

OPEN BANKING

OBIE Read/Write API

An Overview

27th April, 2020

Freddi Gyara

© Open Banking Limited 2020

OB v FAPI security profile

The following are high level / key differences between the two profiles:

- **Client Authentication Algorithms:** The old OB profile allowed for client authentication using client secrets. This is not sufficiently secure as a client secret is a shared secret. FAPI only allows MTLS and private_key_jwt which rely on asymmetric keys
- **Signing algorithms:** OB profile allowed for RS256. FAPI only permits PS256 and ES256. Currently, OBIE directory cannot issue ES certs which constrains us somewhat to PS256
- **Grant types:** OB profile allowed for authorization code grant, but recommended hybrid flow. FAPI requires hybrid flow
- **Request objects:** FAPI has a comprehensive set of requirements on the request object (e.g. having an exp claim). OB profile only stated the need to use a signed request object, but did not go into the details of the JOSE headers etc.

Overview on a Page



Version 3.1.5

<https://standards.openbanking.org.uk/>

Accounts



- Accounts
- Balances
- Transactions
- Statements
- Beneficiaries
- Direct Debits
- Standing Orders
- Scheduled Payments
- Products & Offers
- Parties

Payments



- Domestic
- International
- Immediate
- Future-dated
- Standing orders
- Bulk payments

CBP II



- Consents
- Funds checks

Events



- Push notifications
- Aggregated polling

Dynamic Client Reg



- OBIE SSA
- Non-OBIE SSA
- Eidas based

OPEN BANKING

Dynamic Client Registration

Now supports multiple certificate types and trust anchors!

	① OBIE Directory SSA + OBIE WAC	② OBIE Directory SSA + QWAC	③ Custom SSA + QWAC
TPP registered on OBIE directory	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Directory hosted JWKS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
SSA issued by OBIE directory	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DCR payload signed by signing key on JWKS directory	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DCR payload signed by external signing key e.g. QSealC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

OPEN BANKING

OPEN BANKING

OBIE Functional Conformance Tool (FCT)

27th April, 2020

Julian Coombes / Glyn Jackson

© Open Banking Limited 2020

Background

- OB has created a number of open source tools to allow implementers to test against the Open Banking Standards
- The tools also allow ASPSPs to submit results to Open Banking for certification
- All the tools are open source and available on Bitbucket.org
- <https://bitbucket.org/openbankingteam/conformance-suite>
- The tools are run in Docker containers making them easy to execute
- We've developed two tools - the Functional Conformance Tool and the Dynamic Client Registration Tool. Today we'll be focusing on the Functional Conformance tool

Introduction

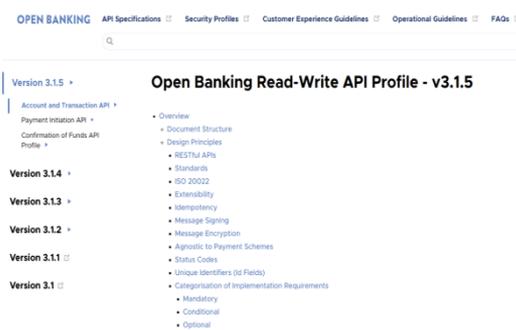
- OB is about Banking Standards
- Specifications define our Standards
- OpenAPI/Swagger help define the API interfaces for a specification
- FCT tests OB APIs for Accounts, Payments and Confirmation of Funds
- The tool uses the information in the Specifications and Swagger definitions to build tests that interrogate an ASPSP implementation of our Standards



OPEN BANKING

Welcome to the Open Banking Standard

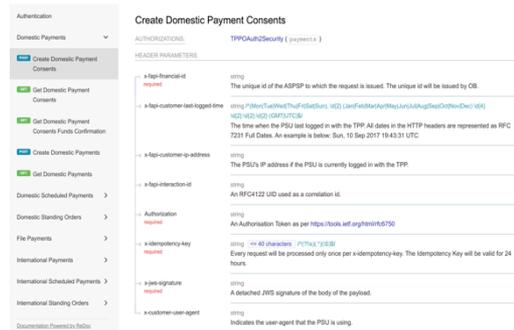
The Standard is designed to assist any European account providers in meeting their PSD2 and RTS requirements as well as supporting their application for an exemption from the contingency mechanism. This market-enabling Standard is built in an optional modular format to most effectively meet consumer and market needs.



