Open Bank Project

Introductory System Documentation

This document serves as an overview of the Open Bank Project (OBP) technology ecosystem and related tools. It provides an introduction to its key components, architecture, deployment and management approaches and capabilities.

For more detailed information or the sources of truths, please refer to the individual repository READMEs or contact TESOBE for commercial licences and support.

Version: 0.5.0 **Last Updated:** Oct 2025 **License:** Copyright TESOBE GmbH 2025 - AGPL V3

Table of Contents

---

1. Executive Summary

1.1 About Open Bank Project

The Open Bank Project (OBP) is an open-source RESTful API platform for banks that enables Open Banking, PSD2, XS2A, and Open Finance compliance. It provides a comprehensive ecosystem for building financial applications with standardized API interfaces.

Core Value Proposition:

- **Tagline:** "Bank as a Platform. Transparency as an Asset"
- **Mission:** Enable account holders to interact with their bank using a wider range of applications and services
- **Key Features:**
    - **Transparency & Privacy**: Configurable data sharing with views, data blurring to preserve sensitive information
    - **Data Enrichment**: Add tags, comments, images, and metadata to transactions
    - **Multi-Bank Abstraction**: Universal API layer across different core banking systems
    - **Flexible Authentication**: OAuth 1.0a, OAuth 2.0, OpenID Connect, Direct Login, Gateway Login
    - **Comprehensive Banking APIs**: 1000+ endpoints covering accounts, payments, customers, KYC, cards, products
    - **Real-Time & Batch Operations**: Support for both synchronous and asynchronous processing

1.2 Core Feature Categories

1.2.1 Account & Banking Operations

- **Account Management**: Account creation, updates, attributes, account holders, account applications
- **Balance & Transaction History**: Real-time balances, transaction retrieval with rich filtering
- **Multi-Account Support**: Manage multiple accounts across different banks
- **Account Views**: Granular permission system (Owner, Public, Accountant, Auditor, custom views)
- **Account Attributes**: Flexible key-value attribute system for extending account metadata

1.2.2 Payment & Transfer Services

- **Transaction Requests**: SEPA, COUNTERPARTY, SANDBOX, FREE_FORM, ACCOUNT, ACCOUNT_OTP
- **Payment Initiation**: Single and bulk payments with SCA (Strong Customer Authentication)
- **Standing Orders**: Recurring payment management
- **Direct Debits**: Direct debit mandate management
- **Transaction Challenges**: OTP and challenge-response for payment authorization
- **Signing Baskets**: Batch payment approval workflows
- **Transaction Attributes**: Custom metadata on transactions

1.2.3 Customer & KYC Management

- **Customer Profiles**: Comprehensive customer data management
- **Customer Attributes**: Extensible customer metadata (address, DOB, dependants, tax residence)
- **Customer-Account Linking**: Associate customers with accounts
- **KYC Processes**: KYC checks, documents, media uploads, status tracking
- **CRM Integration**: Customer relationship management features
- **Consent Management**: PSD2-compliant consent workflows for data access

1.2.4 Card Management

- **Card Lifecycle**: Create, update, retrieve card information
- **Card Attributes**: Flexible metadata for cards (limits, features, preferences)
- **Card Controls**: Activate, deactivate, set limits

1.2.5 Product & Fee Management

- **Banking Products**: Accounts, loans, credit cards, savings products
- **Product Attributes**: Configurable product metadata
- **Product Collections**: Group products into collections
- **Fee Management**: Product fees, charges, pricing information
- **Product Catalog**: Searchable product offerings

1.2.6 Branch & ATM Services

- **Branch Management**: Branch locations, opening hours, services, attributes
- **ATM Management**: ATM locations, capabilities, accessibility features
- **Location Services**: Geographic search and filtering
- **Service Availability**: Real-time status and feature information

1.2.7 Authentication & Authorization

- **Multiple Auth Methods**: OAuth 1.0a, OAuth 2.0, OpenID Connect (OIDC) with multiple concurrent providers, Direct Login, Gateway Login
- **Consumer Management**: API consumer registration and key management
- **Token Management**: Access token lifecycle management
- **Consent Management**: PSD2-compliant consent workflows (AIS, PIS, PIIS)
- **User Locks**: Account security with failed login attempt tracking

1.2.8 Access Control & Security

- **Role-Based Access Control**: 334+ granular static roles
- **Entitlements**: Fine-grained permission system for API access
- **Entitlement Requests**: User-initiated permission request workflows
- **Views System**: Account data visibility control (owner, public, accountant, auditor, custom)
- **Scope Management**: OAuth 2.0 scope definitions
- **Authentication Type Validation**: Enforce authentication requirements per endpoint

1.2.9 Extensibility & Customization

- **Dynamic Endpoints**: Create custom API endpoints without code deployment
- **Dynamic Entities**: Define custom data models dynamically
- **Dynamic Resource Documentation**: Custom endpoint documentation
- **Dynamic Message Docs**: Custom connector message documentation
- **Endpoint Mapping**: Route custom paths to existing endpoints
- **Connector Architecture**: Pluggable adapters for different banking systems
- **Method Routing**: Route connector calls to different implementations

1.2.10 Regulatory & Compliance

- **Multi-Standard Support**: Open Bank Project, Berlin Group NextGenPSD2, UK Open Banking, Bahrain OBF, STET, Polish API, AU CDR, Mexico OF
- **PSD2 Compliance**: SCA, consent management, TPP access
- **Regulated Entities**: Manage regulatory registrations
- **Tax Residence**: Customer tax information management
- **Audit Trails**: Comprehensive logging and tracking

1.2.11 Integration & Interoperability

- **Webhooks**: Event-driven notifications for account and transaction events
- **Foreign Exchange**: FX rate management and conversion
- **API Collections**: Group related endpoints into collections
- **API Versioning**: Multiple concurrent API versions (v1.2.1 through v6.0.0+)
- **Standard Adapters**: Pre-built integrations for common banking standards

1.2.12 Performance & Scalability

- **Rate Limiting**: Consumer-based rate limits (Redis or in-memory)
- **Caching**: Multi-layer caching strategy with ETags
- **Metrics & Monitoring**: API usage metrics, performance tracking
- **Database Support**: PostgreSQL, Oracle, MySQL, MS SQL Server, H2
- **Akka Integration**: Actor-based concurrency model
- **Connection Pooling**: Efficient database connection management

1.2.13 Developer Experience

- **API Explorer**: Interactive API documentation and testing (Vue.js/TypeScript)
- **Swagger/OAS**: OpenAPI specification support
- **Sandbox Mode**: Test environment with mock data
- **Comprehensive Documentation**: Glossary, Resource Docs with Auto-generated API docs from code
- **API Manager**: Django-based admin interface for API governance
- **Web UI Props**: Configurable UI properties

1.2.14 Advanced Features

- **Counterparty Limits**: Transaction limits for counterparties
- **User Refresh**: Synchronize user data from external systems
- **Migration Tools**: Database migration and data transformation utilities
- **Scheduler**: Background job processing
- **AI Integration**: Opey II conversational banking assistant
- **Blockchain Integration**: Cardano blockchain connector support
- **Metadata and Search**: Metadata and Full-text search across transactions and entities
- **Social Media**: Social media handle management for accounts

1.2.15 Technical Capabilities

- **Multi-Standard Support:** Open Bank Project, Berlin Group NextGenPSD2, UK Open Banking, Bahrain OBF, STET PSD2, Polish API, AU CDR, Mexico OF
- **Authentication Methods:** OAuth 1.0a, OAuth 2.0, OpenID Connect (OIDC) with multiple concurrent providers, Direct Login, Gateway Login
- **Extensibility:** Dynamic endpoints, dynamic entities, connector architecture, method routing
- **Rate Limiting:** Built-in support with Redis or in-memory backends
- **Multi-Database Support:** PostgreSQL, Oracle, MySQL, MS SQL Server, H2
- **Internationalization:** Multi-language support
- **API Versions:** Multiple concurrent versions (v1.2.1 through v6.0.0+)
- **Deployment Options:** Standalone, Docker, Kubernetes, cloud-native
- **Data Formats:** JSON, XML support
- **Error Handling:** 400+ distinct error codes with detailed messages

1.3 Key Components

- **OBP-API:** Core RESTful API server (Scala/Lift framework)
- **API Explorer:** Interactive API documentation and testing tool (Vue.js/TypeScript)
- **API Manager:** Administration interface for managing APIs and consumers (Django/Python)
- **Opey II:** AI-powered conversational banking assistant (Python/LangGraph)
- **OBP-OIDC:** Development OpenID Connect provider for testing
- **Keycloak Integration:** Production-grade OIDC provider support

---

2. System Architecture

2.1 High-Level Architecture

```
┌──────────────────┐  ┌──────────────────┐  ┌──────────────────┐
┌────────────────────────────────┐
│   API Explorer   │  │   API Manager    │  │    Opey II       │  │   Client
Applications        │
│   (Frontend)     │  │   (Admin UI)     │  │   (AI Agent)     │  │ (Web/Mobile Apps,
TPP, etc.)  │
└─────────┬────────┘  └─────────┬────────┘  └─────────┬────────┘
└─────────┬──────────────────────┘
          │                     │                     │                     │
          └─────────────────────┴─────────────────────┴─────────────────────┘
                                │
                                │ HTTPS/REST API
                                │
                      ┌─────────┴──────────────────────────┐
                      │          OBP API Dispatch          │
                      └─────────┬──────────────────────────┘
                                │
                                ▼
                      ┌────────────────────────────────────┐
                      │            OBP-API Core            │
                      │       (Scala/Lift Framework)       │
                      │                                    │
                      │  ┌──────────────────────────────┐  │
```

```
              │  │   Client Authentication         │  │
              │  │   (Consumer Keys, Certs)        │  │
              │  └──────────────┬──────────────────┘  │
              │                 │                     │
              │  ┌──────────────▼──────────────────┐  │
              │  │      Rate Limiting              │  │
              │  └──────────────┬──────────────────┘  │
              │                 │                     │
              │  ┌──────────────▼──────────────────┐  │
              │  │   Authentication Layer          │  │
              │  │   (OAuth/OIDC/Direct)           │  │
              │  └──────────────┬──────────────────┘  │
              │                 │                     │
              │  ┌──────────────▼──────────────────┐  │
              │  │   Authorization Layer           │  │
              │  │   (Roles & Entitlements)        │  │
              │  └──────────────┬──────────────────┘  │
              │                 │                     │
              │  ┌──────────────▼──────────────────┐  │
              │  │      API Endpoints              │  │
              │  │      (Multiple Versions)        │  │
              │  └──────────────┬──────────────────┘  │
              │                 │                     │
              │  ┌──────────────▼──────────────────┐  │
              │  │      Views                      │  │
              │  │      (Data Filtering)           │  │
              │  └──────────────┬──────────────────┘  │
              │                 │                     │
              │  ┌──────────────▼──────────────────┐  │
              │  │      OBP Models                 │  │
              │  │      (Data Structures)          │  │
              │  └──────────────┬──────────────────┘  │
              │                 │                     │
              │  ┌──────────────▼──────────────────┐  │
              │  │      Connector Layer            │  │
              │  │   (Pluggable Adapters)          │  │
              │  └──────────────┬──────────────────┘  │
              └─────────────────┼─────────────────────┘
                                │
             ┌──────────────────┴──────────────────┐
             │                                     │
             ▼                                     ▼
  ┌─────────────────────┐         ┌───────────────────────────────┐
  │      Direct         │         │      Adapter Layer            │
  │     (Mapped)        │         │      (Any Language)           │
  └──────────┬──────────┘         └──────────────┬────────────────┘
             │                                   │
             └───────────────┬───────────────────┘
                             │
       ┌─────────────────────┼─────────────────────┐
       │                     │                     │
       ▼                     ▼                     ▼
┌─────────────────┐   ┌─────────────────┐   ┌─────────────────────┐
│   PostgreSQL    │   │      Redis      │   │  Core Banking       │
│    (OBP DB)      │   │      (Cache)     │   │      Systems          │
│                  │   │                  │   │  (via Connectors /    │
│                  │   │                  │   │     Adapters)         │
└──────────────────┘   └──────────────────┘   └───────────────────────┘
```

2.2 Component Interaction Workflow

1. **Client Request:** Application sends authenticated API request
2. **Rate Limiting:** Request checked against consumer limits (Redis)
3. **Authentication:** Token validated (OAuth/OIDC/Direct Login)
4. **Authorization:** User entitlements checked against required roles
5. **API Processing:** Request routed to appropriate API version endpoint
6. **Connector Execution:** Data retrieved/modified via connector to backend
7. **Response:** JSON response returned to client with appropriate data views

2.3 Deployment Topologies

Single Server Deployment

```
┌─────────────────────────────────────┐
│         Single Server               │
│                                     │
│  ┌───────────────────────────────┐  │
│  │       OBP-API (Jetty)         │  │
│  └───────────────────────────────┘  │
│  ┌───────────────────────────────┐  │
│  │      PostgreSQL               │  │
│  └───────────────────────────────┘  │
│  ┌───────────────────────────────┐  │
│  │         Redis                 │  │
│  └───────────────────────────────┘  │
└─────────────────────────────────────┘
```

