LLOYDS BANKING GROUP (LBG)
OPEN BANKING SANDBOX
DEVELOPERS GUIDE V1.4

Table of Contents

1 INTRODUCTION

1.1 OVERVIEW

The Lloyds Banking Group (LBG) Sandbox is designed to replicate the functionality present in the LBG production environment. It is provided for the Third Party Providers (TPP) to be able to develop and test their APIs in a safe, controlled environment, without impacting the live environments.

This Sandbox enables TPPs to complete end-to-end testing of their products and allows them to use the same codebase when they move to the production environment.

The Sandbox, where possible and where appropriate, mirrors all the functionality of the production interface relating to Account Information Service Provider (AISP), Payment Initiation Service Provider (PISP) and Card Based Payment Instrument Issuer (CBPII) use cases. There are however some key differences and this guide seeks to capture them.

This guide should be read in conjunction with the Technical Implementation Guide ("Guide to using Lloyds Banking Group APIs") which is issued as part of the production environment.

1.2 KEY DIFFERENCES BETWEEN THE SANDBOX AND THE PRODUCTION INTERFACE

The functionality works like the production interface for all the happy paths and error scenarios. The exceptions to this are:

1.2.1 BRANDING

- Functionally, the Sandbox is agnostic to the different brands of the Lloyds Banking Group (LBG) like Lloyds, Halifax, Bank of Scotland and MBNA.
- The Sandbox UI is themed on a single common LBG brand unlike the production interface that are themed for the individual brands.
- The Sandbox provides a set of eight inbuilt PSUs as a representative sample of the differences in the data across brands and channels.
    - Lloyds - Retail, Retail Business Banking (RBB), Commercial
    - Halifax – Retail
    - BoS - Retail, Retail Business Banking (RBB), Commercial
    - MBNA – Retail

1.2.2 MOCK DATA

- Mock data includes the PSUs along with their accounts, transactions, balances, beneficiaries, direct debits, standing orders, scheduled payments and product.
- There is no relationship between the mock data and any real-world user details or credentials.

1.2.3 TPP ONBOARDING

- The TPPs must be registered to the Open Banking Sandbox directory (see Onboarding below), whereas, to access the production interface, TPPs must be registered with the Open Banking Production directory.
- Only one app registration per software statement is possible in LBG Sandbox, whereas the production environment supports multiple app registration per software statement.
- LBG Sandbox only supports PS256 signed SSAs without any backward compatibility for RS256.

1.2.4 API SUBSCRIPTION

- Manual API subscription is not required in the Sandbox.
- TPP applications are automatically subscribed to the LBG Sandbox APIs based on the permissions contained within their SSA. This is different to LBG Production environment where the TPPs must manually subscribe to the APIs in order to access them.

1.2.5 PSU CONSENT JOURNEY

- TPPs are not required to be an LBG customer to use the Sandbox. Pre-defined PSU accounts have been established for TPP's use as listed under 1.2.1.
- The Sandbox only supports Primary authentication.
- The Sandbox does not support:
    - Step Up authentication
    - Two Factor authentication
    - App to App consent
    - Decoupled flows

1.2.6 PISP API - PAYMENT EXECUTION

- Though the Sandbox supports the PISP APIs, the payment transactions will not affect the actual account balance of a PSU. Hence the account balance will not change over time, irrespective of the number of times the TPP invokes the payment APIs.

1.2.7 UNSUPPORTED FEATURES

- Customer re-authorization
- Fraud checks
- Daily limit checks

1.2.8 STRESS TESTING

- There is sufficient data capacity and performance to facilitate effective and realistic TPP testing. However, the Sandbox does not provide the capacity to replicate production volumes and hence cannot be used for stress/load testing.

2 COMPLETING AN END-TO-END API JOURNEY

This section explains how to on-board onto the Lloyds Banking Group Sandbox, and how to complete an end-to-end API journey. In this instance, the example used is an Account Information API journey.

