR payload.
OBIE_MEMBSTATE_VALUE	GB	Member state of the SSA.
OBIE_SOFTENV_VALUE	Values can be sandbox/production.	To identify software environment. Value should be 'production' for the production environment.

DCR(Dynamic Client Registration) SCIM Configs to Connect to Open Banking Directory

Table:- DIGX_FW_CONFIG_ALL_B

Category-Id :- OAuthDCRSCIMConfig

Property Id	Property Value	Purpose
MTLS_CERTIFICATE_ALIAS	Alias which was used to create the MLTS certificate e.g. openbanking_obtrans	Required for communication over MTSL with the Open Banking Directory.
MTLS_CERTIFICATE_PWD	Password which was used to create the MLTS certificate	Required for communication over MTSL with the Open Banking Directory.
IDENTITY_STORE_PATH	Path of the identity store jks file. e.g. /scratch/obdx/wls/OpenBanking/SCIM/openbanking_custom_identity.jks	Required for communication over MTSL with the Open Banking Directory.
TRUST_STORE_PATH	Path of the trust store jks file. e.g. /scratch/obdx/wls/OpenBanking/SCIM/openbanking_custom_trust.jks	Required for communication over MTSL with the Open Banking Directory.
PROXY_ENABLED	Values can be Y/N.	To identify whether the proxy is enabled of not for the communication.
PROXY_URL	Value ot the proxy url.	Required for communication over MTSL with the Open Banking Directory with proxy enabled.
PROXY_PORT	Value ot the proxy port.	Required for communication over MTSL with the Open Banking Directory with proxy enabled.
HTTPS_ENABLED	Values can be Y/N.	To identify whether the https is enabled of not for the communication.
softwareStatementId	This should the Software Statement Id of the ASPSP.	Required for communication over MTSL with the Open Banking Directory.
clientScopes	'TPPReadAll AuthoritiesReadAccess QTSPReadAccess'	These are the scopes defined by the Open Banking Directory.
keyId	This should be the ASPSP's key id to be used for the MSTL communication.	Required for communication over MTSL with the Open Banking Directory.
tokenUrl	https://matls-sso.openbankingtest.org.uk/as/token.oauth2	This is defined by the Open Banking Directory to get the access token for accessing the Open Banking APIs.

certUrl	https://matls-dirapi.openbankingtest.org.uk/certificate/validate	This is defined by the Open Banking Directory to get the ASPSP's certificate validated for the MTLS communication.
orgDetUrl	https://matls-api.openbankingtest.org.uk/scim/v2/OBThirdPartyProviders/	This is defined by the Open Banking Directory to get the organisation details.
aud	https://matls-sso.openbankingtest.org.uk/as/token.oauth2	This is defined by the Open Banking Directory for the 'audience' claimset for communication over MTSL.
iss	This should the Software Statement Id of the ASPSP.	Required for communication over MTSL with the Open Banking Directory.
sub	This should the Software Statement Id of the ASPSP.	Required for communication over MTSL with the Open Banking Directory.
grantType	client_credentials	This is defined by the Open Banking Directory for communication over MTSL.
clientAssertionType	urn:ietf:params:oauth:client-assertion-type:jwt-bearer	This is defined by the Open Banking Directory for communication over MTSL.

Sort Code and Branch Mapping for UK.OBIE.SortCodeAccountNumber Scheme

For Sort Code, Account branch mapping following entry needs to be done in DIGX_FW_CONFIG_ALL_B in openBankingConfig preferences. This mapping used in account identification deserializer to replace sort code with appropriate branch code.

Insert into DIGX_FW_CONFIG_ALL_B (PROP_ID,CATEGORY_ID,PROP_VALUE,FACTORY_SHIPPED_FLAG,PROP_COMMENTS,SUMMARY_TEXT,CREATED_BY,CREATION_DATE,LAST_UPDATED_BY,LAST_UPDATED_DATE,OBJECT_STATUS,OBJECT_VERSION_NUMBER,EDITABLE,CATEGORY_DESCRIPTION) values ('SORT_CODE_<6 Digit SortCode>','openBankingConfig','<Branch Code>','N',null,'Sort Code Branch Mapping for UK Openbanking for Sort Code Scheme','ofssuser',sysdate,'ofssuser',sysdate,'A',1,'N',null);

7. OAuth Configuration

7.1 UI configuration

OAuth Identity Domain Maintenance will require below maintenance to configure UI Component for Authorizing consent.

The value of Consent Page URL (Menu -> OAuth -> Identity Domain Maintenance) is configured as http://host:port?existingDashboard=true&homeComponent=authorize-consent&homeModule=open-banking&applicationType=auth.

8. Extensibility and Code Conventions

Code Convention of Account API's

Accounts related API should use below arguments and return type for working with UK Open Banking

Arguments

SessionContext sessionContext

com.ofss.digx.app.openbanking.dto.accounts.uk.AccountRequestDTO accountRequestDTO

Return Type

BaseResponseDTO<T>

Where T extends DataTransferObject

Any service implemented with the above type of argument will be compatible with UK Open Banking.

Code Convention of Payment API's

Payment related API should use below arguments and return type for working with UK Open Banking

Arguments

Create and Read Method

SessionContext sessionContext

Any DTO Object which extends com.ofss.digx.app.openbanking.dto.consent.uk.UKPaymentDTO

Any service implemented with the above type of argument will be compatible with UK Open Banking.

Error Message Framework

The Error Message Framework helps convert the OBAPI error response according to the UK Open Banking Specifications.

The error response structure for Open Banking Read/Write APIs is as follows:

```
{
  "Code": "...",
  "Id": "...",
  "Message": "...",
  "Errors": [
    {
      "ErrorCode": "...",
      "Message": "...",
      "Path": "...",
      "Url": "..."
    }
  ]
}
```

1.1

The UK Open Banking specified error response is handled using DIGX_OB_UK_OBDX_ERROR_MAP table.

The contents of the table are as follows:

Column Name	Description
DIGX_ERROR_CODE	Represents the OBAPI error codes. This is a Primary and Unique Key
UK_ERROR_CODE	Represents the Open Banking specified error code
PATH	Represents the reference to the JSON Path of the field with error. Can be null.
URL	Represents the URL to help remediate the problem, or provide more information etc. Can be null.

For mapping OBAPI error codes with UK Open Banking specified codes below script can be used:

Insert into DIGX_OB_UK_OBDX_ERROR_MAP (DIGX_ERROR_CODE,UK_ERROR_CODE,PATH,URL) values **('%%OBDX Error Code%%',%%Open Banking specified error code%%', '%%Path%%', '%%URL%%');**

For example –

Insert into DIGX_OB_UK_OBDX_ERROR_MAP (DIGX_ERROR_CODE,UK_ERROR_CODE,PATH,URL) values ('DIGX_OB_0010','UK.OBIE.Field.Missing', 'Data.Initiation ',null);

Below Query is used to check the OBAPI errors mapped with UK Open Banking specified error codes in the system

select * from DIGX_OB_UK_OBDX_ERROR_MAP;

For configuring HTTP status codes with custom message, below script can be used:

Insert into DIGX_FW_CONFIG_ALL_B (PROP_ID, CATEGORY_ID, PROP_VALUE, FACTORY_SHIPPED_FLAG, PROP_COMMENTS, SUMMARY_TEXT, CREATED_BY, CREATION_DATE, LAST_UPDATED_BY, LAST_UPDATED_DATE, OBJECT_STATUS, OBJECT_VERSION_NUMBER)

values ('%%**HTTP Status code**%%','OpenBankingErrorConfig','%%**Error Message**%%','N',null,'OpenBanking Error Message','ofssuser',sysdate,'ofssuser',sysdate,'Y',1);

Below Query is used to check the Open Banking HTTP status codes in the system

select * from digx_fw_config_all_b where category_id = ' OpenBankingErrorConfig';

Permission Response Handler

Permissions is used in only Account API’s. Based on Permissions, Response is generated based on permissions.