Distributed Deployment with Akka Remote (requires extra licence / config)

```
┌─────────────────┐         ┌─────────────────┐
│   API Layer     │         │   Data Layer    │
│   (DMZ)         │  Akka   │   (Secure Zone) │
│                 │  Remote │                 │
│  OBP-API        │◄───────►│  OBP-API        │
│  (HTTP Server)  │         │  (Connector)    │
│                 │         │                 │
│  No DB Access   │         │  PostgreSQL     │
│                 │         │  Core Banking   │
└─────────────────┘         └─────────────────┘
```

2.4 Technology Stack

Backend (OBP-API):

- Language: Scala 2.12/2.13
- Framework: Liftweb
- Build Tool: Maven 3 / SBT
- Server: Jetty or other Java Application Server
- Concurrency: Akka
- JDK: OpenJDK 11, Oracle JDK 1.8/13

Frontend (API Explorer):

- Framework: Vue.js 3, TypeScript
- Build Tool: Vite
- UI: Tailwind CSS
- Testing: Vitest, Playwright

Admin UI (API Manager):

- Framework: Django 3.x/4.x
- Language: Python 3.x
- Database: SQLite (dev) / PostgreSQL (prod)
- Auth: OAuth 1.0a (OBP API-driven)
- WSGI Server: Gunicorn

AI Agent (Opey II):

- Language: Python 3.10+
- Framework: LangGraph, LangChain
- Vector DB: Qdrant
- Web Framework: FastAPI
- API: FastAPI-based service
- Frontend: OBP Portal

Databases:

- Primary: PostgreSQL 12+
- Cache: Redis 6+
- Development: H2 (in-memory)
- Support: Postgres and any RDBMS

OIDC Providers:

- Production: Keycloak, Hydra, Google, Yahoo, Auth0, Azure AD
- Development/Testing: OBP-OIDC

---

3. Component Descriptions

3.1 OBP-API (Core Server)

Purpose: Central RESTful API server providing banking operations

Key Features:

- Multi-version API support (v1.2.1 - v5.1.0+)
- Pluggable connector architecture
- OAuth 1.0a/2.0/OIDC authentication
- Role-based access control (RBAC)
- Dynamic endpoint creation
- Rate limiting and quotas
- Webhook support
- Sandbox data generation

Architecture Layers:

1. **API Layer:** HTTP endpoints, request routing, response formatting
2. **Authentication Layer:** Token validation, session management
3. **Authorization Layer:** Entitlements, roles, scopes
4. **Business Logic:** Account operations, transaction processing
5. **Connector Layer:** Backend system integration
6. **Data Layer:** Database persistence, caching

Configuration:

- Properties files: `obp-api/src/main/resources/props/`
    - `default.props` - Development
    - `production.default.props` - Production
    - `test.default.props` - Testing

Key Props Settings:

```
# Server Mode
server_mode=apis,portal  # Options: portal, apis, apis,portal

# Connector
connector=mapped  # Options: mapped, kafka, akka, rest, etc.

# Database
db.driver=org.postgresql.Driver
db.url=jdbc:postgresql://localhost:5432/obpdb?user=obp&password=xxx

# Authentication
allow_oauth2_login=true
oauth2.jwk_set.url=http://localhost:9000/obp-oidc/jwks

# Rate Limiting
use_consumer_limits=true
cache.redis.url=127.0.0.1
cache.redis.port=6379
```

```
# Admin
super_admin_user_ids=uuid-of-admin-user
```

3.2 API Explorer

Purpose: Interactive API documentation and testing interface

Key Features:

- Browse all OBP API endpoints
- Interactive API testing with OAuth flow
- Request/response examples
- API collections management
- Multi-language support (EN, ES)
- Swagger integration

Technology:

- Frontend: Vue.js 3 + TypeScript
- Backend: Express.js (Node.js)
- Build: Vite
- Testing: Vitest (unit), Playwright (integration)

Configuration (excerpt):

```
# .env file
PUBLIC_OBP_BASE_URL=http://127.0.0.1:8080
OBP_OAUTH_CLIENT_ID=your-client-id
OBP_OAUTH_CLIENT_SECRET=your-client-secret
APP_CALLBACK_URL=http://localhost:5173/api/callback
PORT=5173
```

Installation:

```
cd OBP-API-EXPLORER/API-Explorer-II
npm install
npm run dev  # Development
npm run build  # Production build
```

Nginx Configuration:

```
server {
    location / {
        root /path_to_dist/dist;
        try_files $uri $uri/ /index.html;
    }
    location /api {
        proxy_pass http://localhost:8085;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

3.3 API Manager

Purpose: Django-based administrative interface for managing OBP APIs and consumers

Key Features:

- Consumer (App) management and configuration
- API metrics viewing and analysis
- User entitlement grant/revoke functionality
- Resource management (branches, etc.)
- Consumer enable/disable control
- OAuth 1.0a authentication against OBP API
- Web UI for administrative tasks

Technology:

- Framework: Django 3.x/4.x
- Language: Python 3.x
- Database: SQLite (development) / PostgreSQL (production)
- WSGI Server: Gunicorn
- Process Control: systemd / supervisor
- Web Server: Nginx / Apache (reverse proxy)

Configuration (`local_settings.py`):

```python
import os

BASE_DIR = '/path/to/project'

# Django settings
SECRET_KEY = '<random-string>'
DEBUG = False  # Set to True for development
ALLOWED_HOSTS = ['127.0.0.1', 'localhost', 'apimanager.yourdomain.com']

# OBP API Configuration
API_HOST = 'http://127.0.0.1:8080'
API_PORTAL = 'http://127.0.0.1:8080'  # If split deployment

# OAuth credentials for the API Manager app
OAUTH_CONSUMER_KEY = '<your-consumer-key>'
OAUTH_CONSUMER_SECRET = '<your-consumer-secret>'
```

```
# Database
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, '..', '..', 'db.sqlite3'),
    }
}

# Optional: Explicit callback URL
# CALLBACK_BASE_URL = "https://apimanager.example.com"

# Static files
STATIC_ROOT = os.path.join(BASE_DIR, '..', '..', 'static-collected')

# Email (for production)
ADMINS = [('Admin', 'admin@example.com')]
SERVER_EMAIL = 'apimanager@example.com'
EMAIL_HOST = 'mail.example.com'
EMAIL_TLS = True

# Filtering
EXCLUDE_APPS = []
EXCLUDE_FUNCTIONS = []
EXCLUDE_URL_PATTERN = []
API_EXPLORER_APP_NAME = 'API Explorer'

# Date formats
API_DATE_FORMAT_WITH_SECONDS = '%Y-%m-%dT%H:%M:%SZ'
API_DATE_FORMAT_WITH_MILLISECONDS = '%Y-%m-%dT%H:%M:%S.000Z'
```

Installation (Development):

```
# Create project structure
mkdir OpenBankProject && cd OpenBankProject
git clone https://github.com/OpenBankProject/API-Manager.git
cd API-Manager

# Create virtual environment
virtualenv --python=python3 ../venv
source ../venv/bin/activate

# Install dependencies
pip install -r requirements.txt

# Create local settings
cp apimanager/apimanager/local_settings.py.example \
   apimanager/apimanager/local_settings.py

# Edit local_settings.py with your configuration
nano apimanager/apimanager/local_settings.py
```

```
# Initialize database
./apimanager/manage.py migrate

# Run development server
./apimanager/manage.py runserver
# Access at http://localhost:8000
```

Installation (Production):

```
# After development setup, collect static files
./apimanager/manage.py collectstatic

# Run with Gunicorn
cd apimanager
gunicorn --config ../gunicorn.conf.py apimanager.wsgi

# Configure systemd service
sudo cp apimanager.service /etc/systemd/system/
sudo systemctl daemon-reload
sudo systemctl enable apimanager
sudo systemctl start apimanager

# Configure Nginx
sudo cp nginx.apimanager.conf /etc/nginx/sites-enabled/
sudo systemctl reload nginx
```

Directory Structure:

```
/OpenBankProject/
├── API-Manager/
│   ├── apimanager/
│   │   ├── apimanager/
│   │   │   ├── __init__.py
│   │   │   ├── settings.py
│   │   │   ├── local_settings.py  # Your config
│   │   │   ├── urls.py
│   │   │   └── wsgi.py
│   │   └── manage.py
│   ├── apimanager.service
│   ├── gunicorn.conf.py
│   ├── nginx.apimanager.conf
│   ├── supervisor.apimanager.conf
│   └── requirements.txt
├── db.sqlite3
├── logs/
├── static-collected/
└── venv/
```

PostgreSQL Configuration:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'apimanager_db',
        'USER': 'apimanager_user',
        'PASSWORD': 'secure_password',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Management:

- Super Admin users can manage roles at `/users`
- Set `super_admin_user_ids` in OBP-API props file (as temporary bootstrap admin user)
- Users need appropriate roles to execute management functions
- Entitlement management requires proper permissions

3.4 Opey II (AI Agent)

Purpose: Conversational AI assistant for banking operations

Key Features:

- Natural language OBP API queries
- Approve / Deny with tracking of Endpoint OperationId per session / once
- Tool Call inspection
- Multi-LLM support (Anthropic, OpenAI, Ollama)
- OBP Resource Docs and Glossary in vector database.
- Vector-based knowledge retrieval
- LangSmith tracing integration
- Consent-based access control

Architecture:

- Agent Framework: LangGraph (stateful workflows)
- LLM Integration: LangChain
- Vector Database: Qdrant
- Web Service: FastAPI
- API: FastAPI-based service
- Frontend: OBP Portal

Supported LLM Providers:

- Anthropic (Claude)
- OpenAI (GPT-4)
- Ollama (Local models - Llama, Mistral)

Configuration .env file (excerpt, see .env-example):

```
# .env file
# LLM Configuration
MODEL_PROVIDER=anthropic
MODEL_NAME=claude-sonnet-4
ANTHROPIC_API_KEY=your-api-key

# OBP Configuration
OBP_BASE_URL=http://127.0.0.1:8080
OBP_USERNAME=your-username
OBP_PASSWORD=your-password
OBP_CONSUMER_KEY=your-consumer-key

# Vector Database
QDRANT_HOST=localhost
QDRANT_PORT=6333

# Tracing (Optional)
LANGCHAIN_TRACING_V2=true
LANGCHAIN_API_KEY=lsv2_pt_xxx
LANGCHAIN_PROJECT=opey-agent
```

Installation (local/development):

```
cd OPEY/OBP-Opey-II
poetry install
poetry shell

# Create vector database
mkdir src/data
python src/scripts/populate_vector_db.py

# Run API service
python src/run_service.py  # Backend API (port 8000)

# Access via OBP Portal frontend
```

OBP-API Configuration for Opey:

```
# In OBP-API props file
skip_consent_sca_for_consumer_id_pairs=[{ \
    "grantor_consumer_id": "<api-explorer-consumer-id, or portal-consumer-id>",\
    "grantee_consumer_id": "<opey-consumer-id>" \
}]
```

Logging Features:

- Automatic username extraction from JWT tokens
- Function-level log identification
- Request/response tracking
- JWT field priority: email → name → preferred_username → sub → user_id

3.5 OBP-OIDC (Development Provider)

Purpose: Lightweight OIDC provider for development and testing

Key Features:

- Full OpenID Connect support
- JWT token generation
- JWKS endpoint
- Discovery endpoint (.well-known)
- User authentication simulation
- Development-friendly configuration

Configuration (excerpt):

```
# In OBP-API props
oauth2.oidc_provider=obp-oidc
oauth2.obp_oidc.host=http://localhost:9000
oauth2.obp_oidc.issuer=http://localhost:9000/obp-oidc
oauth2.obp_oidc.well_known=http://localhost:9000/obp-oidc/.well-known/openid-configuration
oauth2.jwk_set.url=http://localhost:9000/obp-oidc/jwks

# OpenID Connect Client
openid_connect_1.button_text=OBP-OIDC
openid_connect_1.client_id=obp-api-client
openid_connect_1.client_secret=your-secret
openid_connect_1.callback_url=http://localhost:8080/auth/openid-connect/callback
openid_connect_1.endpoint.discovery=http://localhost:9000/obp-oidc/.well-known/openid-configuration
openid_connect_1.endpoint.authorization=http://localhost:9000/obp-oidc/auth
openid_connect_1.endpoint.userinfo=http://localhost:9000/obp-oidc/userinfo
openid_connect_1.endpoint.token=http://localhost:9000/obp-oidc/token
openid_connect_1.endpoint.jwks_uri=http://localhost:9000/obp-oidc/jwks
openid_connect_1.access_type_offline=true
```

3.6 Keycloak Integration (Production Provider)

Purpose: Enterprise-grade OIDC provider for production deployments

Key Features:

- Full OIDC/OAuth2 compliance
- User federation
- Multi-realm support

- Social login integration
- Advanced authentication flows
- User management UI

Configuration:

```
# In OBP-API props
oauth2.oidc_provider=keycloak
oauth2.keycloak.host=http://localhost:7070
oauth2.keycloak.realm=master
oauth2.keycloak.issuer=http://localhost:7070/realms/master
oauth2.jwk_set.url=http://localhost:7070/realms/master/protocol/openid-connect/certs

# OpenID Connect Client
openid_connect_1.button_text=Keycloak
openid_connect_1.client_id=obp-client
openid_connect_1.client_secret=your-secret
openid_connect_1.callback_url=http://localhost:8080/auth/openid-connect/callback
openid_connect_1.endpoint.discovery=http://localhost:7070/realms/master/.well-known/openid-configuration
```

Pre-built Keycloak image with OBP Keycloak provider:

```
docker pull openbankproject/obp-keycloak:main-themed
```

3.7 Ory Hydra (Production Provider)

Purpose: Cloud-native OAuth2 and OpenID Connect server for production deployments

Overview:

Ory Hydra is a hardened, open-source OAuth 2.0 and OpenID Connect server optimized for low-latency, high-throughput, and low resource consumption. It integrates with OBP-API to provide enterprise-grade authentication and authorization.

Key Features:

- **OAuth2 & OIDC Compliance:** Full implementation of OAuth 2.0 and OpenID Connect specifications
- **Cloud Native:** Designed for containerized deployments (Docker, Kubernetes)
- **Performance:** Low latency and high throughput
- **Separation of Concerns:** Hydra handles OAuth/OIDC flow; identity management delegated to custom Identity Provider
- **Security Hardened:** Regular security audits and compliance certifications
- **Storage Backend:** PostgreSQL, MySQL, CockroachDB support

Architecture:

```
Client → Hydra (OAuth2 Server) → OBP Hydra Identity Provider → OBP-API
                ↓
           Database (PostgreSQL)
```

Components:

- **Ory Hydra:** OAuth2/OIDC server
- **OBP Hydra Identity Provider:** Custom login/consent UI and user management
- **OBP-API:** Banking API with Hydra integration

OBP-API Configuration:

```
# Enable Hydra login
login_with_hydra=true

# Hydra server URLs
hydra_public_url=http://127.0.0.1:4444
hydra_admin_url=http://127.0.0.1:4445

# Consent scopes
hydra_consents=ReadAccountsBasic,ReadAccountsDetail,ReadBalances,ReadTransactionsBasic,ReadTransactionsDebits,ReadTransactionsDetail

# JWKS validation
oauth2.jwk_set.url=http://127.0.0.1:4444/.well-known/jwks.json

# Mirror consumers to Hydra clients
mirror_consumer_in_hydra=true
```

Hydra Identity Provider Configuration:

```
# Server port
server.port=8086

# OBP-API URL
obp.base_url=http://localhost:8080
endpoint.path.prefix=${obp.base_url}/obp/v4.0.0

# Hydra admin URL
oauth2.admin_url=http://127.0.0.1:4445

# Service account credentials
identity_provider.user.username=serviceuser
identity_provider.user.password=password
consumer_key=your-consumer-key

# mTLS configuration (optional)
mtls.keyStore.path=file:///path/to/keystore.jks
mtls.keyStore.password=keystore-password
mtls.trustStore.path=file:///path/to/truststore.jks
mtls.trustStore.password=truststore-password
```

Docker Deployment:

```
# Start Hydra with docker-compose
docker-compose -f quickstart.yml \
    -f quickstart-postgres.yml \
    up --build

# Verify Hydra is running
curl http://127.0.0.1:4444/.well-known/openid-configuration
```

Hydra quickstart.yml environment:

```
environment:
  - URLS_CONSENT=http://localhost:8086/consent
  - URLS_LOGIN=http://localhost:8086/login
  - URLS_LOGOUT=http://localhost:8086/logout
```

Use Cases:

- High-performance OAuth2/OIDC deployments
- Microservices architectures requiring centralized authentication
- Multi-tenant banking platforms
- Open Banking TPP integrations
- Cloud-native banking solutions

Repositories:

- Ory Hydra: https://github.com/ory/hydra
- OBP Hydra Identity Provider: https://github.com/OpenBankProject/OBP-Hydra-Identity-Provider
- Demo OAuth2 Client: https://github.com/OpenBankProject/OBP-Hydra-OAuth2

3.8 OBP-Hola

Purpose: Reference implementation for OAuth2 authentication and consent flow testing

Overview:

OBP-Hola is a Java/Spring Boot application that demonstrates and tests OBP authentication, consent creation, and data access via OBP API. It serves as a reference implementation for developers integrating with OBP's consent framework.

Key Features:

- **OAuth2 Flow Testing:** Complete OAuth2 authorization code flow implementation
- **Multi-Standard Support:** UK Open Banking, Berlin Group, and OBP styles
- **Consent Management:** Create, view, and test consent flows
- **mTLS Support:** Certificate-based authentication testing
- **JWS Signing:** Request signing for enhanced security profiles
- **Interactive UI:** Web interface for testing consent scenarios

Supported Standards:

- UK Open Banking (Account Information & Payment Initiation)
- Berlin Group NextGenPSD2 (AIS/PIS)
- OBP native consent flows

Dependencies:

- **OBP-API:** Core banking API
- **Ory Hydra:** OAuth2 server
- **OBP Hydra Identity Provider:** Identity management

Use Cases:

- Testing consent creation and authorization flows
- Validating OAuth2 integration
- Demonstrating PSD2 compliance workflows
- Training and reference for TPP developers
- Automated testing with OBP-Selenium integration

Configuration:

```
# OAuth2 Server
oauth2.public_url=https://oauth2.example.com

# OBP API
obp.base_url=https://api.example.com

# Client credentials (from OBP consumer registration)
oauth2.client_id=your-client-id
oauth2.redirect_uri=http://localhost:8087/callback
oauth2.client_scope=ReadAccountsDetail ReadBalances ReadTransactionsDetail

# mTLS (if required)
mtls.keyStore=/path/to/keystore.jks
mtls.trustStore=/path/to/truststore.jks
```

Running Hola:

```
# Build with Maven
mvn clean package
```

```
# Run locally
java -jar target/obp-hola-app-0.0.29-SNAPSHOT.jar

# Access at http://localhost:8087
```

Docker Deployment:

```
docker build -t obp-hola .
docker run -p 8087:8087 \
  -e OAUTH2_PUBLIC_URL=https://oauth2.example.com \
  -e OBP_BASE_URL=https://api.example.com \
  obp-hola
```

Repository: https://github.com/OpenBankProject/OBP-Hola

3.9 OBP-SEPA-Adapter

Purpose: Reference implementation for SEPA payment processing with OBP-API

Overview:

OBP-SEPA-Adapter is a Scala/Akka-based adapter that enables SEPA (Single Euro Payments Area) payment processing through OBP-API. It handles incoming and outgoing SEPA messages, storing transactions and transaction requests via the OBP-API.

Key Features:

- **SEPA Credit Transfer:** Full support for pacs.008.001.02 messages
- **Payment Returns:** Handle payment return messages (pacs.004.001.02)
- **Payment Rejections:** Process rejection messages (pacs.002.001.03)
- **Payment Recalls:** Support for recall messages (camt.056.001.01)
- **File Processing:** Generate and process SEPA XML files
- **PostgreSQL Storage:** Dedicated database for SEPA message management
- **Akka Connector Integration:** Communicates with OBP-API via Akka remote

Supported SEPA Messages:

Integrated with OBP-API:

- Credit Transfer (pacs.008.001.02)
- Payment Return (pacs.004.001.02)
- Payment Reject (pacs.002.001.03)
- Payment Recall (camt.056.001.01)
- Payment Recall Negative Answer (camt.029.001.03)

Supported but not integrated:

- Inquiry Claim messages (camt.027, camt.087, camt.029)

- Request Status Update (pacs.028.001.01)

Architecture:

```
OBP-API (Star Connector) → Akka Remote → SEPA Adapter → PostgreSQL
                                              ↓
                                         SEPA Files (in/out)
```

Configuration:

OBP-API props:

```
connector=star
starConnector_supported_types=mapped,akka
transactionRequests_supported_types=SANDBOX_TAN,COUNTERPARTY,SEPA,ACCOUNT_OTP,ACCOUNT,REFUND
akka_connector.hostname=127.0.0.1
akka_connector.port=2662
```

SEPA Adapter application.conf:

```
databaseConfig = {
  dataSourceClass = "org.postgresql.ds.PGSimpleDataSource"
  properties = {
    databaseName = "sepa_db"
    user = "sepa_user"
    password = "password"
  }
}

obp-api = {
  authorization = {
    direct-login-token = "YOUR_DIRECTLOGIN_TOKEN"
  }
}

sepa-adapter {
  bank-id = "THE_DEFAULT_BANK_ID"
  bank-bic = "OBPBDEB1XXX"
}
```

Method Routing Setup:

Create method routings to route payment methods through Akka connector:

```
{
  "is_bank_id_exact_match": false,
  "method_name": "makePaymentv210",
  "connector_name": "akka_vDec2018",
  "bank_id_pattern": ".*",
  "parameters": []
}
```

Repeat for: `makePaymentV400`, `notifyTransactionRequest`

Required Entitlements:

- CanCreateHistoricalTransaction
- CanCreateAnyTransactionRequest

Use Cases:

- SEPA credit transfer payment processing
- Payment returns and rejections handling
- Payment recall workflows
- SEPA file generation and processing
- Euro zone payment integration

Technology Stack:

- Language: Scala
- Framework: Akka
- Database: PostgreSQL with Slick ORM
- Message Format: SEPA XML (ISO 20022 standard)
- Code Generation: scalaxb for XSD classes

ISO 20022 Compliance:

The SEPA Adapter implements ISO 20022 message standards for financial messaging:

- **pacs.008.001.02** - FIToFICustomerCreditTransfer (Credit Transfer)
- **pacs.004.001.02** - PaymentReturn (Payment Return)
- **pacs.002.001.03** - FIToFIPaymentStatusReport (Payment Reject)
- **pacs.028.001.01** - FIToFIPaymentStatusRequest (Request Status Update)
- **camt.056.001.01** - FIToFIPaymentCancellationRequest (Payment Recall)
- **camt.029.001.03** - ResolutionOfInvestigation (Recall Negative Answer)
- **camt.027.001.06** - ClaimNonReceipt (Inquiry Claim Non Receipt)
- **camt.087.001.05** - RequestToModifyPayment (Inquiry Claim Value Date Correction)
- **camt.029.001.08** - ResolutionOfInvestigation (Inquiry Responses)

ISO 20022 provides standardized XML schemas for electronic data interchange between financial institutions, ensuring interoperability across the SEPA payment network.

Running the Adapter:

```
# Setup database
psql -f src/main/scala/model/DatabaseSchema.sql

# Configure application.conf
# Edit src/main/resources/application.conf

# Run the adapter
sbt run
```

Processing Files:

```
# Generate outgoing SEPA files
sbt "runMain sepa.scheduler.ProcessOutgoingFiles"
# Files created in: src/main/scala/sepa/sct/file/out

# Process incoming SEPA files
sbt "runMain sepa.scheduler.ProcessIncomingFilesActorSystem"
```

Repository: https://github.com/OpenBankProject/OBP-SEPA-Adapter

---

3.10 Connectors

Purpose: Connectors provide the integration layer between OBP-API and backend banking systems or data sources.

Available Connectors:

Mapped (Internal)

- Direct database connector for sandbox/development
- Stores data in OBP's own database tables
- No external system required
- Configuration: `connector=mapped`

Kafka

- Message-based connector using Apache Kafka
- Asynchronous communication with backend systems
- Supports high-throughput scenarios
- Configuration: `connector=kafka_vMar2017`

RabbitMQ

- Message queue-based connector
- Alternative to Kafka for message-based integration
- Supports reliable message delivery
- Configuration: Configure via props with RabbitMQ connection details

Akka Remote

- Actor-based remote connector
- Separates API layer from data layer
- Enables distributed deployments
- Configuration: `connector=akka_vDec2018`

Cardano

- Blockchain connector for Cardano network
- Enables blockchain-based banking operations
- Supports smart contract integration
- Configuration: `connector=cardano`

Ethereum

- Blockchain connector for Ethereum network
- Smart contract integration for DeFi applications
- Web3 compatibility
- Configuration: `connector=ethereum`

REST/Stored Procedure

- Direct REST API or stored procedure connectors
- Custom integration with existing systems
- Flexible adapter pattern

Custom Connectors:

- Create custom connectors by extending the `Connector` trait
- See section 11.3 for implementation details

3.11 Adapters

Purpose: Adapters are backend services that receive messages from OBP-API connectors and respond according to Message Doc definitions.

Overview:

Adapters act as the bridge between OBP-API and core banking systems:

- **Receive:** Accept messages from OBP-API via message queues (Kafka/RabbitMQ) or remote calls (Akka)
- **Process:** Interact with core banking systems, databases, or other backend services
- **Respond:** Return data formatted according to Message Doc specifications

Architecture:

```
OBP-API (Connector) → Message Queue → Adapter → Core Banking System
                    ←              ←        ←
```

Key Characteristics:

- **Language Agnostic:** Adapters can be written in any programming language
- **Message Doc Compliance:** Must implement request/response formats defined in Message Docs
- **Scalability:** Multiple adapter instances can process messages concurrently
- **Flexibility:** Different adapters can serve different banking systems or functions

Implementation:

Adapters listen to message queues or remote calls, parse incoming messages according to Message Doc schemas, execute business logic, and return responses in the required format.

Example Use Cases:

- Adapter in Java connecting to legacy mainframe systems
- Adapter in Python integrating with modern REST APIs
- Adapter in Go for high-performance transaction processing
- Adapter in Scala for Akka-based distributed systems

3.12 Message Docs

Purpose: Message Docs define the structure and schema of messages exchanged between OBP-API connectors and backend adapters.

Overview:

Message Docs serve as API contracts for connector-adapter communication, specifying:

- Request message format and required fields
- Response message format and data structure
- Field types and validation rules
- Example messages for testing

Key Features:

- **Dynamic Definition:** Message Docs can be created dynamically via API without code changes
- **Version Control:** Different connector versions can have different message formats
- **Documentation:** Auto-generated documentation for adapter developers
- **Validation:** Ensures message compatibility between connectors and adapters

Available Message Docs:

Message Docs are available for various connectors including Kafka, RabbitMQ, and Akka. Each connector version has its own set of message definitions.

Example: RabbitMQ Message Docs

Configuration:

```
# Enable message doc endpoints
connector=rabbitmq_vOct2024
```

Related Roles:

- CanCreateDynamicMessageDoc
- CanGetDynamicMessageDoc
- CanGetAllDynamicMessageDocs
- CanUpdateDynamicMessageDoc
- CanDeleteDynamicMessageDoc

---

4. Standards Compliance

4.1 Berlin Group NextGenPSD2

Overview: European PSD2 XS2A standard for payment services

Supported Features:

- Account Information Service (AIS)
- Payment Initiation Service (PIS)
- Confirmation of Funds (CoF)
- Strong Customer Authentication (SCA)
- Consent management

API Version Support:

- Berlin Group 1.3
- STET 1.4

Key Endpoints:

```
POST /v1/consents
GET /v1/accounts
GET /v1/accounts/{account-id}/transactions
POST /v1/payments/sepa-credit-transfers
GET /v1/funds-confirmations
```

Implementation Notes:

- Consent-based access model
- OAuth2/OIDC for authentication
- TPP certificate validation
- Transaction signing support

4.2 UK Open Banking

Overview: UK's Open Banking standard (Version 3.1)

Supported Features:

- Account and Transaction API
- Payment Initiation API
- Confirmation of Funds API
- Event Notification API
- Variable Recurring Payments (VRP)

API Version: UK 3.1

Security Profile:

- FAPI compliance
- OBIE Directory integration
- Qualified certificates (eIDAS)
- MTLS support

Key Endpoints:

```
GET /open-banking/v3.1/aisp/accounts
GET /open-banking/v3.1/aisp/transactions
POST /open-banking/v3.1/pisp/domestic-payments
POST /open-banking/v3.1/cbpii/funds-confirmation-consents
```

4.3 Open Bank Project Standard

Overview: OBP's native API standard with extensive banking operations

Current Version: v6.0.0

Key Features:

- 600+ endpoints
- Multi-bank support
- Extended customer data
- Consent management
- Product management
- Webhook support
- Dynamic entity/endpoint creation

Versioning:

- v1.2.1, v1.3.0, v1.4.0 (Legacy, STABLE)
- v2.0.0, v2.1.0, v2.2.0 (STABLE)
- v3.0.0, v3.1.0 (STABLE)
- v4.0.0 (STABLE)
- v5.0.0, v5.1.0 (STABLE)
- v6.0.0 (BLEEDING-EDGE)

Key Endpoint Categories:

- Account Management
- Transaction Operations
- Customer Management
- Consent Management
- Product & Card Management
- KYC/AML Operations
- Webhook Management
- Dynamic Resources

4.4 Other Supported Standards

Polish API 2.1.1.1:

- Polish Banking API standard
- Local market adaptations

AU CDR v1.0.0:

- Australian Consumer Data Right
- Banking sector implementation

BAHRAIN OBF 1.0.0:

- Bahrain Open Banking Framework
- Central Bank of Bahrain standard

CNBV v1.0.0:

- Mexican banking standard

Regulatory Compliance:

- GDPR (EU data protection)
- PSD2 (EU payment services)
- FAPI (Financial-grade API security)
- eIDAS (Electronic identification)

---

5. Installation and Configuration

5.1 Prerequisites

Software Requirements:

- Java: OpenJDK 11+ or Oracle JDK 1.8/13
- Maven: 3.6+
- Node.js: 18+ (for frontend components)
- PostgreSQL: 12+ (production)
- Redis: 6+ (for rate limiting and sessions)
- Docker: 20+ (optional, for containerized deployment)

Hardware Requirements (Minimum):

- CPU: 2 cores
- RAM: 8GB
- Disk: 50GB
- Network: 100 Mbps

Hardware Requirements (Production):

- CPU: 8+ cores
- RAM: 16GB+
- Disk: 200GB+ SSD
- Network: 1 Gbps

5.2 OBP-API Installation

5.2.1 Installing JDK

Using sdkman (Recommended):

```
curl -s "https://get.sdkman.io" | bash
source "$HOME/.sdkman/bin/sdkman-init.sh"
sdk env install  # Uses .sdkmanrc in project
```

Manual Installation:

```
# Ubuntu/Debian
sudo apt update
sudo apt install openjdk-11-jdk

# Verify
java -version
```

5.2.2 Clone and Build

```
# Clone repository
git clone https://github.com/OpenBankProject/OBP-API.git
cd OBP-API

# Create configuration
mkdir -p obp-api/src/main/resources/props
cp obp-api/src/main/resources/props/sample.props.template \
   obp-api/src/main/resources/props/default.props

# Edit configuration
nano obp-api/src/main/resources/props/default.props
```

```
# Build and run
mvn install -pl .,obp-commons && mvn jetty:run -pl obp-api
```

Alternative with increased stack size:

```
export MAVEN_OPTS="-Xss128m"
mvn install -pl .,obp-commons && mvn jetty:run -pl obp-api
```

For Java 11+ (if needed):

```
mkdir -p .mvn
cat > .mvn/jvm.config << 'EOF'
--add-opens java.base/java.lang=ALL-UNNAMED
--add-opens java.base/java.lang.reflect=ALL-UNNAMED
--add-opens java.base/java.security=ALL-UNNAMED
--add-opens java.base/java.util.jar=ALL-UNNAMED
--add-opens java.base/sun.nio.ch=ALL-UNNAMED
--add-opens java.base/java.nio=ALL-UNNAMED
--add-opens java.base/java.net=ALL-UNNAMED
--add-opens java.base/java.io=ALL-UNNAMED
EOF
```

5.2.3 Database Setup

PostgreSQL Installation:

```
# Ubuntu/Debian
sudo apt install postgresql postgresql-contrib

# macOS
brew install postgresql
brew services start postgresql
```

Database Configuration:

```
-- Connect to PostgreSQL
psql postgres

-- Create database
CREATE DATABASE obpdb;

-- Create user
CREATE USER obp WITH PASSWORD 'your-secure-password';

-- Grant privileges
```

```
GRANT ALL PRIVILEGES ON DATABASE obpdb TO obp;

-- For PostgreSQL 16+
\c obpdb;
GRANT USAGE ON SCHEMA public TO obp;
GRANT CREATE ON SCHEMA public TO obp;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO obp;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO obp;
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL ON TABLES TO obp;
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL ON SEQUENCES TO obp;
```

Props Configuration:

```
db.driver=org.postgresql.Driver
db.url=jdbc:postgresql://localhost:5432/obpdb?user=obp&password=your-secure-password
```

PostgreSQL with SSL:

```
db.url=jdbc:postgresql://localhost:5432/obpdb?user=obp&password=xxx&ssl=true

# In postgresql.conf
ssl = on
ssl_cert_file = '/etc/ssl/certs/server.crt'
ssl_key_file = '/etc/ssl/private/server.key'

# In pg_hba.conf
hostssl all all 0.0.0.0/0 md5
```

H2 Database (Development):

```
db.driver=org.h2.Driver
db.url=jdbc:h2:./obp_api.db;DB_CLOSE_ON_EXIT=FALSE
```

5.2.4 Redis Setup

```
# Ubuntu/Debian
sudo apt install redis-server
sudo systemctl start redis-server
sudo systemctl enable redis-server

# macOS
brew install redis
brew services start redis
```

```
# Verify
redis-cli ping  # Should return PONG
```

Props Configuration:

```
use_consumer_limits=true
cache.redis.url=127.0.0.1
cache.redis.port=6379
```

5.3 Production Deployment

5.3.1 Jetty 9 Configuration

Install Jetty:

```
sudo apt install jetty9
```

Configure Jetty (`/etc/default/jetty9`):

```
NO_START=0
JETTY_HOST=127.0.0.1  # Change to 0.0.0.0 for external access
JAVA_OPTIONS="-Drun.mode=production \
  -XX:PermSize=256M \
  -XX:MaxPermSize=512M \
  -Xmx768m \
  -verbose \
  -Dobp.resource.dir=$JETTY_HOME/resources \
  -Dprops.resource.dir=$JETTY_HOME/resources"
```

Build WAR file:

```
mvn package
# Output: target/OBP-API-1.0.war
```

Deploy:

```
sudo cp target/OBP-API-1.0.war /usr/share/jetty9/webapps/root.war
sudo chown jetty:jetty /usr/share/jetty9/webapps/root.war

# Edit /etc/jetty9/jetty.conf - comment out:
# etc/jetty-logging.xml
# etc/jetty-started.xml
```

```
sudo systemctl restart jetty9
```

5.3.2 Production Props Configuration

Create `production.default.props`:

```
# Server Mode
server_mode=apis
run.mode=production

# Database
db.driver=org.postgresql.Driver
db.url=jdbc:postgresql://db-server:5432/obpdb?user=obp&password=xxx&ssl=true

# Connector
connector=mapped

# Redis
cache.redis.url=redis-server
cache.redis.port=6379

# Rate Limiting
use_consumer_limits=true
user_consumer_limit_anonymous_access=100

# OAuth2/OIDC
allow_oauth2_login=true
oauth2.jwk_set.url=https://keycloak.yourdomain.com/realms/obp/protocol/openid-connect/certs

# Security
webui_override_style_sheet=/path/to/custom.css

# Admin (use temporarily for bootstrap only)
# super_admin_user_ids=bootstrap-admin-uuid
```

5.3.3 SSL/HTTPS Configuration

Enable secure cookies (`webapp/WEB-INF/web.xml`):

```
<session-config>
  <cookie-config>
    <secure>true</secure>
    <http-only>true</http-only>
  </cookie-config>
</session-config>
```

Nginx Reverse Proxy:

```
server {
    listen 443 ssl http2;
    server_name api.yourdomain.com;

    ssl_certificate /etc/ssl/certs/yourdomain.crt;
    ssl_certificate_key /etc/ssl/private/yourdomain.key;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;

    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_buffering off;
    }
}
```

5.3.4 Docker Deployment

OBP-API Docker:

```
# Pull image
docker pull openbankproject/obp-api

# Run with environment variables
docker run -d \
  --name obp-api \
  -p 8080:8080 \
  -e OBP_DB_DRIVER=org.postgresql.Driver \
  -e OBP_DB_URL=jdbc:postgresql://postgres:5432/obpdb \
  -e OBP_CONNECTOR=mapped \
  -e OBP_CACHE_REDIS_URL=redis \
  openbankproject/obp-api
```

Docker Compose:

```
version: "3.8"

services:
  obp-api:
    image: openbankproject/obp-api
    ports:
      - "8080:8080"
    environment:
      - OBP_DB_DRIVER=org.postgresql.Driver
      - OBP_DB_URL=jdbc:postgresql://postgres:5432/obpdb
      - OBP_CONNECTOR=mapped
      - OBP_CACHE_REDIS_URL=redis
    depends_on:
      - postgres
      - redis
    networks:
      - obp-network

  postgres:
    image: postgres:13
    environment:
      - POSTGRES_DB=obpdb
      - POSTGRES_USER=obp
      - POSTGRES_PASSWORD=obp_password
    volumes:
      - postgres-data:/var/lib/postgresql/data
    networks:
      - obp-network

  redis:
    image: redis:6-alpine
    networks:
      - obp-network

volumes:
  postgres-data:

networks:
  obp-network:
```

6. Authentication and Security

6.1 Authentication Methods

OBP-API supports multiple authentication methods to accommodate different use cases and integration scenarios.

6.1.1 OAuth 1.0a

Overview: Traditional three-legged OAuth flow for third-party applications

Use Cases:

- Legacy integrations
- Apps requiring delegated access without OpenID Connect support

Flow:

1. Consumer obtains request token

2. User redirected to OBP for authorization
3. User approves access
4. Consumer exchanges request token for access token
5. Access token used for API calls

Implementation:

```
# Get request token
POST /oauth/initiate
Authorization: OAuth oauth_consumer_key="xxx", oauth_signature_method="HMAC-SHA256"

# User authorization
GET /oauth/authorize?oauth_token=REQUEST_TOKEN

# Get access token
POST /oauth/token
Authorization: OAuth oauth_token="REQUEST_TOKEN", oauth_verifier="VERIFIER"

# API call with access token
GET /obp/v5.1.0/banks
Authorization: OAuth oauth_token="ACCESS_TOKEN", oauth_signature="..."
```

6.1.2 OAuth 2.0

Overview: Modern authorization framework supporting various grant types

Supported Grant Types:

- Authorization Code (recommended for web apps)
- Client Credentials (for server-to-server)
- Implicit (deprecated, not recommended)

Configuration:

```
allow_oauth2_login=true
oauth2.jwk_set.url=https://idp.example.com/jwks
```

Authorization Code Flow:

```
# 1. Authorization request
GET /oauth/authorize?
  response_type=code&
  client_id=CLIENT_ID&
  redirect_uri=CALLBACK_URL&
  scope=openid profile&
  state=RANDOM_STATE
```