2.1 ON-BOARDING ONTO THE SANDBOX

2.1.1 FIRST TIME USERS

- To use the Sandbox, TPPs must be registered with the Open Banking Sandbox Directory
- TPPs should get familiarized with the following documents:
  - Open Banking Directory Specification v1.2
  - Open Banking Security Profile v1.1.2
  - Open Banking Read/Write Data API Specification

2.1.2 PRE-REQUISITES FOR REGISTERING APPLICATIONS IN THE SANDBOX

- Certificate Signing Request (CSR) and private keys for transport and signing.
  - TPPs should use their own key generation tool and management policies and must create the CSR and the private keys, each for transport and signing.
- Software Statement Assertion (SSA) to identify the TPP's Open Banking application.
  - Navigate to the organization summary page on the Open Banking Sandbox Directory
  - Click 'Add new statement' in Software Statements section.
  - Complete 'Add new statement form'
  - Upload the CSRs to Open Banking to get the Software Statement Assertion (SSA).
  - Check if the new Software Statement has been completed.
  - Generate the SSA and copy to clipboard for future use in on-boarding.
- Digital certificates(PEM files) required for transport and signing.
  - To retrieve the certificates, decrypt the SSA using tools like https://jwt.io/ and get the "software_jwks_endpoint". Download the PEM files for transport and signing using the "software_jwks_endpoint".

2.1.3 REGISTER AN APPLICATION WITH LBG SANDBOX

Once you are registered on the Open Banking Sandbox Directory you can access the LBG Sandbox:

- To use the Sandbox APIs, TPPs must have an application identified by an SSA which is used to get the Client ID and Client Secret (to identify who is calling the APIs).
- Navigate to the Sandbox tab in the LBG Developer Portal Support Page
- Click the “Sandbox” button to login to the Sandbox using Open Banking Sandbox credentials.
- To create an application, click on the ‘+’ icon (in the bottom right hand corner).
- When creating a new application, give it a name, a description and a redirect URI (should match the redirect URI in the SSA) and the SSA which was created above.
- Once the application is created successfully, it will be displayed in the listing with its registered name. The row should be clicked to reveal the Client ID and the Client Secret.
- The application is automatically subscribed to all the available APIs, dependent on the credential type available in the SSA (e.g. AISP, PISP, CBPII) and is ready for testing.
- There is an option available to delete the registered applications within the listing.

2.1.4 SELECTING PSU CREDENTIALS

- After logging into the Sandbox, TPPs will see a PSU Page link in the header navigation that will take them to the Lloyds Banking Group (LBG) branded PSU credential information page.
- This page will have a title and introduction text, followed by the PSU credentials.
- The PSU credential information will include usernames for all available LBG brands and channels.
- The password is the same for all the PSUs.
- This page will remain accessible to TPPs for future reference.

2.2 LIST OF APIS AVAILABLE FOR TESTING

The following list of APIs are in scope for testing

AISP

Service	Endpoints	METHOD	V2.0	V3.1
Accounts	/accounts	GET	Yes	Yes
Accounts	/accounts/{AccountId}	GET	Yes	Yes
Accounts	/account-requests	POST	Yes	NA
Accounts	/account-requests/{AccountRequestId}	GET	Yes	NA
Accounts	/account-requests/{AccountRequestId}	DELETE	Yes	NA
Accounts	/account-access-consents	POST	NA	Yes
Accounts	/account-access-consents/{ConsentId}	GET	NA	Yes
Accounts	/account-access-consents/{ConsentId}	DELETE	NA	Yes
Balances	/accounts/{AccountId}/balances	GET	Yes	Yes
Transactions	/accounts/{AccountId}/transactions	GET	Yes	Yes
Beneficiaries	/accounts/{AccountId}/beneficiaries	GET	Yes	Yes
Direct Debits	/accounts/{AccountId}/direct-debits	GET	Yes	Yes
Standing Orders	/accounts/{AccountId}/standing-orders	GET	Yes	Yes
Products	/accounts/{AccountId}/product	GET	Yes	Yes
Offers	/accounts/{AccountId}/offers	GET	NA	Yes

Statements	/accounts/{AccountId}/statements	GET	NA	Yes
Statements	/accounts/{AccountId}/statements/{StatementId}/file	GET	NA	Yes
Scheduled Payments	/accounts/{AccountId}/scheduled-payments	GET	NA	Yes

CBPII

Service	Endpoints	METHOD	V2.0	V3.1
Confirmation of Funds	/funds-confirmation	POST	NA	Yes
Confirmation of Funds	/funds-confirmations-consents	POST	NA	Yes
Confirmation of Funds	/funds-confirmation-consents/{ConsentId}	GET	NA	Yes
Confirmation of Funds	/funds-confirmation-consents/{ConsentId}	DELETE	NA	Yes