OBAPI consists of Permission Handler against each type of permissions. This configuration is availble in the table **DIGX_OB_UK_PERMISSIONS_PRIMARY**

The contents of the table are as follows:

Column Name	Description
SERVICEID	Represents the OBAPI Service Id for which the permission and its handler is available
PERMISSION	Represents Permission
RESPONSEHANDLER	Represent Permission Handler

Permission Handler can be overriden or can be newly introduced. This will be required for additional fields mapping which is not available OOTB. Steps for the same are as follows

Introducing Permission Handler

New Permisison Handler should implement interface IResponseHandler

New Permission Handler should have below methods

public static <T implements IResponseHandler> getInstance()

public <T extends DataTransferObject> assembleResponse(DataTransferObject object, List<String> permissions) – This method assembles response from object to the require response object which needs shown in the API response. Object is the response got from base sevice and T will be the response object require by API specifications. Assembling of the values will be done this method

public int getPriority() – This defines the high priority of the handler to be applied for assembling response in case of permissions and its handler has been consented by the user i.e. Basic and Detail permission will have different handlers but if the consent is both the permisison the priority of the handler will decide which needs to be executed on high priority.

8.1 Key Providers support

Key Providers Overview

Whenever TPP initiates a DCR request, the payload is signed with the TPP's private key and same needs to be verified with the TPP's public key at the Bank's side. There could be different ways to get the TPP's public key which can vary as per open banking directory services and the geographical regions.

To accommodate those varying approaches of getting the public key, OBDX has provided factory pattern to get a 'Key Provider'. The main job of the key provider is to get the public key of the TPP, to verify the DCR payload, based on the Software Statement Issuer Name.

To implement the above, one IKeyProvider interface is added. This contains the methods which may differ based on the parameters mentioned above.

```java

import java.security.interfaces.RSAPublicKey;

public interface IKeyProvider {

    public String getPublicKey(String clientId, String kid);

    public Map<String, String> fetchPublicKey(String dcr_request_token);

    public Map<String, String> getPublicKeyClaims(String x509Certificate, String keyId);

    /**
     * Derives the RSA public key from the Base64 public key/certificate
     *
     * @param encodedKeyOrCert
     * @return
     */
    public RSAPublicKey getRSAPublicKey(String encodedKeyOrCert);
}
```

There are 4 methods to be implemented.

1. `public Map<String, String> fetchPublicKey(String dcr_request_token);` - to fetch the TPP's public key when the TPP is being onboarded with the bank with the help of DRC Request Token data.
2. `public String getPublicKey(String clientId, String kid);` - to fetch the TPP's public key based on the client id and the key id for further requests processing as and when required when the TPP is already onboarded with the bank.
3. `public Map<String, String> getPublicKeyClaims(String x509Certificate, String keyId);` - to get the various types of claims like certificate type, validity, expiry, revocation etc.
4. `public RSAPublicKey getRSAPublicKey(String encodedKeyOrCert);` - to get the decrypted RSA public key from the encoded key or extracted from the certificate.

In addition to above methods, to make the key provider class singleton, provider class must implement to return the singleton instance of the class

```java
public static IKeyProvider getInstance();
```

Key Provider Implementation & Configuration

To create a key provider, one needs to create a KeyProvider class by extending the com.ofss.digx.oauth2.spi.IKeyProvider interface and making the provider class entry in the DIGX_FW_CONFIG_ALL_B table.

For example, we have a SSA Issuer called 'XYZ Ltd'.
We will need to follow below two steps to configure the XYZ key provider

1. Need to create a new key provider implementation class - com.ofss.digx.openid.service.XYZKeyProvider which must implement the IKeyProvider interface. Name and the package of the key provider class could be anything, those are not compelled to be same as the mentioned above, but it must implement the IKeyProvider interface.
2. Need to make the provider class entry in the DIGX_FW_CONFIG_ALL_B with prop_id = 'XYZ Ltd_KEY_PROVIDER'. In this entry, the naming convention should strictly be followed as <SSA_Issuer>*_KEY_PROVIDE* and the CATEGORY_ID must be 'openBankingConfig'.

To configure new key provider in DB, refer below insert query and its values are described as below:

Insert into DIGX_FW_CONFIG_ALL_B (PROP_ID,CATEGORY_ID,PROP_VALUE,FACTORY_SHIPPED_FLAG,PROP_COMMENTS,SUMMARY_TEXT,CREATED_BY,CREATION_DATE,LAST_UPDATED_BY,LAST_UPDATED_DATE,OBJECT_STATUS,OBJECT_VERSION_NUMBER,EDITABLE,CATEGORY_DESCRIPTION) values ('XYZ Ltd_KEY_PROVIDER','openBankingConfig','com.ofss.digx.openid.service.XYZKeyProvider','N',null,'XYZ Ltd Key Provider Class','ofssuser',sysdate,'ofssuser',sysdate,'A',1,'N',null);

As per the current standards, there are mainly two open banking authorities in European Continent:

1. Open Banking Directory (OBD)
2. European Banking Authority (EBA)

A Third-Party Provider (TPP) gets registered with any of the above two authorities and obtains the Software Statement (SSA) before getting onboarded with the bank.

In this release, OBDX has provided the out of the box implementation of key providers for both directory services.

1. com.ofss.digx.openid.service.OBDKeyProvider – for Open Banking Directory
2. com.ofss.digx.openid.service.EBAKeyProvider – for European Banking Authority

To get the public key, OBD has provided ‘software_jwks_endpoint’. This endpoint provides a JSON Web Key Set (JWKS), which is a set of keys containing the public keys used to verify any JSON Web Token (JWT). Based on the key id, TTP’s public key is extracted from the JWKS to verify the payload.

Both the key providers currently communicate with the Open Banking Directory to fetch the TTP’s public key currently as per the implementation.

We have below two configurations:

1. OpenBanking Ltd_KEY_PROVIDER – to fetch the public keys of TPP’s whose SSA Issuer is the ‘OpenBanking Ltd’.
2. DEFAULT_KEY_PROVIDER - to fetch the public keys of TPP’s whose SSA Issuer is NOT the ‘OpenBanking Ltd’.

Besides above two configured providers, we have a mock key provider (for which, no configuration is needed in the DB):

3. MOCK_KEY_PROVIDER - "com.ofss.digx.oauth2.service.DBBasedKeyProvider" – this is only a dummy DB based key provider. If none of the above two providers are configured in the DB, KeyProviderFactory would return the mock key provider. It stores only single public-private key pair in the DB itself and uses the same pair for all the TPP payload verifications.

Below is a sample code snippet to get the key provider for reference:

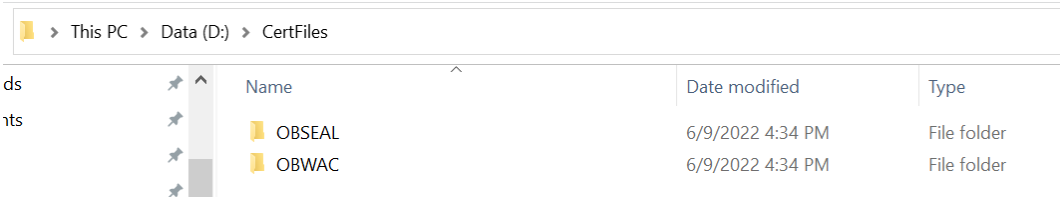
```
IKeyProvider keyProvider = KeyProviderFactory.getInstance().getProvider(issuer);
Map<String, String> publicKeysMap = keyProvider.fetchPublicKey(dcr_request_token);
```

9. Keystore and Certificate for UK Open Banking Directory

This section describes the steps to generate the 'jks' files and configure the same in OBDX for Open Banking Directory integration.