OPEN BANKING API Specifications Security Profiles Customer Experience Guidelines Operational Guidelines FAQs

Open Banking Read-Write API Profile - v3.1.5

- Version 3.1.5
 - Account and Transaction API
 - Payment Initiation API
 - Confirmation of Funds API Profile
 - Version 3.1.4
 - Version 3.1.3
 - Version 3.1.2
 - Version 3.1.1
 - Version 3.1
- Overview
 - Document Structure
 - Design Principles
 - RESTful APIs
 - Standards
 - ISO 20022
 - Extensibility
 - Idempotency
 - Message Signing
 - Message Encryption
 - Agnostic to Payment Schemes
 - Status Codes
 - Unique Identifiers (Id Fields)
 - Category of Implementation Requirements
 - Mandatory
 - Conditional
 - Optional



Authentication

Domestic Payments

- Create Domestic Payment Consents
- Get Domestic Payment Consents
- Get Domestic Payment Consents Funds Confirmation
- Create Domestic Payments
- Get Domestic Payments

Domestic Scheduled Payments

Domestic Standing Orders

File Payments

International Payments

International Scheduled Payments

International Standing Orders

Documentation Developer Guide

Create Domestic Payment Consents

AUTHORIZATIONS TPPAuthSecurity (payments)

HEADER PARAMETERS

- App-branch-id: string. The unique id of the ASPSP to which the request is issued. The unique id will be issued by OB.
- App-customer-tenant-logged-in: boolean. A flag to indicate if the customer is logged in.
- App-customer-q-address: string. The PSU's IP address if the PSU is currently logged in with the TPP.
- App-transaction-id: string. An RFC4122 UID used as a correlation id.
- Authorization: string. An Authorisation Token as per https://tools.ietf.org/html/6750
- Idempotency-key: string. An Idempotency Key (IDK) used to ensure that every request will be processed only once per x-idempotency-key. The Idempotency Key will be valid for 24 hours.
- Json-signature: string. A detached JWS signature of the body of the payload.
- Customer-user-agent: string. Indicates the user-agent that the PSU is using.

OPEN BANKING

Functional Conformance Tool

Two things determine how an API Implementation should behave :-

1. Guidance and definitions in the API Specification
2. Swagger/OpenAPI file, a formal definition of :-
 - EndPoints
 - Response codes
 - Data formats

The tool checks the Swagger Endpoint constraints and the wider API behaviours defined in the Specifications that can't easily be expressed in Swagger.

Things that can't easily be expressed in Swagger:

- Results of a sequence of calls
- Time related concerns, eg. token expiry, Payment dates
- Pagination rules for transaction listings

What the tool provides

The tool consists of 3 main parts:-

- Discovery
- Testing
- Reporting

Discovery - builds a picture of the ASPSP implementation - which endpoints are supported, authentication methods, server locations etc.

Testing - allows us to craft any request we like and examine any aspect of an ASPSPs response

Reporting - captures the results of the tests for submission to OB

Structure

The tool uses JSON to define all inputs and outputs.

Discovery

- Discovery.json - Endpoint definitions for the API under test
- Config.json - Credentials, certs, account

Testing

- JSON is used to define testcase collections for each api set
- The testcases are then filtered by the discovery endpoints to get the run set of test

Reporting

- Report.json file - contains the results of running the tool - showing each test passed and failed

Test cases

A Simple TestCase:

- Defined in JSON
- "input" - defines request
- "expect" - defines tests
- Calls /accounts endpoint
- Response checks:-
 - Status code 200
 - Swagger response validation
 - x-fapi-interaction-id present

```
1 {
2     "@id": "OB-301-ACC-100000",
3     "name": "Check Account endpoint",
4     "input": {
5         "method": "GET",
6         "endpoint": "/accounts/${accountId}",
7         "headers": {
8             "Authorization": "Bearer $accountToken0001",
9             "x-fapi-financial-id": "$fapiFinancialId",
10            "x-fapi-interaction-id": "$interactionId",
11            "x-fcs-testcase-id": "OB-301-ACC-100000"
12        }
13    },
14    "expect": {
15        "status-code": 200,
16        "schema-validation": true,
17        "matches": [
18            {
19                "header-present": "x-fapi-interaction-id"
20            }
21        ]
22    }
23 }
24
```

Prerequisites

Prerequisites for running the tool:

- Docker - The FCT runs in a docker container
- OB Transport and Signing Certificates
- Valid account credentials
- A working .well-known/openid-configuration containing authentication mechanisms, signing methods etc.

The following command will launch the FCT :-

```
docker run --rm -it -p 8443:8443 "openbanking/conformance-suite:v1.4.0"
```

Pointing a browser at <https://localhost:8443> will show the FCT UI.

OPEN BANKING

Thank you

www.openbanking.org.uk