```
# 2. Token exchange
POST /oauth/token
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=AUTH_CODE&
redirect_uri=CALLBACK_URL&
client_id=CLIENT_ID&
client_secret=CLIENT_SECRET

# 3. API call with bearer token
GET /obp/v5.1.0/users/current
Authorization: Bearer ACCESS_TOKEN
```

6.1.3 OpenID Connect (OIDC)

Overview: Identity layer on top of OAuth 2.0 providing user authentication

Providers:

- **Production:** Keycloak, Hydra, Google, Yahoo, Auth0, Azure AD
- **Development:** OBP-OIDC

Configuration Example (Keycloak):

```
# OpenID Connect Configuration
openid_connect_1.button_text=Keycloak Login
openid_connect_1.client_id=obp-client
openid_connect_1.client_secret=your-secret
openid_connect_1.callback_url=http://localhost:8080/auth/openid-connect/callback
openid_connect_1.endpoint.discovery=http://keycloak:7070/realms/obp/.well-known/openid-configuration
openid_connect_1.endpoint.authorization=http://keycloak:7070/realms/obp/protocol/openid-connect/auth
openid_connect_1.endpoint.userinfo=http://keycloak:7070/realms/obp/protocol/openid-connect/userinfo
openid_connect_1.endpoint.token=http://keycloak:7070/realms/obp/protocol/openid-connect/token
openid_connect_1.endpoint.jwks_uri=http://keycloak:7070/realms/obp/protocol/openid-connect/certs
openid_connect_1.access_type_offline=true
```

Multiple OIDC Providers:

```
# Provider 1 - Google
openid_connect_1.button_text=Google
openid_connect_1.client_id=google-client-id
openid_connect_1.client_secret=google-secret
```

```
openid_connect_1.endpoint.discovery=https://accounts.google.com/.well-known/openid-configuration
openid_connect_1.access_type_offline=false

# Provider 2 - Azure AD
openid_connect_2.button_text=Microsoft
openid_connect_2.client_id=azure-client-id
openid_connect_2.client_secret=azure-secret
openid_connect_2.endpoint.discovery=https://login.microsoftonline.com/common/v2.0/.well-known/openid-configuration
openid_connect_2.access_type_offline=true
```

JWT Token Validation:

```
oauth2.jwk_set.url=http://keycloak:7070/realms/obp/protocol/openid-connect/certs
```

6.1.4 Direct Login

Overview: Simplified username/password authentication for trusted applications

Use Cases:

- Internal applications
- Testing and development
- Mobile apps with secure credential storage

Implementation:

```
# Direct Login
POST /obp/v5.1.0/my/logins/direct
Content-Type: application/json
DirectLogin: username=user@example.com,
            password=secret,
            consumer_key=CONSUMER_KEY

# Response includes token
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}

# Subsequent API calls
GET /obp/v5.1.0/users/current
Authorization: DirectLogin token="TOKEN"
```

Security Considerations:

- Only use over HTTPS
- Implement rate limiting

- Use strong passwords
- Token expiration and refresh

6.2 JWT Token Structure

Standard Claims:

```
{
  "iss": "http://keycloak:7070/realms/obp",
  "sub": "user-uuid",
  "aud": "obp-client",
  "exp": 1704067200,
  "iat": 1704063600,
  "email": "user@example.com",
  "name": "John Doe",
  "preferred_username": "johndoe"
}
```

JWT Validation Process:

1. Verify signature using JWKS
2. Check issuer matches configured provider
3. Validate expiration time
4. Verify audience claim
5. Extract user identifier

7. Access Control and Security Mechanisms

7.1 Role-Based Access Control (RBAC)

Overview: OBP uses an entitlement-based system where users are granted specific roles that allow them to perform certain operations.

Core Concepts:

- **Entitlement:** Permission to perform a specific action
- **Role:** Collection of entitlements (used interchangeably)
- **Scope:** Optional constraint on entitlement (bank-level, system-level)

Common Roles:

Role	Description	Scope
`CanCreateAccount`	Create bank accounts	Bank
`CanGetAnyUser`	View any user details	System
`CanCreateEntitlementAtAnyBank`	Grant roles at any bank	System
`CanCreateBranch`	Create branch records	Bank

Role	Description	Scope
`CanReadMetrics`	View API metrics	System
`CanCreateConsumer`	Create OAuth consumers	System

Granting Entitlements:

```
POST /obp/v5.1.0/users/USER_ID/entitlements
Authorization: DirectLogin token="ADMIN_TOKEN"
Content-Type: application/json

{
  "bank_id": "gh.29.uk",
  "role_name": "CanCreateAccount"
}
```

Super Admin Bootstrap:

```
# In props file (temporary, for bootstrap only)
super_admin_user_ids=uuid-1,uuid-2

# After bootstrap, grant CanCreateEntitlementAtAnyBank
# Then remove super_admin_user_ids from props
```

Checking User Entitlements:

```
GET /obp/v5.1.0/users/USER_ID/entitlements
Authorization: DirectLogin token="TOKEN"
```

7.2 Consent Management

Overview: PSD2-compliant consent mechanism for controlled data access

Consent Types:

- Account Information (AIS)
- Payment Initiation (PIS)
- Confirmation of Funds (CoF)
- Variable Recurring Payments (VRP)

Consent Lifecycle:

```
┌─────────────┐     ┌──────────────┐     ┌──────────────┐
│   INITIATED │────▶│   ACCEPTED   │────▶│   REVOKED    │
└─────────────┘     └──────────────┘     └──────────────┘
```

```
       │                    │
       │                    ▼
       │            ┌──────────────┐
       └───────────►│   REJECTED   │
                    └──────────────┘
```

Creating a Consent:

```
POST /obp/v5.1.0/my/consents/IMPLICIT
Authorization: Bearer ACCESS_TOKEN
Content-Type: application/json

{
  "everything": false,
  "account_access": [{
    "account_id": "account-123",
    "view_id": "owner"
  }],
  "valid_from": "2024-01-01T00:00:00Z",
  "time_to_live": 7776000,
  "email": "user@example.com"
}

Note: Replace `IMPLICIT` with `SMS` or `EMAIL` for other SCA methods.
```

Challenge Flow (SCA):

```
# 1. Create consent - returns challenge
POST /obp/v5.1.0/banks/BANK_ID/consents/CONSENT_ID/challenge

# 2. Answer challenge
POST /obp/v5.1.0/banks/BANK_ID/consents/CONSENT_ID/challenge
{
  "answer": "123456"
}
```

Consent for Opey:

```
# Skip SCA for trusted consumer pairs
skip_consent_sca_for_consumer_id_pairs=[{
  "grantor_consumer_id": "api-explorer-id",
  "grantee_consumer_id": "opey-id"
}]
```

7.3 Views System

Overview: Fine-grained control over what data is visible to different actors

Standard Views:

- `owner` - Full account access (account holder)
- `accountant` - Transaction data, no personal info
- `auditor` - Read-only comprehensive access
- `public` - Public information only

Custom Views:

```
POST /obp/v5.1.0/banks/BANK_ID/accounts/ACCOUNT_ID/views
{
  "name": "manager_view",
  "description": "Branch manager view",
  "is_public": false,
  "which_alias_to_use": "private",
  "hide_metadata_if_alias_used": false,
  "allowed_permissions": [
    "can_see_transaction_description",
    "can_see_transaction_amount",
    "can_see_transaction_currency"
  ]
}
```

7.4 Rate Limiting

Overview: Protect API resources from abuse and ensure fair usage

Configuration:

```
# Enable rate limiting
use_consumer_limits=true

# Redis backend
cache.redis.url=127.0.0.1
cache.redis.port=6379

# Anonymous access limit (per minute)
user_consumer_limit_anonymous_access=60
```

Setting Consumer Limits:

```
PUT /obp/v5.1.0/management/consumers/CONSUMER_ID/consumer/call-limits
{
  "per_second_call_limit": "10",
  "per_minute_call_limit": "100",
  "per_hour_call_limit": "1000",
  "per_day_call_limit": "10000",
  "per_week_call_limit": "50000",
  "per_month_call_limit": "200000"
}
```

Rate Limit Headers:

```
HTTP/1.1 429 Too Many Requests
X-Rate-Limit-Limit: 100
X-Rate-Limit-Remaining: 0
X-Rate-Limit-Reset: 45

{
  "error": "OBP-10018: Too Many Requests. We only allow 100 requests per minute
for this Consumer."
}
```

7.5 Security Best Practices

Password Security:

```
# Props encryption using OpenSSL
jwt.use.ssl=true
keystore.path=/path/to/api.keystore.jks
keystore.alias=KEYSTORE_ALIAS

# Encrypted props
db.url.is_encrypted=true
db.url=BASE64_ENCODED_ENCRYPTED_VALUE
```

Transport Security:

- Always use HTTPS in production
- Enable HTTP Strict Transport Security (HSTS)
- Use TLS 1.2 or higher
- Implement certificate pinning for mobile apps

API Security:

- Validate all input parameters
- Implement request signing
- Use CSRF tokens for web forms
- Enable audit logging
- Regular security updates

Jetty Password Obfuscation:

```
# Generate obfuscated password
java -cp /usr/share/jetty9/lib/jetty-util-*.jar \
  org.eclipse.jetty.util.security.Password password123

# Output: OBF:1v2j1uum1xtv1zej1zer1xtn1uvk1v1v

# In props
db.password.is_obfuscated=true
db.password=OBF:1v2j1uum1xtv1zej1zer1xtn1uvk1v1v
```

8. Monitoring, Logging, and Troubleshooting

8.1 Logging Configuration

Logback Configuration (`logback.xml`):

```xml
<configuration>
  <appender name="FILE" class="ch.qos.logback.core.FileAppender">
    <file>logs/obp-api.log</file>
    <encoder>
      <pattern>%date %level [%thread] %logger{10} - %msg%n</pattern>
    </encoder>
  </appender>

  <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
      <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
    </encoder>
  </appender>

  <!-- API logging -->
  <logger name="code.api" level="INFO"/>

  <!-- OAuth/OIDC debugging -->
  <logger name="code.api.OAuth2" level="DEBUG"/>
  <logger name="code.api.util.JwtUtil" level="DEBUG"/>

  <!-- Connector logging -->
  <logger name="code.bankconnectors" level="INFO"/>

  <!-- Database queries -->
  <logger name="net.liftweb.mapper" level="WARN"/>

  <root level="INFO">
    <appender-ref ref="FILE"/>
    <appender-ref ref="CONSOLE"/>
  </root>
</configuration>
```

Component-Specific Logging:

```
<!-- Enable specific components -->
<logger name="code.api.v5_1_0.APIMethods510" level="DEBUG"/>
<logger name="code.bankconnectors.Connector" level="TRACE"/>
<logger name="code.api.util.RateLimiting" level="DEBUG"/>
```

8.2 API Metrics

Metrics Endpoint:

```
GET /obp/v5.1.0/management/metrics
Authorization: DirectLogin token="TOKEN"

# With filters
GET /obp/v5.1.0/management/metrics?
  from_date=2024-01-01T00:00:00Z&
  to_date=2024-01-31T23:59:59Z&
  consumer_id=CONSUMER_ID&
  user_id=USER_ID&
  implemented_by_partial_function=getBank&
  verb=GET
```

Aggregate Metrics:

```
GET /obp/v5.1.0/management/aggregate-metrics
{
  "aggregate_metrics": [{
    "count": 1500,
    "average_response_time": 145.3,
    "minimum_response_time": 23,
    "maximum_response_time": 2340
  }]
}
```

Top APIs:

```
GET /obp/v5.1.0/management/metrics/top-apis
```

Elasticsearch Integration:

```
# Enable ES metrics
es.metrics.enabled=true
es.metrics.url=http://elasticsearch:9200
es.metrics.index=obp-metrics

# Query via API
POST /obp/v5.1.0/search/metrics
```

8.3 Monitoring Endpoints

Health Check:

```
GET /obp/v5.1.0/root
{
  "version": "v5.1.0",
  "version_status": "STABLE",
  "git_commit": "abc123...",
  "connector": "mapped"
}
```

Connector Status:

```
GET /obp/v5.1.0/connector-loopback
{
  "connector_version": "mapped_2024",
  "git_commit": "def456...",
  "duration_time": "10 ms"
}
```

Database Info:

```
GET /obp/v5.1.0/database/info
{
  "name": "PostgreSQL",
  "version": "13.8",
  "git_commit": "...",
  "date": "2024-01-15T10:30:00Z"
}
```

Rate Limiting Status:

```
GET /obp/v5.1.0/rate-limiting
{
  "enabled": true,
  "technology": "REDIS",
  "service_available": true,
  "is_active": true
}
```

8.4 Common Issues and Troubleshooting

8.4.1 Authentication Issues

Problem: OBP-20208: Cannot match the issuer and JWKS URI

Solution:

```
# Ensure issuer matches JWT iss claim
oauth2.jwk_set.url=http://keycloak:7070/realms/obp/protocol/openid-connect/certs

# Check JWT token issuer
curl -X GET http://localhost:8080/obp/v5.1.0/users/current \
  -H "Authorization: Bearer TOKEN" -v

# Enable debug logging
<logger name="code.api.OAuth2" level="DEBUG"/>
```

Problem: OAuth signature mismatch

Solution:

- Verify consumer key/secret
- Check URL encoding
- Ensure timestamp is current
- Verify signature base string construction

8.4.2 Database Connection Issues

Problem: Connection timeout to PostgreSQL

Solution:

```
# Check PostgreSQL is running
sudo systemctl status postgresql

# Test connection
psql -h localhost -U obp -d obpdb

# Check max connections
# In postgresql.conf
max_connections = 200

# Check connection pool in props
db.url=jdbc:postgresql://localhost:5432/obpdb?...&maxPoolSize=50
```

Problem: Database migration needed

Solution:

```
# OBP-API handles migrations automatically on startup
# Check logs for migration status
tail -f logs/obp-api.log | grep -i migration
```

8.4.3 Redis Connection Issues

Problem: Rate limiting not working

Solution:

```
# Check Redis connectivity
redis-cli ping

# Test from OBP-API server
telnet redis-host 6379

# Check props configuration
cache.redis.url=correct-hostname
cache.redis.port=6379

# Verify rate limiting is enabled
use_consumer_limits=true
```

8.4.4 Memory Issues

Problem: OutOfMemoryError

Solution:

```
# Increase JVM memory
export MAVEN_OPTS="-Xmx2048m -Xms1024m -XX:MaxPermSize=512m"

# For production (in jetty config)
JAVA_OPTIONS="-Xmx4096m -Xms2048m"

# Monitor memory usage
jconsole  # Connect to JVM process
```

8.4.5 Performance Issues

Problem: Slow API responses

Diagnosis:

```
# Check metrics for slow endpoints
GET /obp/v5.1.0/management/metrics?
  sort_by=duration&
  limit=100

# Enable connector timing logs
<logger name="code.bankconnectors" level="DEBUG"/>

# Check database query performance
<logger name="net.liftweb.mapper" level="DEBUG"/>
```

Solutions:

- Enable Redis caching
- Optimize database indexes
- Increase connection pool size
- Use Akka remote for distributed setup
- Enable HTTP/2

8.5 Debug Tools

API Call Context:

```
GET /obp/v5.1.0/development/call-context
# Returns current request context for debugging
# Required role: CanGetCallContext
```

Log Cache:

```
GET /obp/v5.1.0/management/logs/INFO
# Retrieves cached log entries
```

Testing Endpoints:

```
# Test delay/timeout handling
GET /obp/v5.1.0/development/waiting-for-godot?sleep=1000

# Test rate limiting
GET /obp/v5.1.0/rate-limiting
```

---

9. API Documentation and Service Guides

9.1 API Explorer Usage

Accessing API Explorer:

```
http://localhost:5173  # Development
https://apiexplorer.yourdomain.com  # Production
```

Key Features:

1. **Browse APIs:** Navigate through 600+ endpoints organized by category
2. **Try APIs:** Execute requests directly from the browser
3. **OAuth Flow:** Built-in OAuth authentication
4. **Collections:** Save and organize frequently-used endpoints
5. **Examples:** View request/response examples
6. **Multi-language:** English and Spanish support

Authentication Flow:

1. Click "Login" button
2. Select OAuth provider (OBP-OIDC, Keycloak, etc.)
3. Authenticate with credentials
4. Grant permissions
5. Redirected back with access token

9.2 API Versioning

Accessing Different Versions:

```
# v5.1.0 (latest)
GET /obp/v5.1.0/banks

# v4.0.0 (stable)
GET /obp/v4.0.0/banks

# Berlin Group
GET /berlin-group/v1.3/accounts
```

Version Status Check:

```
GET /obp/v5.1.0/root
{
  "version": "v5.1.0",
  "version_status": "STABLE"  # or DRAFT, BLEEDING-EDGE
}
```

9.3 API Documentation Formats

Resource Docs (OBP Native Format):

OBP's native documentation format provides comprehensive endpoint information including roles, example bodies, and implementation details.

```
# OBP Standard
GET /obp/v5.1.0/resource-docs/v5.1.0/obp

# Berlin Group
GET /obp/v5.1.0/resource-docs/BGv1.3/obp

# UK Open Banking
GET /obp/v5.1.0/resource-docs/UKv3.1/obp

# Filter by tags
GET /obp/v5.1.0/resource-docs/v5.1.0/obp?tags=Account,Bank

# Filter by functions
GET /obp/v5.1.0/resource-docs/v5.1.0/obp?functions=getBank,getAccounts

# Filter by content type (dynamic/static)
GET /obp/v5.1.0/resource-docs/v5.1.0/obp?content=dynamic
```

Swagger Documentation:

Swagger/OpenAPI format for integration with standard API tools.

```
# OBP Standard
GET /obp/v5.1.0/resource-docs/v5.1.0/swagger

# Berlin Group
GET /obp/v5.1.0/resource-docs/BGv1.3/swagger

# UK Open Banking
GET /obp/v5.1.0/resource-docs/UKv3.1/swagger
```

Import to Postman/Insomnia:

1. Get Swagger JSON from endpoint above
2. Import into API client
3. Configure authentication
4. Test endpoints

Note: The Swagger format is generated from Resource Docs. Resource Docs contain additional information not available in Swagger format.

9.4 Common API Workflows

Workflow 1: Account Information Retrieval

```
# 1. Authenticate
POST /obp/v5.1.0/my/logins/direct
DirectLogin: username=user@example.com,password=pwd,consumer_key=KEY

# 2. Get available banks
GET /obp/v5.1.0/banks

# 3. Get accounts at bank
GET /obp/v5.1.0/banks/gh.29.uk/accounts/private

# 4. Get account details
GET /obp/v5.1.0/banks/gh.29.uk/accounts/ACCOUNT_ID/owner/account

# 5. Get transactions
GET /obp/v5.1.0/banks/gh.29.uk/accounts/ACCOUNT_ID/owner/transactions
```

Workflow 2: Payment Initiation

```
# 1. Authenticate (OAuth2/OIDC recommended)

# 2. Create consent
POST /obp/v5.1.0/my/consents/IMPLICIT
{
  "everything": false,
  "account_access": [...],
  "permissions": ["CanCreateTransactionRequest"]
}

# 3. Create transaction request
POST /obp/v5.1.0/banks/BANK_ID/accounts/ACCOUNT_ID/owner/transaction-request-types/SEPA/transaction-requests
{
  "to": {
    "iban": "DE89370400440532013000"
  },
  "value": {
    "currency": "EUR",
    "amount": "10.00"
  },
  "description": "Payment description"
}

# 4. Answer challenge (if required)
POST /obp/v5.1.0/banks/BANK_ID/accounts/ACCOUNT_ID/owner/transaction-request-types/SEPA/transaction-requests/TR_ID/challenge
{
  "answer": "123456"
}
```

```
# 5. Check transaction status
GET /obp/v5.1.0/banks/BANK_ID/accounts/ACCOUNT_ID/owner/transaction-requests/TR_ID
```

Workflow 3: Consumer Management

```
# 1. Authenticate as admin

# 2. Create consumer
POST /obp/v5.1.0/management/consumers
{
  "app_name": "My Banking App",
  "app_type": "Web",
  "description": "Customer portal",
  "developer_email": "dev@example.com",
  "redirect_url": "https://myapp.com/callback"
}

# 3. Set rate limits
PUT /obp/v5.1.0/management/consumers/CONSUMER_ID/consumer/call-limits
{
  "per_minute_call_limit": "100",
  "per_hour_call_limit": "1000"
}

# 4. Monitor usage
GET /obp/v5.1.0/management/metrics?consumer_id=CONSUMER_ID
```

10. Deployment Workflows

10.1 Development Workflow

```
# 1. Clone and setup
git clone https://github.com/OpenBankProject/OBP-API.git
cd OBP-API
cp obp-api/src/main/resources/props/sample.props.template \
   obp-api/src/main/resources/props/default.props

# 2. Configure for H2 (dev database)
# Edit default.props
db.driver=org.h2.Driver
db.url=jdbc:h2:./obp_api.db;DB_CLOSE_ON_EXIT=FALSE
connector=mapped

# 3. Build and run
mvn clean install -pl .,obp-commons
mvn jetty:run -pl obp-api

# 4. Access
# API: http://localhost:8080
# API Explorer: http://localhost:5173 (separate repo)
```

10.2 Staging Deployment

```
# 1. Setup PostgreSQL
sudo -u postgres psql
CREATE DATABASE obpdb_staging;
CREATE USER obp_staging WITH PASSWORD 'secure_password';
GRANT ALL PRIVILEGES ON DATABASE obpdb_staging TO obp_staging;

# 2. Configure props
# Create production.default.props
db.driver=org.postgresql.Driver
db.url=jdbc:postgresql://localhost:5432/obpdb_staging?
user=obp_staging&password=xxx
connector=mapped
allow_oauth2_login=true

# 3. Build WAR
mvn clean package

# 4. Deploy to Jetty
sudo cp target/OBP-API-1.0.war /usr/share/jetty9/webapps/root.war
sudo systemctl restart jetty9

# 5. Setup API Explorer
cd API-Explorer-II
npm install
npm run build
# Deploy dist/ to web server
```

10.3 Production Deployment (High Availability)

Architecture:

```
                     ┌──────────────┐
                     │  Load        │
                     │  Balancer    │
                     │  (HAProxy)   │
                     └──────┬───────┘
                            │
         ┌──────────────────┼──────────────────┐
         │                  │                  │
    ┌────▼────┐        ┌────▼────┐       ┌────▼────┐
    │ OBP-API │        │ OBP-API │       │ OBP-API │
    │ Node 1  │        │ Node 2  │       │ Node 3  │
    └────┬────┘        └────┬────┘       └────┬────┘
         │                  │                  │
         └──────────────────┴──────────────────┘
                            │
              ┌─────────────┴─────────────┐
              │                           │
         ┌────▼────┐                ┌─────▼────┐
         │ PostgreSQL               │  Redis   │
         │ (Primary +               │ Cluster  │
         │  Replicas)               │          │
         └─────────┘                └──────────┘
```

Steps:

1. **Database Setup (PostgreSQL HA):**

```
# Primary server
postgresql.conf:
  wal_level = replica
  max_wal_senders = 3

# Standby servers
recovery.conf:
  standby_mode = 'on'
  primary_conninfo = 'host=primary port=5432 user=replicator'
```

2. **Redis Cluster:**

```
# 3 masters + 3 replicas
redis-cli --cluster create \
  node1:6379 node2:6379 node3:6379 \
  node4:6379 node5:6379 node6:6379 \
  --cluster-replicas 1
```

3. **OBP-API Configuration (each node):**

```
# PostgreSQL connection
db.url=jdbc:postgresql://pg-primary:5432/obpdb?user=obp&password=xxx

# Redis cluster
cache.redis.url=redis-node1:6379,redis-node2:6379,redis-node3:6379
cache.redis.cluster=true

# Session stickiness (important!)
session.provider=redis
```

4. **HAProxy Configuration:**

```
frontend obp_frontend
    bind *:443 ssl crt /etc/ssl/certs/obp.pem
    default_backend obp_nodes

backend obp_nodes
    balance roundrobin
    option httpchk GET /obp/v5.1.0/root
    cookie SERVERID insert indirect nocache
    server node1 obp-node1:8080 check cookie node1
    server node2 obp-node2:8080 check cookie node2
    server node3 obp-node3:8080 check cookie node3
```

5. **Deploy and Monitor:**

```
# Deploy to all nodes
for node in node1 node2 node3; do
    scp target/OBP-API-1.0.war $node:/usr/share/jetty9/webapps/root.war
    ssh $node "sudo systemctl restart jetty9"
done

# Monitor health
watch -n 5 'curl -s http://lb-endpoint/obp/v5.1.0/root | jq .version'
```

10.4 Docker/Kubernetes Deployment

Kubernetes Manifests:

```
# obp-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: obp-api
spec:
  replicas: 3
  selector:
    matchLabels:
      app: obp-api
  template:
    metadata:
      labels:
        app: obp-api
    spec:
      containers:
        - name: obp-api
          image: openbankproject/obp-api:latest
          ports:
            - containerPort: 8080
          env:
            - name: OBP_DB_DRIVER
              value: "org.postgresql.Driver"
            - name: OBP_DB_URL
              valueFrom:
                secretKeyRef:
                  name: obp-secrets
                  key: db-url
            - name: OBP_CONNECTOR
              value: "mapped"
            - name: OBP_CACHE_REDIS_URL
              value: "redis-service"
          resources:
            requests:
              memory: "2Gi"
              cpu: "1000m"
            limits:
              memory: "4Gi"
              cpu: "2000m"
          livenessProbe:
            httpGet:
              path: /obp/v5.1.0/root
              port: 8080
            initialDelaySeconds: 60
            periodSeconds: 10
          readinessProbe:
            httpGet:
              path: /obp/v5.1.0/root
              port: 8080
            initialDelaySeconds: 30
            periodSeconds: 5
---
apiVersion: v1
kind: Service
metadata:
  name: obp-api-service
spec:
  selector:
    app: obp-api
  ports:
    - port: 80
      targetPort: 8080
  type: LoadBalancer
```

Secrets Management:

```
kubectl create secret generic obp-secrets \
  --from-literal=db-url='jdbc:postgresql://postgres:5432/obpdb?user=obp&password=xxx' \
  --from-literal=oauth-consumer-key='key' \
  --from-literal=oauth-consumer-secret='secret'
```

10.5 Backup and Disaster Recovery

Database Backup:

```
#!/bin/bash
# backup-obp.sh
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_DIR="/backups/obp"