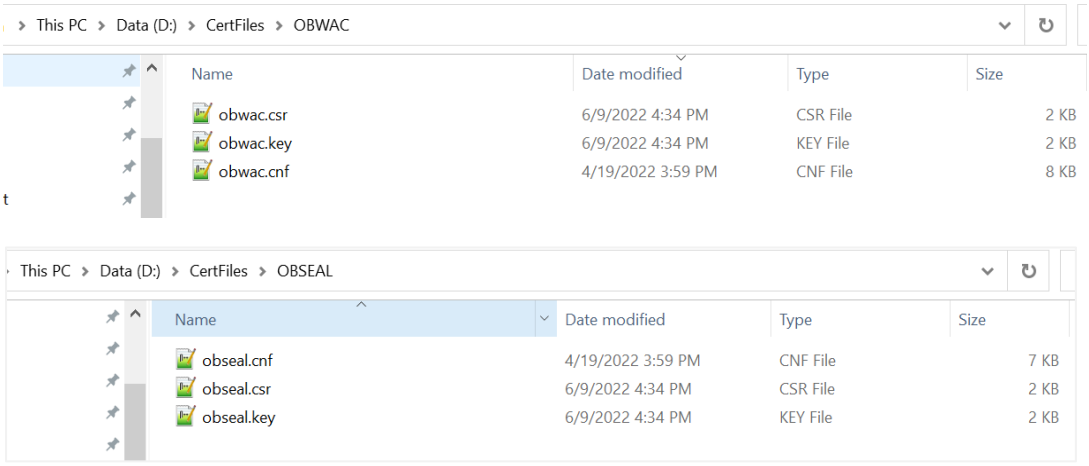
Steps to create 'identity' & 'trust' JKS files

1. Create two different folders OBWAC and OBSEAL and perform the below steps in the respective folders.



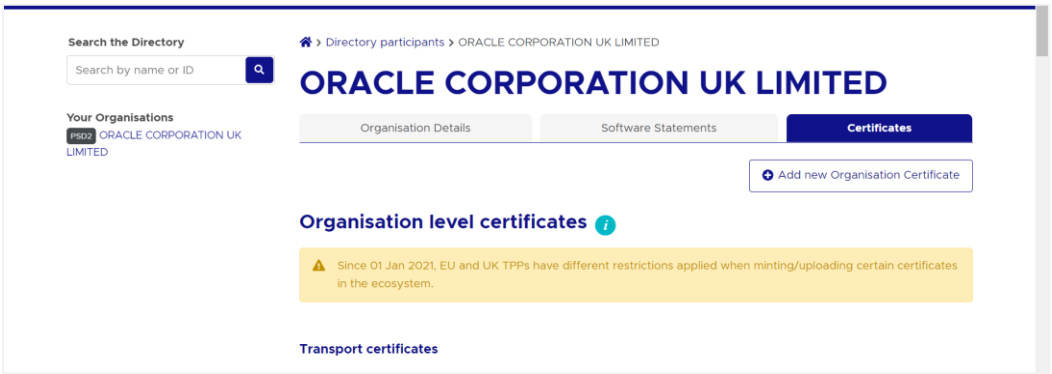
2. One should have the bank's OBWAC and OBSEAL configuration files(.cnf) to proceed further. Copy the files in the respective folders created above.
3. To generate CSR and key files for OBWAC and OBSEAL certificate with the help of .cnf file, execute below openssl commands
   OBWAC> **openssl req -new -config obwac.cnf -out obwac.csr -keyout obwac.key**
   OSEAL> **openssl req -new -config obseal.cnf -out obseal.csr -keyout obseal.key**
   Note: enter the same pass phrase(pass1234 for example) for both obwac and obseal and make a note of it.

.csr and .key files have been generated with the above commands

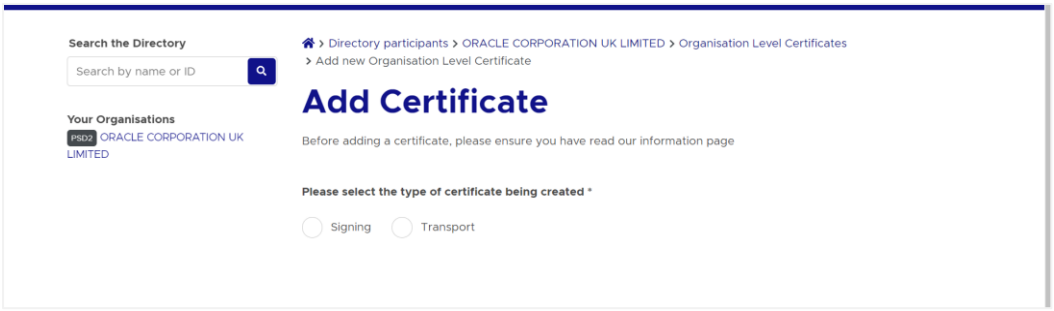


4. Upload the above generated .csr files in Open Banking Directory Account to get OBWAC and OBSEAL pem files.
Let's assume, below output on uploading .csr files in the OB directory account
Your OB WAC certificate xT-9_jWfAME1feTKZGaf8Dd_x1s was successfully created
Your OB Seal certificate I6cfLYUSt91fOw13kdO0HYdlVTc was successfully created
Below are the steps to generate the OB WAC and OB Seal certificates in the Open Banking Directory Account(***Note: below screenshots are from the Sandbox account, kindly use Production Open Banking Directory Account details for the production setup***):
    a. Login with Open Banking Directory account credentials and select the desired Directory Participant(Your Organization).

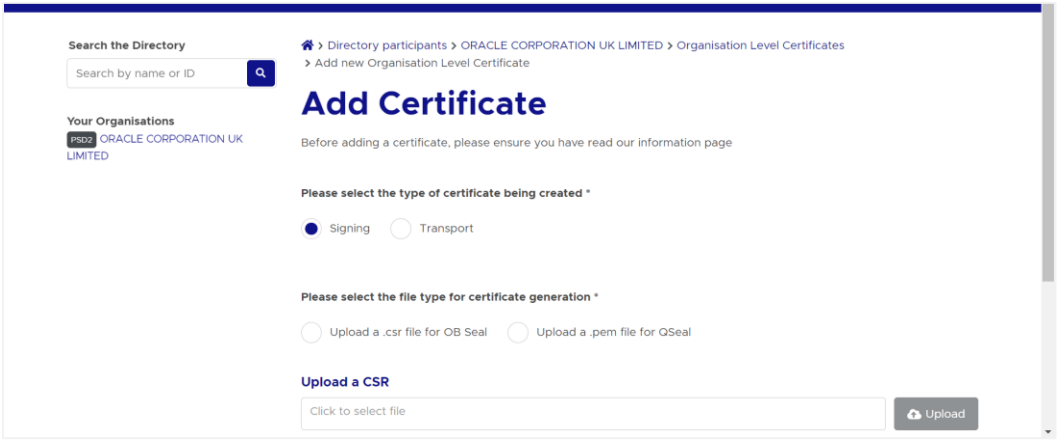
    b. Go to 'Certificates' tab



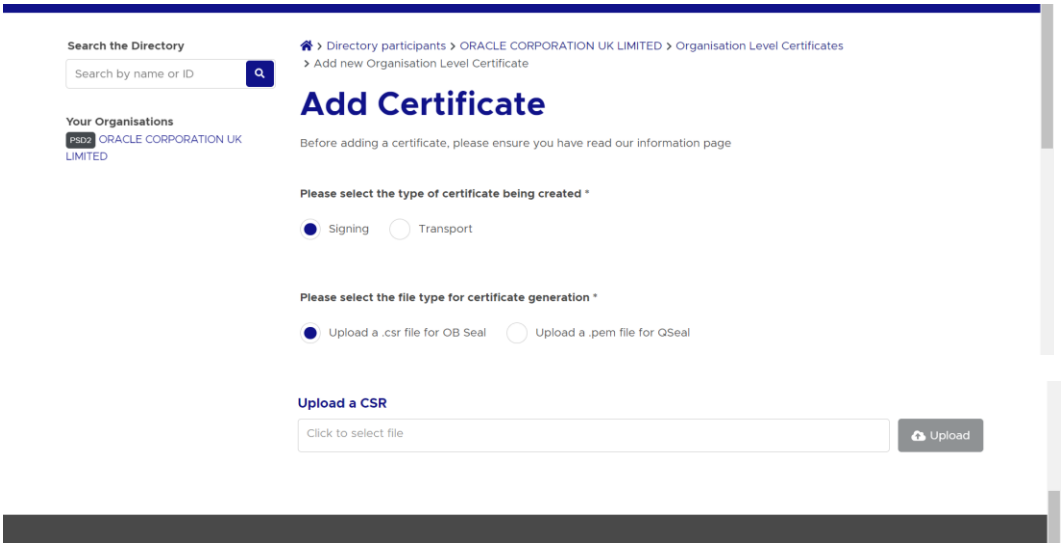
c. Click on ‘Add new Organisation Certificate’ button