PISP

Service	Endpoints	METHOD	v1.1	V3.1
Payment Initiation	/payments	POST	Yes	NA
Payment Initiation	/payment-submissions	POST	Yes	NA
Domestic Payments	/domestic-payment-consents	POST	NA	Yes
Domestic Payments	/domestic-payment-consents/{ConsentId}	GET	NA	Yes
Domestic Payments	/domestic-payments	POST	NA	Yes
Domestic Payments	/domestic-payments/{DomesticPaymentId}	GET	NA	Yes

Domestic Scheduled Payments	/domestic-scheduled-payment-consents	POST	NA	Yes
Domestic Scheduled Payments	/domestic-scheduled-payment-consents/{ConsentId}	GET	NA	Yes
Domestic Scheduled Payments	/domestic-scheduled-payments	POST	NA	Yes
Domestic Scheduled Payments	/domestic-scheduled-payments/{DomesticScheduledPaymentId}	GET	NA	Yes
Domestic Standing Orders	/domestic-standing-order-consents	POST	NA	Yes
Domestic Standing Orders	/domestic-standing-order-consents/{ConsentId}	GET	NA	Yes
Domestic Standing Orders	/domestic-standing-orders	POST	NA	Yes
Domestic Standing Orders	/domestic-standing-orders/{DomesticStandingOrderId}	GET	NA	Yes
International Payments	/international-payment-consents	POST	NA	Yes
International Payments	/international-payment-consents/{ConsentId}	GET	NA	Yes
International Payments	/international-payments	POST	NA	Yes
International Payments	/international-payments/{InternationalPaymentId}	GET	NA	Yes
International Scheduled Payments	/international-scheduled-payment-consents	POST	NA	Yes
International Scheduled Payments	/international-scheduled-payment-consents/{ConsentId}	GET	NA	Yes
International Scheduled Payments	/international-scheduled-payments	POST	NA	Yes
International Scheduled Payments	/international-scheduled-payments/{InternationalScheduledPaymentId}	GET	NA	Yes

International Standing Order APIs and File Payment APIs are not available.

2.3 PSU CONSENT USING HYBRID FLOW

- As part of PSU consent, TPP will be redirected to the LBG login page.
- TPP will provide the PSU credentials in the login page. These credentials are available in the PSU page as specified under section 2.1.4
- Based on the API request (AISP, PISP and CBPII) TPP will see the relevant screen post Login page.
- AISP Consent Page.
    - Available accounts for the PSU are listed with an option to choose specific accounts for consent.
    - The page will have additional details on the information to be shared with the TPPs
    - Options are available to either agree by clicking the ‘Continue’ button or to cancel the request.
- PISP Consent Page
    - Payment details page will be shown with the amount to be paid to the merchant/PISP provider along with the bank details
    - An account can be selected from where the amount can be debited
    - Option is available to either agree to pay by clicking the ‘Continue’ button or to cancel the request
- CoF Consent Page
    - The account number sent by TPP for the confirmation of funds will be displayed on the screen.
    - The page will have expiry date for the consent to be shared with the TPPs (The expiry date displayed will depend on the customer choice or will display as No Expiry date)
    - The note related to confirmation of funds is displayed.
    - Options are available to either agree by clicking the ‘Continue’ button or to cancel the request.

2.4 OTHER USEFUL INFORMATION

1. LBG Sandbox APIs are behind the following domains – these are over MA-TLS – hence the TPPs should present their OB-signed client cert against these domains to validate if the SSL handshake is working as expected (e.g. by using an appropriate openSSL command, in chrome etc.)

   https://matls.as.aspsp.sandbox.lloydsbanking.com/open-banking/mtlsTest

   https://matls.as.aspsp.sandbox.lloydsbanking.com:443/open-banking/mtlsTest

   https://matls.rs.aspsp.sandbox.lloydsbanking.com/open-banking/mtlsTest

   https://matls.rs.aspsp.sandbox.lloydsbanking.com:443/open-banking/mtlsTest

2. Discovery Endpoints

   a. API discovery
      https://rs.aspsp.sandbox.lloydsbanking.com/open-banking/discovery

   b. Well-known endpoint
      https://as.aspsp.sandbox.lloydsbanking.com/oauth2/.well-known/openid-configuration

      Please note that as part of token end point call, Client ID should not be populated in the body of the request since only the client_secret_basic is supported.