# Backup PostgreSQL
pg_dump -h localhost -U obp obpdb | gzip > \
  $BACKUP_DIR/obpdb_$DATE.sql.gz

# Backup props files
tar -czf $BACKUP_DIR/props_$DATE.tar.gz \
  /path/to/OBP-API/obp-api/src/main/resources/props/

# Upload to S3 (optional)
aws s3 cp $BACKUP_DIR/obpdb_$DATE.sql.gz \
  s3://obp-backups/database/

# Cleanup old backups (keep 30 days)
find $BACKUP_DIR -name "*.sql.gz" -mtime +30 -delete
```

Restore Process:

```
# 1. Stop OBP-API
sudo systemctl stop jetty9

# 2. Restore database
gunzip -c obpdb_20240115.sql.gz | psql -h localhost -U obp obpdb

# 3. Restore configuration
tar -xzf props_20240115.tar.gz -C /path/to/restore/

# 4. Start OBP-API
sudo systemctl start jetty9
```

---

11. Development Guide

11.1 Setting Up Development Environment

Prerequisites:

```
# Install Java
sdk install java 11.0.2-open
```

```
# Install Maven
sdk install maven 3.8.6

# Install SBT (alternative)
sdk install sbt 1.8.2

# Install PostgreSQL
sudo apt install postgresql postgresql-contrib

# Install Redis
sudo apt install redis-server

# Install Git
sudo apt install git
```

IDE Setup (IntelliJ IDEA):

1. Install Scala plugin
2. Import project as Maven project
3. Configure JDK (File → Project Structure → SDK)
4. Set VM options: `-Xmx2048m -XX:MaxPermSize=512m`
5. Configure test runner: Use ScalaTest runner
6. Enable annotation processing

Building from Source:

```
# Clone repository
git clone https://github.com/OpenBankProject/OBP-API.git
cd OBP-API

# Build
mvn clean install -pl .,obp-commons

# Run tests
mvn test

# Run single test
mvn -DwildcardSuites=code.api.directloginTest test

# Run with specific profile
mvn -Pdev clean install
```

11.2 Running Tests

Unit Tests:

```
# All tests
mvn clean test
```

```
# Specific test class
mvn -Dtest=MappedBranchProviderTest test

# Pattern matching
mvn -Dtest=*BranchProvider* test

# With coverage
mvn clean test jacoco:report
```

Integration Tests:

```
# Setup test database
createdb obpdb_test
psql obpdb_test < test-data.sql

# Run integration tests
mvn integration-test -Pintegration

# Test props file
# Create test.default.props
connector=mapped
db.driver=org.h2.Driver
db.url=jdbc:h2:mem:test_db
```

Test Configuration:

```
// In test class
class AccountTest extends ServerSetup {
  override def beforeAll(): Unit = {
    super.beforeAll()
    // Setup test data
  }

  feature("Account operations") {
    scenario("Create account") {
      val request = """{"label": "Test Account"}"""
      When("POST /accounts")
      val response = makePostRequest(request)
      Then("Account should be created")
      response.code should equal(201)
    }
  }
}
```

11.3 Creating Custom Connectors

Connector Structure:

```scala
// CustomConnector.scala
package code.bankconnectors

import code.api.util.OBPQueryParam
import code.bankconnectors.Connector
import net.liftweb.common.Box

object CustomConnector extends Connector {

  val connectorName = "custom_connector_2024"

  override def getBankLegacy(bankId: BankId, callContext: Option[CallContext]):
Box[(Bank, Option[CallContext])] = {
    // Your implementation
    val bank = // Fetch from your backend
    Full((bank, callContext))
  }

  override def getAccountLegacy(bankId: BankId, accountId: AccountId, callContext:
Option[CallContext]): Box[(BankAccount, Option[CallContext])] = {
    // Your implementation
    val account = // Fetch from your backend
    Full((account, callContext))
  }

  // Implement other required methods...
}
```

Registering Connector:

```
# In props file
connector=custom_connector_2024
```

11.4 Creating Dynamic Endpoints

Define Dynamic Endpoint:

```
POST /obp/v5.1.0/management/dynamic-endpoints
{
  "dynamic_endpoint_id": "my-custom-endpoint",
  "swagger_string": "{
    \"swagger\": \"2.0\",
    \"info\": {\"title\": \"Custom API\"},
    \"paths\": {
      \"/custom-data\": {
        \"get\": {
          \"summary\": \"Get custom data\",
          \"responses\": {
            \"200\": {
              \"description\": \"Success\"
            }
          }
        }
      }
    }
  }",
  "bank_id": "gh.29.uk"
}
```

Define Dynamic Entity:

```
POST /obp/v5.1.0/management/dynamic-entities
{
  "dynamic_entity_id": "customer-preferences",
  "entity_name": "CustomerPreferences",
  "bank_id": "gh.29.uk"
}
```

11.5 Code Style and Conventions

Scala Code Style:

```scala
// Good practices
class AccountService {

  // Use descriptive names
  def createNewAccount(bankId: BankId, userId: UserId): Future[Box[Account]] = {

    // Use pattern matching
    account match {
      case Full(acc) => Future.successful(Full(acc))
      case Empty => Future.successful(Empty)
      case Failure(msg, _, _) => Future.successful(Failure(msg))
    }

    // Use for-comprehensions
    for {
      bank <- getBankFuture(bankId)
      user <- getUserFuture(userId)
      account <- createAccountFuture(bank, user)
    } yield account
  }

  // Document public APIs
  /**
   * Retrieves account by ID
   * @param bankId The bank identifier
   * @param accountId The account identifier
   * @return Box containing account or error
   */
  def getAccount(bankId: BankId, accountId: AccountId): Box[Account] = {
    // Implementation
  }
}
```

11.6 Contributing to OBP

Contribution Workflow:

1. Fork the repository
2. Create feature branch: `git checkout -b feature/amazing-feature`
3. Make changes following code style
4. Write/update tests
5. Run tests: `mvn test`
6. Commit: `git commit -m 'Add amazing feature'`
7. Push: `git push origin feature/amazing-feature`
8. Create Pull Request

Pull Request Checklist:

- □ Tests pass
- □ Code follows style guidelines
- □ Documentation updated
- □ Changelog updated (if applicable)
- □ No merge conflicts
- □ Descriptive PR title and description

Signing Contributor Agreement:

- Required for first-time contributors
- Sign the Harmony CLA
- Preserves open-source license

---

12. Roadmap and Future Development

12.1 Overview

The Open Bank Project follows an agile roadmap that evolves based on feedback from banks, regulators, developers, and the community. This section outlines current and future developments across the OBP ecosystem.

12.2 OBP-API-II (Next Generation API)

Status: Experimental

Purpose: A modernized version of OBP-API for selected endpoints.

Architecture Enhancements:

- Fewer dependencies including Jetty.

Technology Stack:

- Scala 2.13/3.x (upgraded from 2.12)

12.3 OBP-Dispatch (Request Router)

Status: In Development

Purpose: A lightweight proxy/router to route API requests to different OBP-API implementations.

Key Features:

Routing:

- Route traffic according to Resouce Docs available on OBP-API-II, OBP-Trading or OBP-API

Use Cases:

1. **Implementation Migration:**
   - Re-Implement an endpoint in OBP-API-II
2. **New Endpoint implementation:**
   - Implement a new endpoint in OBP-API-II or OBP-Trading

Deployment:

```
# Build
mvn clean package

# Run
java -jar target/OBP-API-Dispatch-1.0-SNAPSHOT-jar-with-dependencies.jar

# Docker
docker run -p 8080:8080 \
  -v /path/to/application.conf:/config/application.conf \
  obp-dispatch:latest
```

Architecture:

```
                    ┌──────────────────┐
                    │   OBP-Dispatch   │
                    │   (Port 8080)    │
                    └────────┬─────────┘
                             │
         ┌───────────────────┼───────────────────┐
         │                   │                   │
```

```
      ┌────▼────┐          ┌────▼────┐         ┌────▼────┐
      │ OBP-API  │          │ OBP-API-II │       │ OBP-Trading │
      │ v4.0.0   │          │ v5.0.0   │         │ v5.1.0   │
      │ (EU)     │          │ (US)     │         │ (APAC)   │
      └──────────┘          └──────────┘         └──────────┘
```

12.1 Glossary

Account: Bank account holding funds

API Explorer: Interactive API documentation tool

Bank: Financial institution entity in OBP (also called "Space")

Connector: Plugin that connects OBP-API to backend systems

Consumer: OAuth client application (has consumer key/secret)

Consent: Permission granted by user for data access

Direct Login: Username/password authentication method

Dynamic Entity: User-defined data structure

Dynamic Endpoint: User-defined API endpoint

Entitlement: Permission to perform specific operation (same as Role)

OIDC: OpenID Connect identity layer

Opey: AI-powered banking assistant

Props: Configuration properties file

Role: Permission granted to user (same as Entitlement)

Sandbox: Development/testing environment

SCA: Strong Customer Authentication (PSD2 requirement)

View: Permission set controlling data visibility

Webhook: HTTP callback triggered by events

See the OBP Glossary for a full list of terms.

12.2 Environment Variables Reference

OBP-API Environment Variables:

```
# Database
OBP_DB_DRIVER=org.postgresql.Driver
OBP_DB_URL=jdbc:postgresql://localhost:5432/obpdb
```

```
# Connector
OBP_CONNECTOR=mapped

# Redis
OBP_CACHE_REDIS_URL=localhost
OBP_CACHE_REDIS_PORT=6379

# OAuth
OBP_OAUTH_CONSUMER_KEY=key
OBP_OAUTH_CONSUMER_SECRET=secret

# OIDC
OBP_OAUTH2_JWK_SET_URL=http://oidc:9000/jwks
OBP_OPENID_CONNECT_ENABLED=true

# Rate Limiting
OBP_USE_CONSUMER_LIMITS=true

# Logging
OBP_LOG_LEVEL=INFO
```

Opey II Environment Variables:

```
# LLM Provider
MODEL_PROVIDER=anthropic
MODEL_NAME=claude-sonnet-4
ANTHROPIC_API_KEY=sk-...

# OBP API
OBP_BASE_URL=http://localhost:8080
OBP_USERNAME=user@example.com
OBP_PASSWORD=password
OBP_CONSUMER_KEY=consumer-key

# Vector Database
QDRANT_HOST=localhost
QDRANT_PORT=6333

# Tracing
LANGCHAIN_TRACING_V2=true
LANGCHAIN_API_KEY=lsv2_pt_...
```

12.3 Props File Complete Reference

Core Settings:

```
# Server Mode
server_mode=apis,portal  # portal | apis | apis,portal
```

```
run.mode=production  # development | production | test

# HTTP Server
http.port=8080
https.port=8443

# Database
db.driver=org.postgresql.Driver
db.url=jdbc:postgresql://localhost:5432/obpdb?user=obp&password=xxx

# Connector
connector=mapped  # mapped | kafka | akka | rest | star

# Redis Cache
cache.redis.url=127.0.0.1
cache.redis.port=6379

# OAuth 1.0a
allow_oauth1_login=true

# OAuth 2.0
allow_oauth2_login=true
oauth2.jwk_set.url=http://localhost:9000/jwks

# OpenID Connect
openid_connect_1.button_text=Login
openid_connect_1.client_id=client-id
openid_connect_1.client_secret=secret
openid_connect_1.callback_url=http://localhost:8080/callback
openid_connect_1.endpoint.discovery=http://oidc/.well-known/openid-configuration