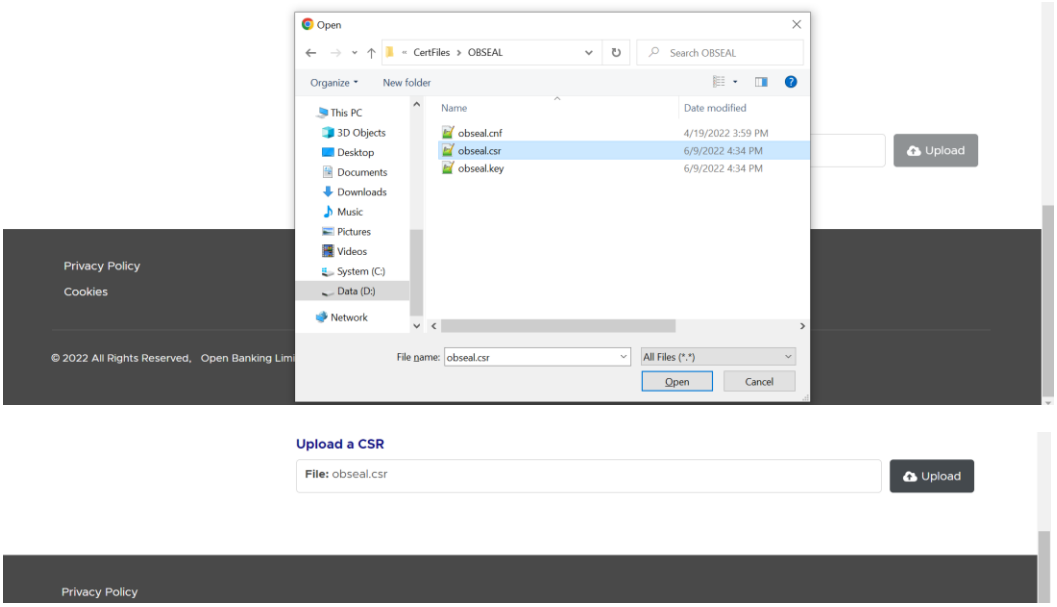


d. Select ‘Signing’ radio button to upload OB Seal .crs file

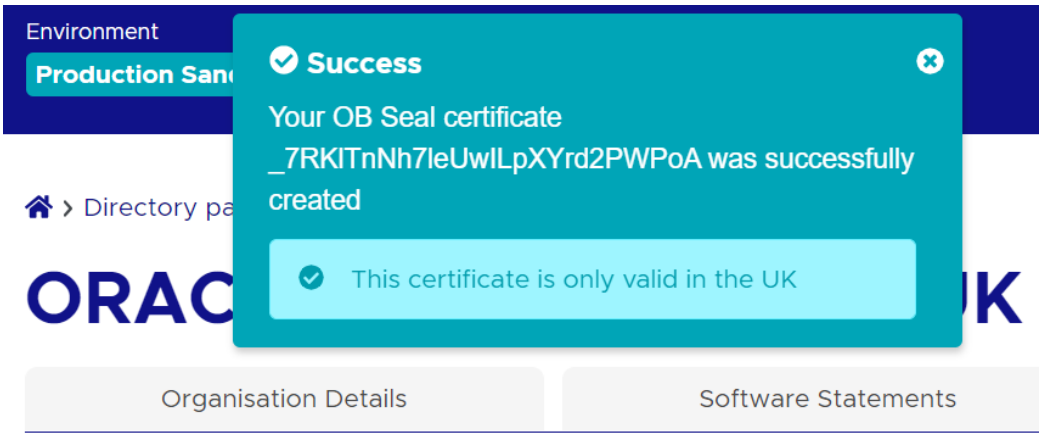


e. Select and upload the OB Seal .csr file

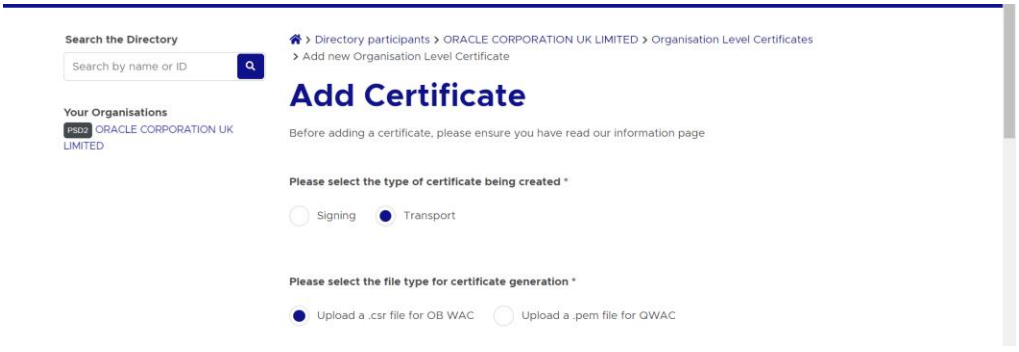


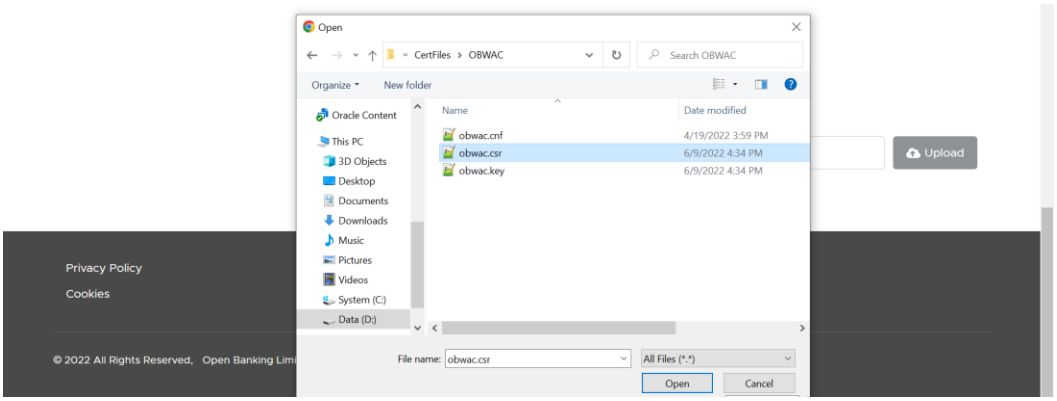


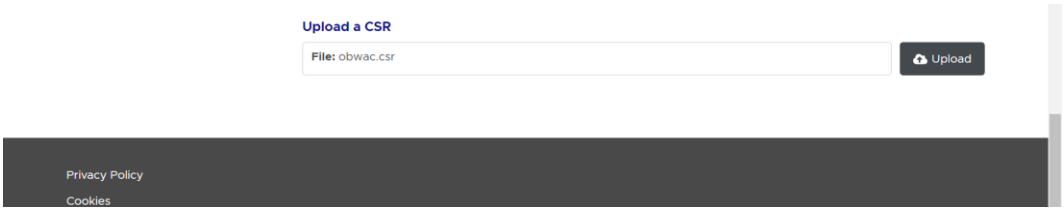
f. Clicking on the ‘Upload’ button will upload and display success popup