      "token_endpoint_auth_methods_supported": ["client_secret_basic"]

3. We support OIDC Hybrid Flow.

4. All APIs are secured with TLS Mutual Authentication where Certs are signed by Open Banking CA.

5. We do not support dynamic client registration.

6. The authorize url journey in the browser uses Open Banking signed certificates. Hence the Open Banking Sandbox root and issuer certificates should be added to the trust store to avoid untrusted certificate issues in the browser. The Open Banking sandbox root and issuer certificates are available for download from the Open Banking confluence page

2.5 TESTING THE ACCOUNT INFORMATION APIS

2.5.1 SETUP CLIENT CREDENTIALS TOKEN

In this step, the AISP obtains an Access Token using the “client_credentials” grant type. When an Access Token expires, you will need to re-request for another Access Token.

```
curl -X POST \
--key <TRANSPORT_PRIVATE_KEY_FILE> \
--cert <TRANSPORT_PEM_FILE> \
  https://matls.as.aspsp.sandbox.lloydsbanking.com/oauth2/access_token \
  -H 'Authorization: Basic <BASE64_ENCODED_CLIENT_ID:CLIENT_SECRET>' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -H 'cache-control: no-cache' \
  -d 'grant_type=client_credentials&scope=accounts'
```

Headers

Parameter	Example Value	Description
Authorization	'Authorization: Basic MmIyYmJlZmEtNWNjZC00NzM5LTg4MTEtND…..	The authorization token is the base64 encoded value of the Client Id and Client Secret obtained during onboarding

Data

Parameter	Example Value	Description
grant_type	client_credentials	The grant type being requested
Scope	Accounts	The scope being requested.

Note down the “access_token” in the response for further use.

2.5.2 INVOKE THE ACCOUNT REQUEST API

Using the Access Token obtained in the previous step, invoke the Accounts Consent API with the desired permissions in the request body.

```
curl -X POST \
--key <TRANSPORT_PRIVATE_KEY_FILE> \
--cert <TRANSPORT_PEM_FILE> \
  https://matls.rs.aspsp.sandbox.lloydsbanking.com/open-
banking/v3.1/aisp/account-access-consents \
  -H 'Accept: application/json' \
  -H 'Authorization: Bearer <ACCESS_TOKEN_FROM_THE_PREVIOUS_STEP>' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -H 'x-fapi-financial-id: <Value of 'FinancialId' from the API Discovery
Endpoint>' \
  -d '{
  "Data": {
    "Permissions": [
      "ReadAccountsDetail",
      "ReadBalances"
    ]
  },
  "Risk": {}
}'
```

Headers

Parameter	Example Value	Description
authorization	Bearer AAIkMDM5NDJmZTUtOGNiMi00NzVmLWIwMTItNDgyZjM0ZTE...	The "access_token" obtained in the previous step.

Body

Parameter	Example Value	Description
Data	{ "Data": { "Permissions": ["ReadAccountsDetail", "ReadBalances"] }, "Risk": {} }	The permissions being requested.

Please refer here for a full list of the allowed permissions.

Note down the AccountRequestID or ConsentId in the response data. This is required in the next step to be used in the claims body.

2.5.3 AUTHORISE CONSENT – HYBRID FLOW

In this step, create the authorization request (using a signed JWT request containing the AccountRequestId or ConsentId as a claim) for the customer to consent to the account consent request.

Using a web browser, invoke the URL below to emulate the customer giving consent after changing the parameterized values as required.

```
https://as.aspsp.sandbox.lloydsbanking.com:443/oauth2/authorize?response_type=code%20id_token&client_id=<CLIENT_ID_FROM_THE_FIRST_STEP>&state=<STATE>&nonce=<NONCE>&scope=openid%20accounts&redirect_uri=<REDIRECT_URI_USED_DURING_ONBOARDING>&request=<SIGNED_CLAIMS_IN_BASE64>
```

URL Parameters

Parameter	Example Value	Description
response_type	code id_token	The OAuth flow type being used
client_id	10d260bf-a7d9-444a-92d9-7b7a5f088208	The client ID of your application registered in the Sandbox
State	12345	The state as specified by the TPP
Scope	openid accounts	The scope being requested.
redirect_uri	https://example.com/redirect	The redirect URL of the application registered in the Sandbox
Nonce	4987594875485-j	The nonce as specified by the TPP
request	eyJhbGciOiJSUzI1NiIsImtpZCI6Ikd4bElpd2lhblZxc0R1dXNoZ2pFME9UVXhPVGsifQ.eyJpc3MiOiJodHRwczovL2FwaS5hbHBoYWJhbmsuY29tIiwiYXVkIjoiczZCaGRSa3F0MyIsInJlc3BvbnNlX3R5cGUiOi…..	This is an OIDC request object containing a claim (sample given below) identifying the accounts that needs to be authorised. Please ensure it is signed using the signing kid, signing private key and the PS256 algorithm. To modify it to support your