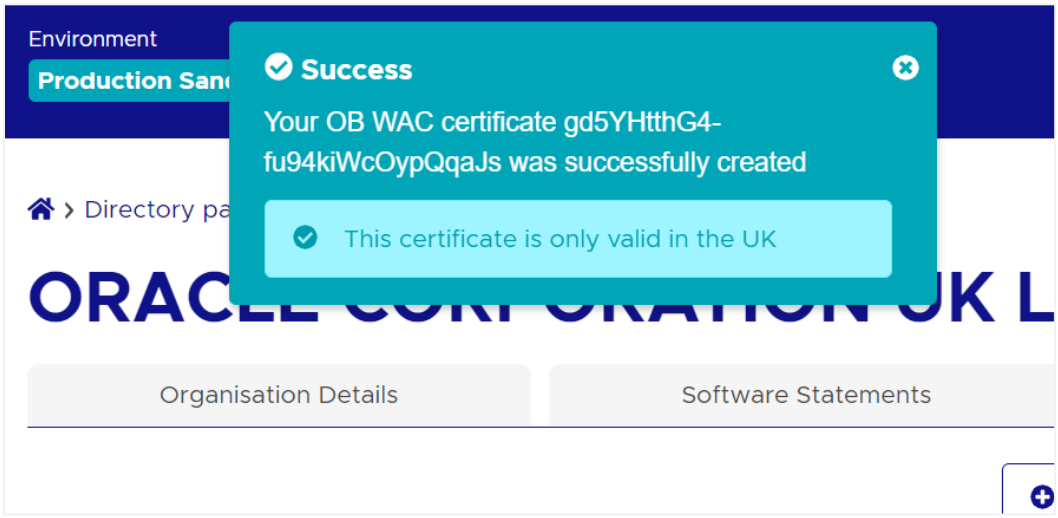


g. Repeat the above steps for OB WAC certificate generation. Select the ‘Transport’ radio button for OB WAC.

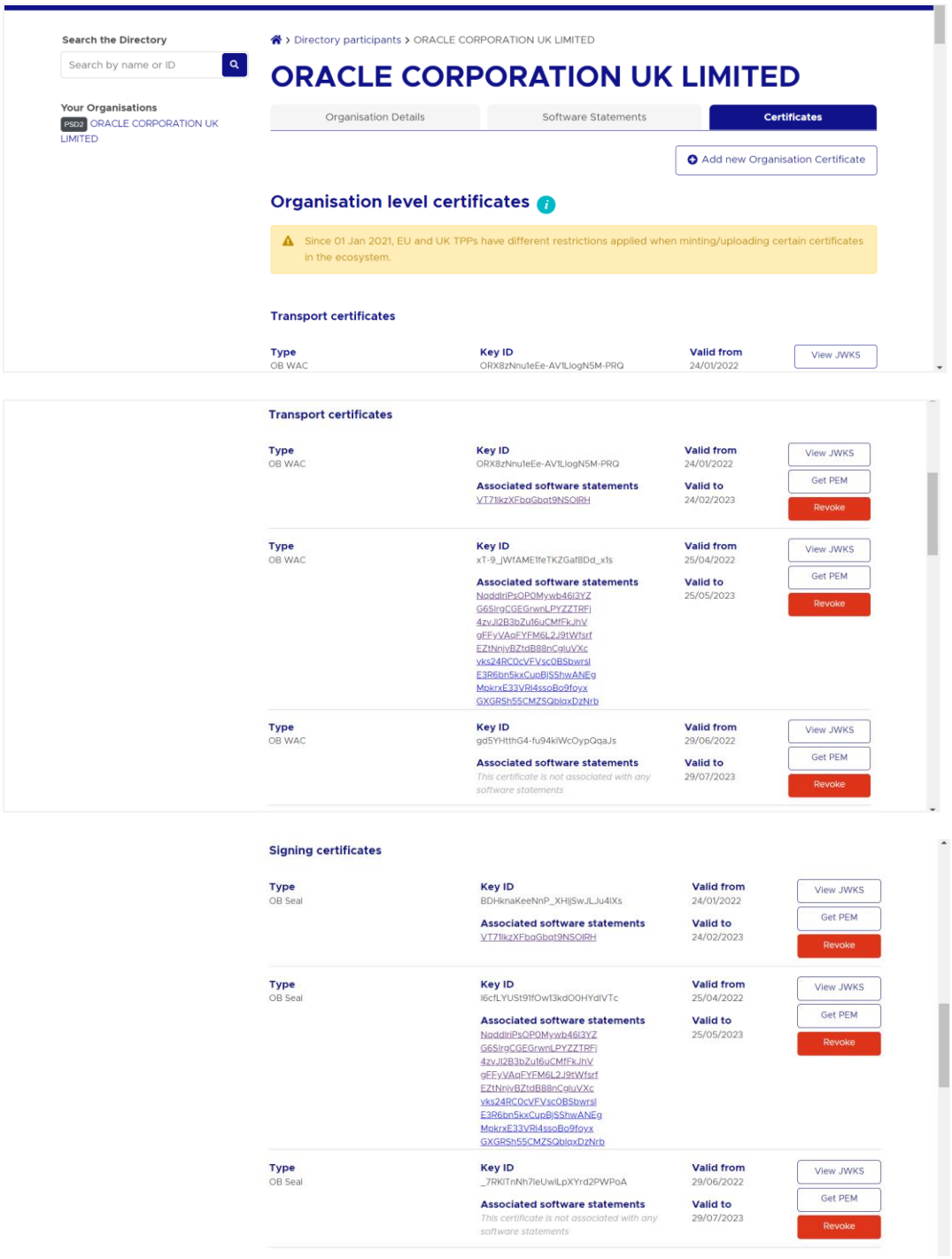




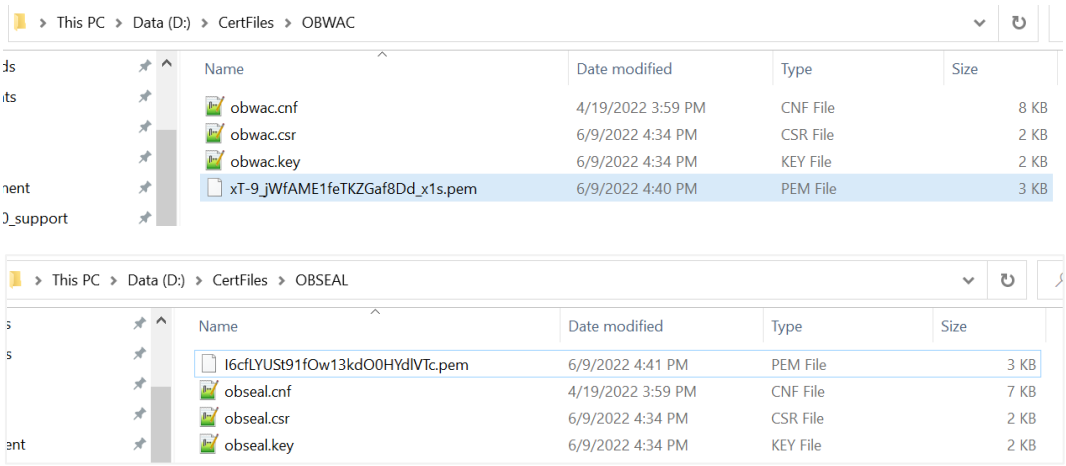




h. Generated certificates would be visible on the certificates listing page. Certificate .pem files can be downloaded with the help of ‘Get PEM’ button displayed next to the respective certificates

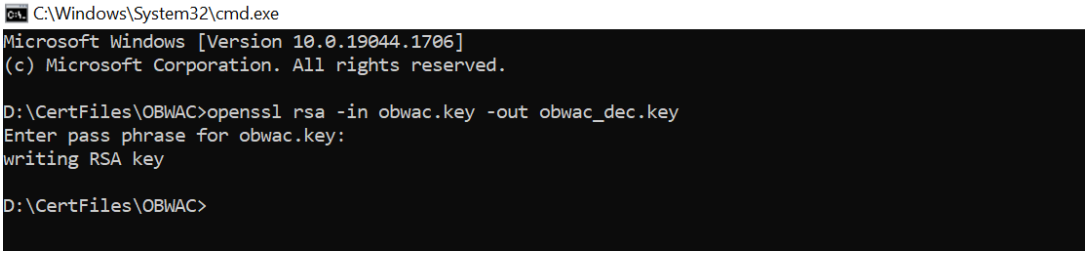


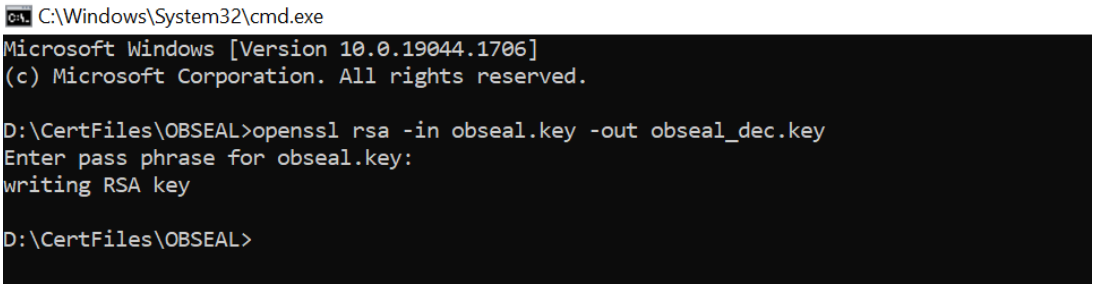
5. Download the generated OBWAC and OBSEAL files and copy in the respective folders which have created locally. Change the extension from '.cer' to '.pem' of the downloaded files if required.

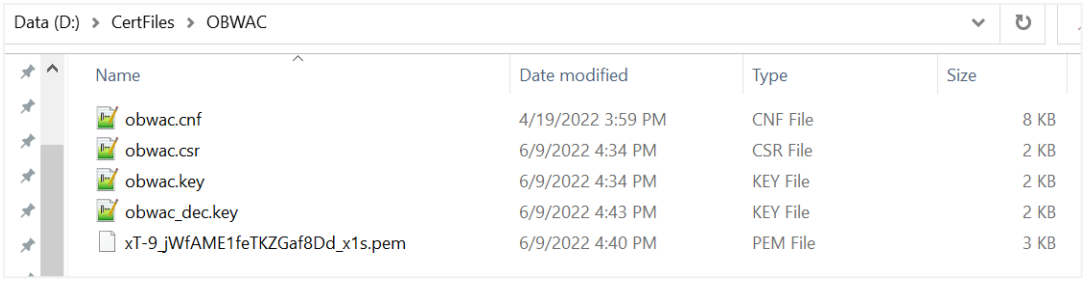


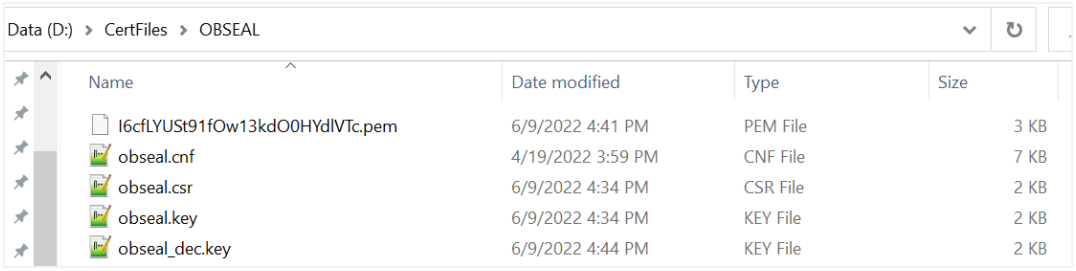
6. Generate decrypted keys by executing below commands
OBWAC> **openssl rsa -in obwac.key -out obwac_dec.key**
OBSEAL> **openssl rsa -in obseal.key -out obseal_dec.key**

Enter the pass phrase ‘pass1234’ when provided, which had been entered at the time of the .key files.