Parameter	Example Value	Description
		test scenario use the debugger at https://jwt.io

A sample claims body syntax that may be used for signing (Replace <xxxx> values with actuals):

```
{
    "sub": "urn-example-bank",
    "exp": <EXPIRY_IN_SECONDS_SINCE_EPOCH>,
    "iat": <ISSUED_AT_IN_SECONDS_SINCE_EPOCH>,
    "iss": "<CLIENT_ID_FROM_THE_FIRST_STEP>",
    "aud": "<Value of the “issuer” in the well-known endpoint>",
    "response_type": "code id_token",
    "client_id": "<CLIENT_ID_FROM_THE_FIRST_STEP>",
    "redirect_uri": "<REDIRECT_URI_USED_DURING_ONBOARDING>",
    "scope": "openid accounts",
    "state": "<STATE>",
    "nonce": "<NONCE>",
    "claims": {
        "userinfo": {
            "openbanking_intent_id": {
                "value": "<ACCOUNT_REQ_ID_OR_CONSENT_ID>",
                "essential": true
            }
        },
        "id_token": {
            "openbanking_intent_id": {
                "value": "<ACCOUNT_REQ_ID_OR_CONSENT_ID>",
                "essential": true
            },
            "acr": {
                "value": "urn:openbanking:psd2:sca",
                "essential": true
            }
        }
    }
}
```

Follow the UI flow using the credentials of any of the static PSU user accounts supplied with the Sandbox as described under section 2.1.4

If there were untrusted certificate issues when the authorize url was launched in the browser, ensure instructions under section 2.4(6) is completed.

At the end of the flow, take note of the authorization “code” and id token appended to the URL

e.g.
https://example.com/redirect?code=**xxxxxxxxxxxxx**&id_token=xxxxxxxxxx

2.5.4 GET THE ACCESS TOKEN REQUIRED TO ACCESS THE ACCOUNT

Following the OAuth2.0 protocol, exchange the access code for the access token required in the next step, using the “authorization_code” grant type.

```
curl -X POST \
--key <TRANSPORT_PRIVATE_KEY_FILE> \
--cert <TRANSPORT_PEM_FILE> \
  https://matls.as.aspsp.sandbox.lloydsbanking.com/oauth2/access_token \
  -H 'Authorization: Basic <BASE64_ENCODED_CLIENT_ID:CLIENT_SECRET>' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -H 'cache-control: no-cache' \
  -d
'grant_type=authorization_code&code=<CODE_FROM_THE_PREVIOUS_STEP>&redirect_
uri=<REDIRECT_URI_USED_DURING_ONBOARDING>'
```

Headers

Parameter	Example Value	Description
authorization	'Authorization: Basic MmIyYmJlZmEtNWNjZC00NzM5LTg4MTEtND…..	The authorization token is the base64 encoded value of the Client Id and Client Secret

Body

Parameter	Example Value	Description
grant_type	authorization_code	The grant type being requested
redirect_uri	https://example.com/redirect	The redirect URL which must match that of the application registered in the Sandbox.
code	21a0ff30-3adb-423a…..	The authorization code retrieved in step 3

Take note of the “access_token” in the response data for further use.

2.5.5 GET ACCOUNTS

You can use the Access Token to retrieve data using the AISP APIs.

The following example is from the Account API v3.1 Specification.

Where the initial Access Token expires, you must obtain a new Access Token (refresh tokens are not supported).

```
curl -X GET \
--key <TRANSPORT_PRIVATE_KEY_FILE> \
--cert <TRANSPORT_PEM_FILE> \
  https://matls.rs.aspsp.sandbox.lloydsbanking.com/open-
banking/v3.1/aisp/accounts \
  -H 'Accept: application/json' \
  -H 'Authorization: Bearer <ACCESS_TOKEN_FROM_THE_PREVIOUS_STEP>' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -H 'x-fapi-financial-id: <Value of ‘FinancialId’ from the API Discovery
Endpoint>' \
  -H 'x-idempotency-key: <IDEMPOTENCY_KEY>'
```

Headers

Parameter	Example Value	Description
Authorization	Bearer xyzAbcDef…	The “access_token” obtained in the previous step