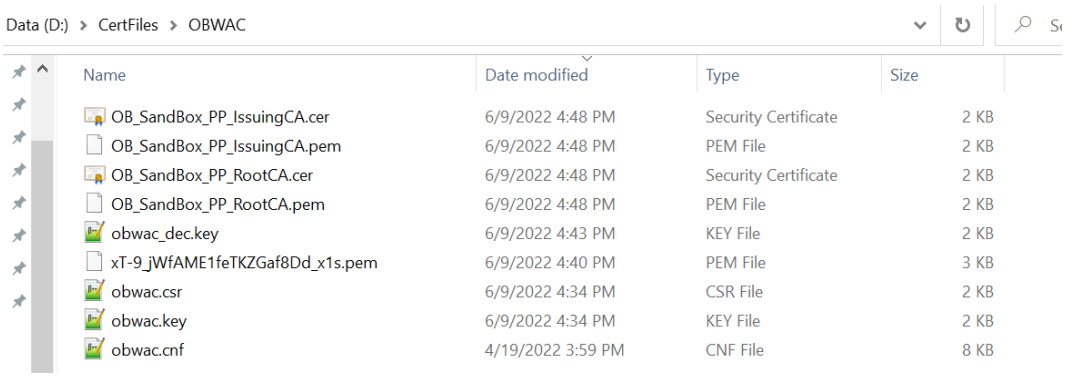




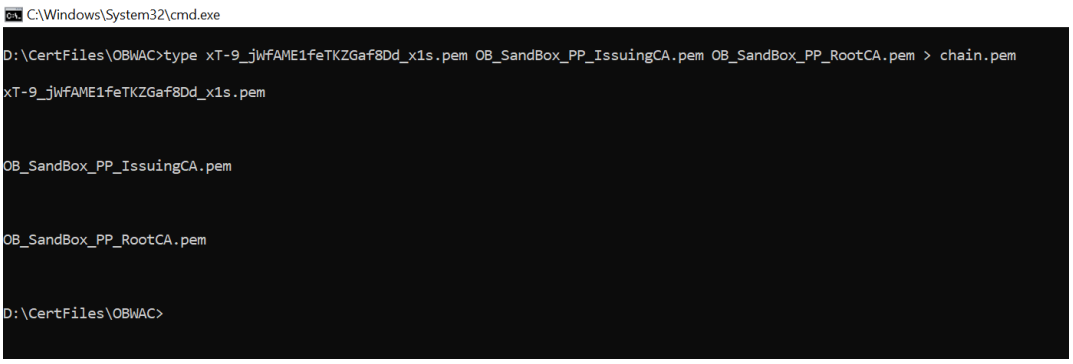


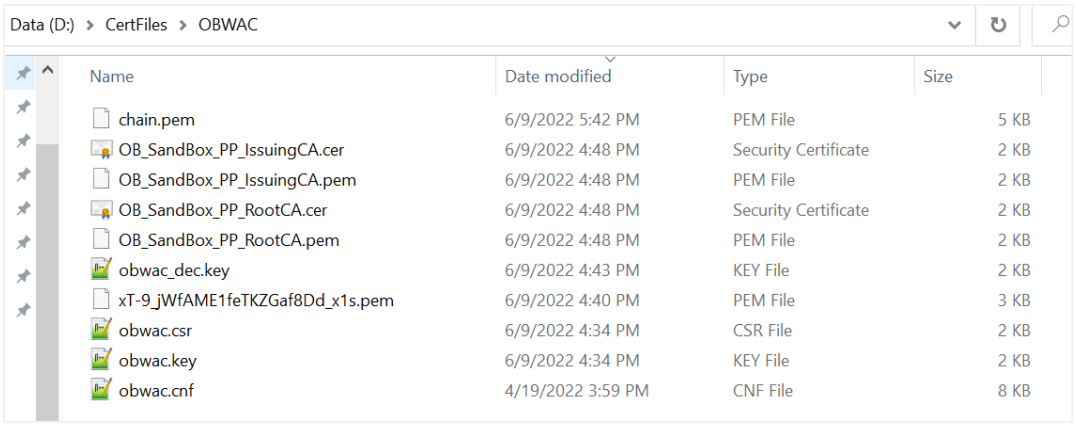
7. Download OB Root and Issuing Certificates from the Open Banking directory
    a. URL for sandbox certificates: https://openbanking.atlassian.net/wiki/spaces/DZ/pages/252018873/OB+Root+and+Issuing+Certificates+for+Sandbox
    b. URL for production certificates: https://openbanking.atlassian.net/wiki/spaces/DZ/pages/80544075/OB+Root+and+Issuing+Certificates+for+Production
8. Create a copy of both the downloaded certificate files and change the extension from .cer to .pem and copy in the OBWAC folder. Keep the file names same
    a. OB_SandBox_PP_IssuingCA.cer to OB_SandBox_PP_IssuingCA.pem
    b. OB_SandBox_PP_RootCA.cer to OB_SandBox_PP_RootCA.pem

    Note: remove the spaces from the pem file names if there are any.



9. Use ‘cat’ command on linux or ‘type’ command in Windows machine to build the certificate chain from the above three .pem files
    a. cat xT-9_jWfAME1feTKZGaf8Dd_x1s.pem OB_SandBox_PP_IssuingCA.pem OB_SandBox_PP_RootCA.pem > chain.pem
    or
    b. type xT-9_jWfAME1feTKZGaf8Dd_x1s.pem OB_SandBox_PP_IssuingCA.pem OB_SandBox_PP_RootCA.pem > chain.pem

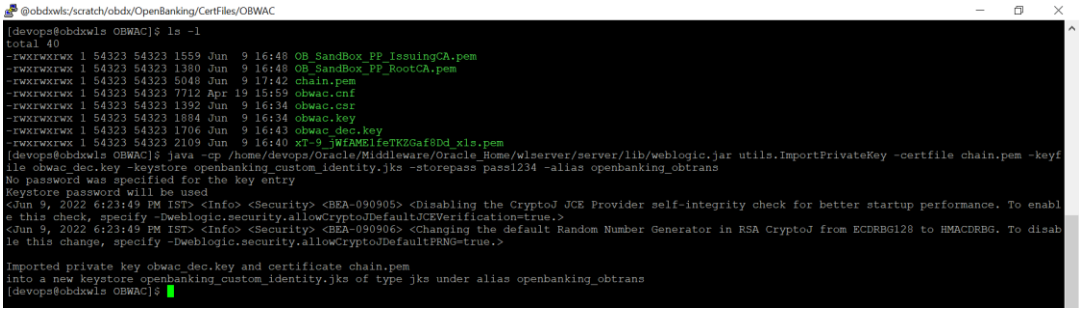




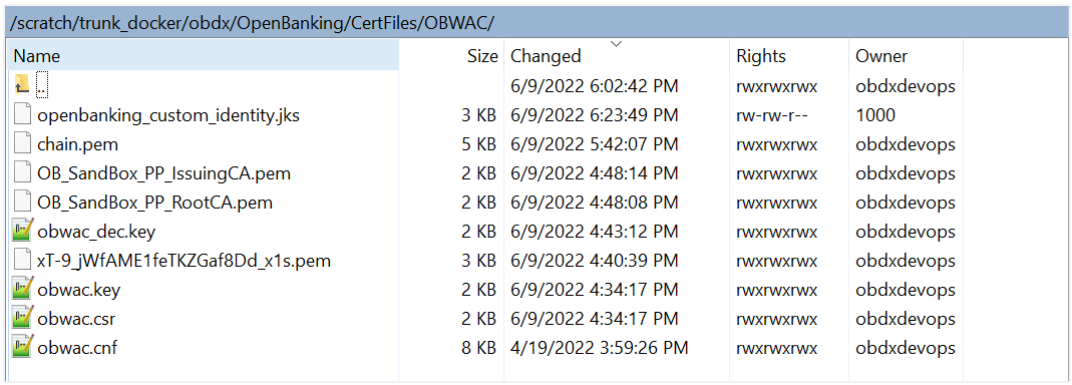
10. **Creating Custom Keystore and importing chain**

    WebLogic Server Java Utilities is used to create the custom keystore and importing private key & the certificates chains.
    Resource URL for reference:
    https://docs.oracle.com/cd/E13222_01/wls/docs81/admin_ref/utils20.html
    Execute the below command with files in the OBWAC directory
    java -cp /home/devops/Oracle/Middleware/Oracle_Home/wlserver/server/lib/weblogic.jar utils.ImportPrivateKey -certfile chain.pem -keyfile obwac_dec.key -keystore openbanking_custom_identity.jks -storepass pass1234 -alias openbanking_obtrans

    Note: "/home/devops/Oracle/Middleware/Oracle_Home/wlserver/server/lib/" this path is to locate the weblogic.jar file, this may differ as per the setup.



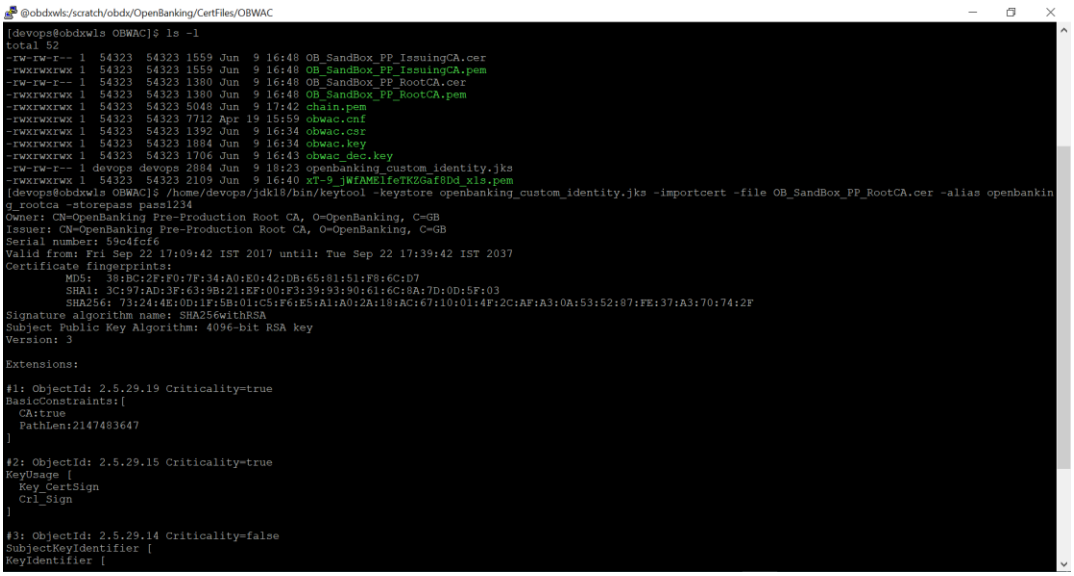
A new .jks file with the filename 'openbanking_custom_identity.jks' is created.

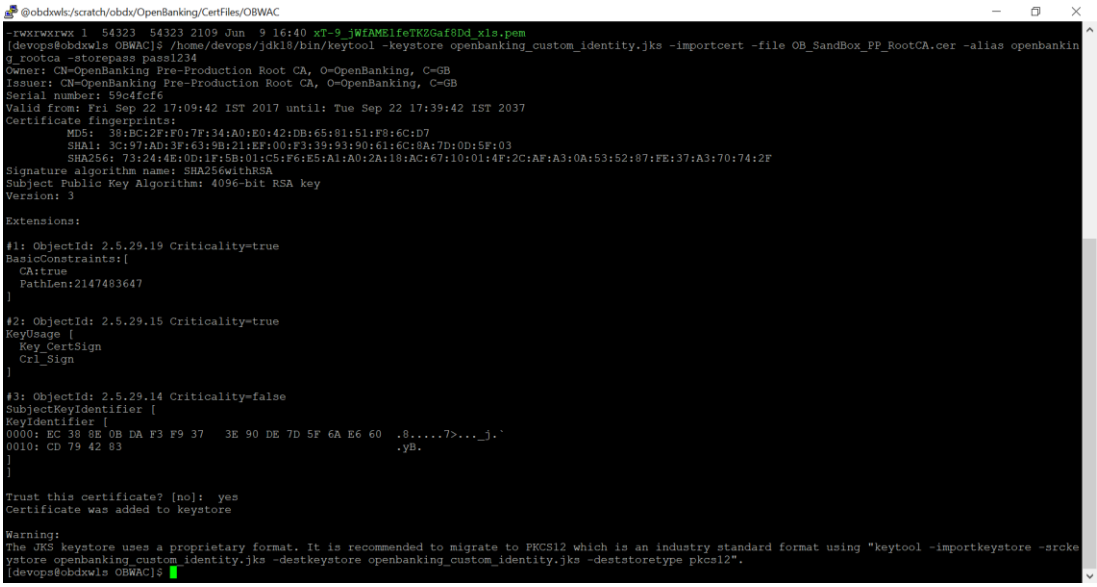


11. Creating Custom Identity Trust

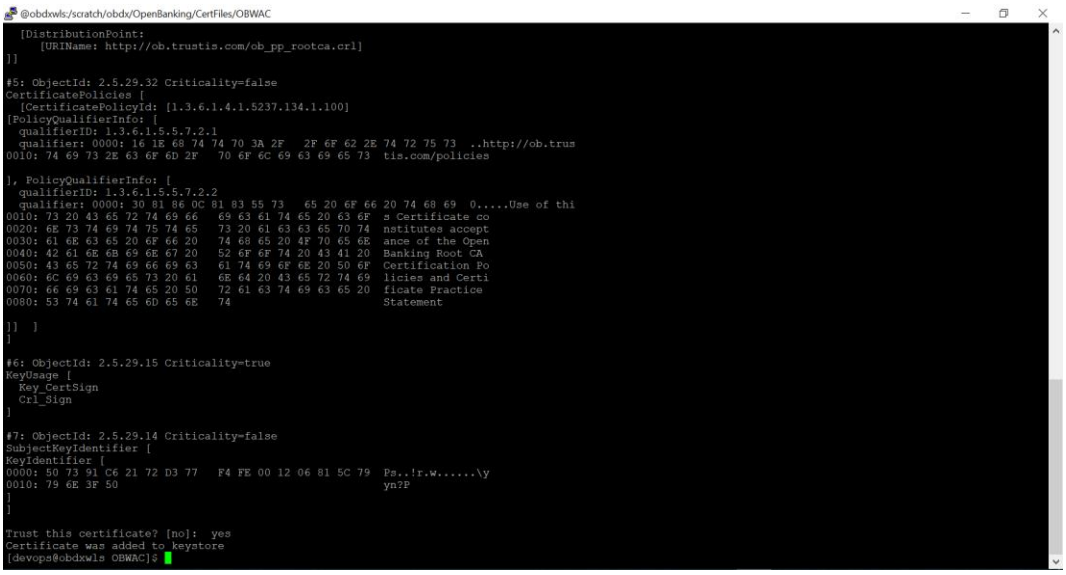
Execute below two commands.
Enter ‘yes’ and press enter when prompted “Trust this certificate? [no]:”.
Note: “/home/devops/jdk18/bin/” this path is to locate the java keytool, this may differ as per the setup.

1) **/home/devops/jdk18/bin/keytool -keystore openbanking_custom_identity.jks -importcert -file OB_SandBox_PP_RootCA.cer -alias openbanking_rootca -storepass pass1234**

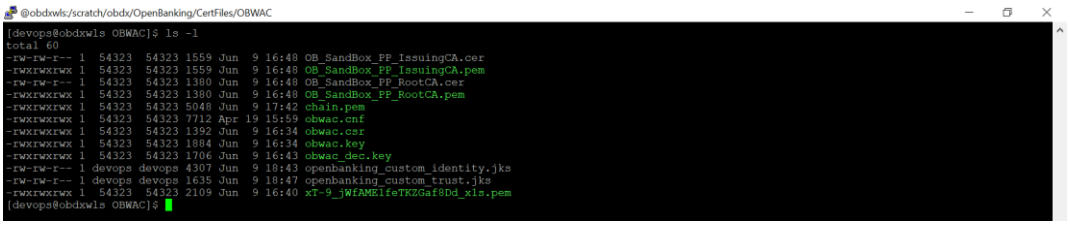


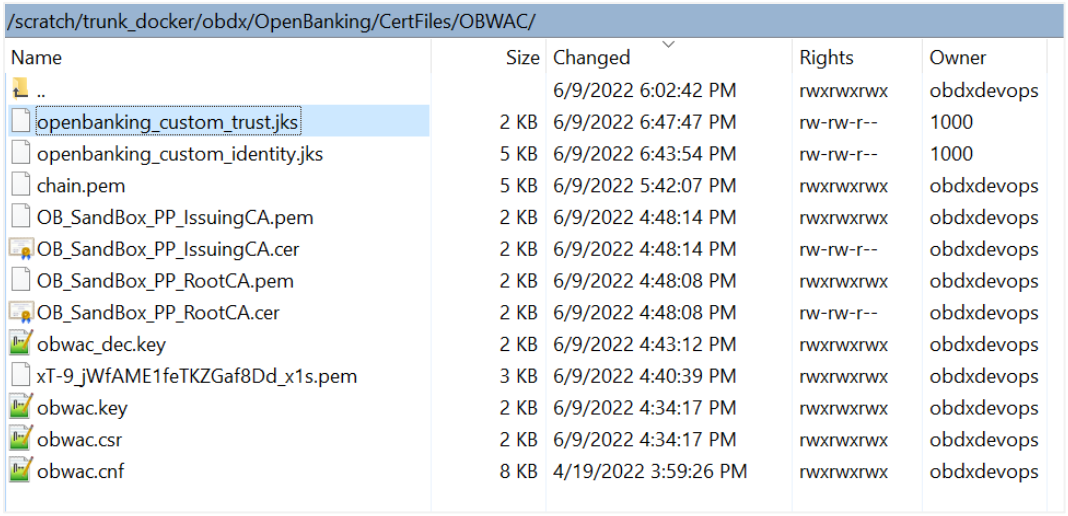


2) **/home/devops/jdk18/bin/keytool -keystore openbanking_custom_trust.jks -importcert -file OB_SandBox_PP_IssuingCA.cer -alias openbanking_issueca -storepass pass1234**

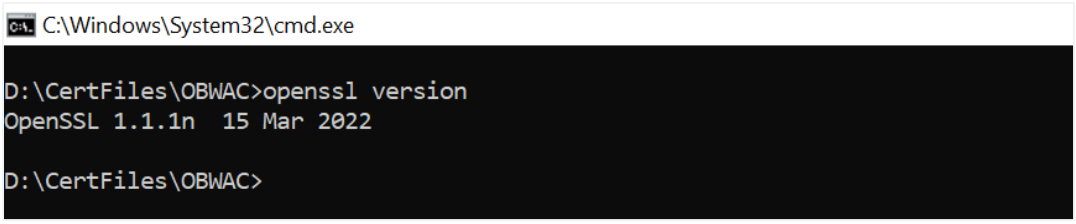


Another .jks file with filename ‘openbanking_custom_trust.jks’ is created.





Note: OpenSSL 1.1.1n 15 Mar 2022 is used to perform above steps.


C:\Windows\System32\cmd.exe
D:\CertFiles\OBWAC>openssl version
OpenSSL 1.1.1n 15 Mar 2022
D:\CertFiles\OBWAC>