# Rate Limiting
use_consumer_limits=true
user_consumer_limit_anonymous_access=60

# Admin
super_admin_user_ids=uuid1,uuid2

# Sandbox
allow_sandbox_data_import=true

# API Explorer
api_explorer_url=http://localhost:5173

# Security
jwt.use.ssl=false
keystore.path=/path/to/keystore.jks

# Webhooks
webhooks.enabled=true

# Akka
akka.remote.enabled=false
akka.remote.hostname=localhost
akka.remote.port=2662

# Elasticsearch
es.metrics.enabled=false
es.metrics.url=http://localhost:9200

# Session
session.timeout.minutes=30

# CORS
allow_cors=true
allowed_origins=http://localhost:5173
```

12.4 Complete Error Codes Reference

Infrastructure / Config Level (OBP-00XXX)

Error Code	Message	Description
OBP-00001	Hostname not specified	Props configuration missing hostname
OBP-00002	Data import disabled	Sandbox data import not enabled
OBP-00003	Transaction disabled	Transaction requests not enabled
OBP-00005	Public views not allowed	Public views disabled in props
OBP-00008	API version not supported	Requested API version not enabled
OBP-00009	Account firehose not allowed	Account firehose disabled in props
OBP-00010	Missing props value	Required property not configured
OBP-00011	No valid Elasticsearch indices	ES indices not configured
OBP-00012	Customer firehose not allowed	Customer firehose disabled
OBP-00013	API instance id not specified	Instance ID missing from props
OBP-00014	Mandatory properties not set	Required props missing

Exceptions (OBP-01XXX)

Error Code	Message	Description
OBP-01000	Request timeout	Backend service timeout

WebUI Props (OBP-08XXX)

Error Code	Message	Description
OBP-08001	Invalid WebUI props format	Name format incorrect
OBP-08002	WebUI props not found	Invalid WEB_UI_PROPS_ID

Dynamic Entities/Endpoints (OBP-09XXX)

Error Code	Message	Description
OBP-09001	DynamicEntity not found	Invalid DYNAMIC_ENTITY_ID
OBP-09002	DynamicEntity name exists	Duplicate entityName
OBP-09003	DynamicEntity not exists	Check entityName
OBP-09004	DynamicEntity missing argument	Required argument missing
OBP-09005	Entity not found	Invalid entityId
OBP-09006	Operation not allowed	Data exists, cannot delete
OBP-09007	Validation failure	Data validation failed
OBP-09008	DynamicEndpoint exists	Duplicate endpoint
OBP-09009	DynamicEndpoint not found	Invalid DYNAMIC_ENDPOINT_ID
OBP-09010	Invalid user for DynamicEntity	Not the creator
OBP-09011	Invalid user for DynamicEndpoint	Not the creator
OBP-09013	Invalid Swagger JSON	DynamicEndpoint Swagger invalid
OBP-09014	Invalid request payload	JSON doesn't match validation
OBP-09015	Dynamic data not found	Invalid data reference
OBP-09016	Duplicate query parameters	Query params must be unique
OBP-09017	Duplicate header keys	Header keys must be unique

General Messages (OBP-10XXX)

Error Code	Message	Description
OBP-10001	Incorrect JSON format	JSON syntax error
OBP-10002	Invalid number	Cannot convert to number
OBP-10003	Invalid ISO currency code	Not a valid 3-letter code
OBP-10004	FX currency not supported	Invalid currency pair
OBP-10005	Invalid date format	Cannot parse date
OBP-10006	Invalid currency value	Currency value invalid
OBP-10007	Incorrect role name	Role name invalid
OBP-10008	Cannot transform JSON	JSON to model failed
OBP-10009	Cannot save resource	Save/update failed
OBP-10010	Not implemented	Feature not implemented

Error Code	Message	Description
OBP-10011	Invalid future date	Date must be in future
OBP-10012	Maximum limit exceeded	Max value is 10000
OBP-10013	Empty box	Attempted to open empty box
OBP-10014	Cannot decrypt property	Decryption failed
OBP-10015	Allowed values	Invalid value provided
OBP-10016	Invalid filter parameters	URL filter incorrect
OBP-10017	Incorrect URL format	URL format invalid
OBP-10018	Too many requests	Rate limit exceeded
OBP-10019	Invalid boolean	Cannot convert to boolean
OBP-10020	Incorrect JSON	JSON content invalid
OBP-10021	Invalid connector name	Connector name incorrect
OBP-10022	Invalid connector method	Method name incorrect
OBP-10023	Sort direction error	Use DESC or ASC
OBP-10024	Invalid offset	Must be positive integer
OBP-10025	Invalid limit	Must be >= 1
OBP-10026	Date format error	Wrong date string format
OBP-10028	Invalid anon parameter	Use TRUE or FALSE
OBP-10029	Invalid duration	Must be positive integer
OBP-10030	SCA method not defined	No SCA method configured
OBP-10031	Invalid outbound mapping	JSON structure invalid
OBP-10032	Invalid inbound mapping	JSON structure invalid
OBP-10033	Invalid IBAN	IBAN format incorrect
OBP-10034	Invalid URL parameters	URL params invalid
OBP-10035	Invalid JSON value	JSON value incorrect
OBP-10036	Invalid is_deleted	Use TRUE or FALSE
OBP-10037	Invalid HTTP method	HTTP method incorrect
OBP-10038	Invalid HTTP protocol	Protocol incorrect
OBP-10039	Incorrect trigger name	Trigger name invalid
OBP-10040	Service too busy	Try again later
OBP-10041	Invalid locale	Unsupported locale

Error Code	Message	Description
OBP-10050	Cannot create FX currency	FX creation failed
OBP-10051	Invalid log level	Log level invalid
OBP-10404	404 Not Found	URI not found
OBP-10405	Resource does not exist	Resource not found

Authentication/Authorization (OBP-20XXX)

Error Code	Message	Description
OBP-20001	User not logged in	Authentication required
OBP-20002	DirectLogin missing parameters	Required params missing
OBP-20003	DirectLogin invalid token	Token invalid or expired
OBP-20004	Invalid login credentials	Username/password wrong
OBP-20005	User not found by ID	Invalid USER_ID
OBP-20006	User missing roles	Insufficient entitlements
OBP-20007	User not found by email	Email not found
OBP-20008	Invalid consumer key	Consumer key invalid
OBP-20009	Invalid consumer credentials	Credentials incorrect
OBP-20010	Value too long	Value exceeds limit
OBP-20011	Invalid characters	Invalid chars in value
OBP-20012	Invalid DirectLogin parameters	Parameters incorrect
OBP-20013	Account locked	User account locked
OBP-20014	Invalid consumer ID	Invalid CONSUMER_ID
OBP-20015	No permission to update consumer	Not the creator
OBP-20016	Unexpected login error	Login error occurred
OBP-20017	No view access	No access to VIEW_ID
OBP-20018	Invalid redirect URL	Internal redirect invalid
OBP-20019	No owner view	User lacks owner view
OBP-20020	Invalid custom view format	Must start with _
OBP-20021	System views immutable	Cannot modify system views
OBP-20022	View permission denied	View doesn't permit access
OBP-20023	Consumer missing roles	Insufficient consumer roles

Error Code	Message	Description
OBP-20024	Consumer not found	Invalid CONSUMER_ID
OBP-20025	Scope not found	Invalid SCOPE_ID
OBP-20026	Consumer lacks scope	Missing SCOPE_ID
OBP-20027	User not found	Provider/username not found
OBP-20028	GatewayLogin missing params	Parameters missing
OBP-20029	GatewayLogin error	Unknown error
OBP-20030	Gateway host missing	Property not defined
OBP-20031	Gateway whitelist	Not allowed address
OBP-20040	Gateway JWT invalid	JWT corrupted
OBP-20041	Cannot extract JWT	JWT extraction failed
OBP-20042	No need to call CBS	CBS call unnecessary
OBP-20043	Cannot find user	User not found
OBP-20044	Cannot get CBS token	CBS token failed
OBP-20045	Cannot get/create user	User operation failed
OBP-20046	No JWT for response	JWT unavailable
OBP-20047	Insufficient grant permission	Cannot grant view access
OBP-20048	Insufficient revoke permission	Cannot revoke view access
OBP-20049	Source view less permission	Fewer permissions than target
OBP-20050	Not super admin	User not super admin
OBP-20051	Elasticsearch index not found	ES index missing
OBP-20052	Result set too small	Privacy threshold
OBP-20053	ES query body empty	Query cannot be empty
OBP-20054	Invalid amount	Amount value invalid
OBP-20055	Missing query params	Required params missing
OBP-20056	Elasticsearch disabled	ES not enabled
OBP-20057	User not found by userId	Invalid userId
OBP-20058	Consumer disabled	Consumer is disabled
OBP-20059	Cannot assign account access	Assignment failed
OBP-20060	No read access	User lacks view access
OBP-20062	Frequency per day error	Invalid frequency value

Error Code	Message	Description
OBP-20063	Frequency must be one	One-off requires freq=1
OBP-20064	User deleted	User is deleted
OBP-20065	Cannot get/create DAuth user	DAuth user failed
OBP-20066	DAuth missing parameters	Parameters missing
OBP-20067	DAuth unknown error	Unknown DAuth error
OBP-20068	DAuth host missing	Property not defined
OBP-20069	DAuth whitelist	Not allowed address
OBP-20070	No DAuth JWT	JWT unavailable
OBP-20071	DAuth JWT invalid	JWT corrupted
OBP-20072	Invalid DAuth header	Header format wrong
OBP-20079	Invalid provider URL	Provider mismatch
OBP-20080	Invalid auth header	Header format unsupported
OBP-20081	User attribute not found	Invalid USER_ATTRIBUTE_ID
OBP-20082	Missing DirectLogin header	Header missing
OBP-20083	Invalid DirectLogin header	Missing DirectLogin word
OBP-20084	Cannot grant system view	Insufficient permissions
OBP-20085	Cannot grant custom view	Permission denied
OBP-20086	Cannot revoke system view	Insufficient permissions
OBP-20087	Cannot revoke custom view	Permission denied
OBP-20088	Consent access empty	Access must be requested
OBP-20089	Recurring indicator invalid	Must be false for allAccounts
OBP-20090	Frequency invalid	Must be 1 for allAccounts
OBP-20091	Invalid availableAccounts	Must be 'allAccounts'
OBP-20101	Not super admin or missing role	Admin check failed
OBP-20102	Cannot get/create user	User operation failed
OBP-20103	Invalid user provider	Provider invalid
OBP-20104	User not found	Provider/ID not found
OBP-20105	Balance not found	Invalid BALANCE_ID

OAuth 2.0 (OBP-202XX)

Error Code	Message	Description
OBP-20200	Application not identified	Cannot identify app
OBP-20201	OAuth2 not allowed	OAuth2 disabled
OBP-20202	Cannot verify JWT	JWT verification failed
OBP-20203	No JWKS URL	JWKS URL missing
OBP-20204	Bad JWT	JWT error
OBP-20205	Parse error	Parsing failed
OBP-20206	Bad JOSE	JOSE exception
OBP-20207	JOSE exception	Internal JOSE error
OBP-20208	Cannot match issuer/JWKS	Issuer/JWKS mismatch
OBP-20209	Token has no consumer	Consumer not linked
OBP-20210	Certificate mismatch	Different certificate
OBP-20211	OTP expired	One-time password expired
OBP-20213	Token endpoint auth forbidden	Auth method unsupported
OBP-20214	OAuth2 not recognized	Token not recognized
OBP-20215	Token validation error	Validation problem
OBP-20216	Invalid OTP	One-time password invalid

Headers (OBP-2025X)

Error Code	Message	Description
OBP-20250	Authorization ambiguity	Ambiguous auth headers
OBP-20251	Missing mandatory headers	Required headers missing
OBP-20252	Empty request headers	Null/empty not allowed
OBP-20253	Invalid UUID	Must be UUID format
OBP-20254	Invalid Signature header	Signature header invalid
OBP-20255	Request ID already used	Duplicate request ID
OBP-20256	Invalid Consent-Id usage	Header misuse
OBP-20257	Invalid RFC 7231 date	Date format wrong

X.509 Certificates (OBP-203XX)

Error Code	Message	Description

Error Code	Message	Description
OBP-20300	PEM certificate issue	Certificate error
OBP-20301	Parsing failed	Cannot parse PEM
OBP-20302	Certificate expired	Cert is expired
OBP-20303	Certificate not yet valid	Cert not active yet
OBP-20304	No RSA public key	RSA key not found
OBP-20305	No EC public key	EC key not found
OBP-20306	No certificate	Cert not in header
OBP-20307	Action not allowed	Insufficient PSD2 role
OBP-20308	No PSD2 roles	PSD2 roles missing
OBP-20309	No public key	Public key missing
OBP-20310	Cannot verify signature	Signature verification failed
OBP-20311	Request not signed	Signature missing
OBP-20312	Cannot validate public key	Key validation failed

OpenID Connect (OBP-204XX)

Error Code	Message	Description
OBP-20400	Cannot exchange code	Token exchange failed
OBP-20401	Cannot save OIDC user	User save failed
OBP-20402	Cannot save OIDC token	Token save failed
OBP-20403	Invalid OIDC state	State parameter invalid
OBP-20404	Cannot handle OIDC data	Data handling failed
OBP-20405	Cannot validate ID token	ID token invalid

Resources (OBP-30XXX)

Error Code	Message	Description
OBP-30001	Bank not found	Invalid BANK_ID
OBP-30002	Customer not found	Invalid CUSTOMER_NUMBER
OBP-30003	Account not found	Invalid ACCOUNT_ID
OBP-30004	Counterparty not found	Invalid account reference
OBP-30005	View not found	Invalid VIEW_ID

Error Code	Message	Description
OBP-30006	Customer number exists	Duplicate customer number
OBP-30007	Customer already exists	User already linked
OBP-30008	User customer link not found	Link not found
OBP-30009	ATM not found	Invalid ATM_ID
OBP-30010	Branch not found	Invalid BRANCH_ID
OBP-30011	Product not found	Invalid PRODUCT_CODE
OBP-30012	Counterparty not found	Invalid IBAN
OBP-30013	Counterparty not beneficiary	Not a beneficiary
OBP-30014	Counterparty exists	Duplicate counterparty
OBP-30015	Cannot create branch	Insert failed
OBP-30016	Cannot update branch	Update failed
OBP-30017	Counterparty not found	Invalid COUNTERPARTY_ID
OBP-30018	Bank account not found	Invalid BANK_ID/ACCOUNT_ID
OBP-30019	Consumer not found	Invalid CONSUMER_ID
OBP-30020	Cannot create bank	Insert failed
OBP-30021	Cannot update bank	Update failed
OBP-30022	No view permission	Permission missing
OBP-30023	Cannot update consumer	Update failed
OBP-30024	Cannot create consumer	Insert failed
OBP-30025	Cannot create user link	Link creation failed
OBP-30026	Consumer key exists	Duplicate key
OBP-30027	No account holders	Holders not found
OBP-30028	Cannot create ATM	Insert failed
OBP-30029	Cannot update ATM	Update failed
OBP-30030	Cannot create product	Insert failed
OBP-30031	Cannot update product	Update failed
OBP-30032	Cannot create card	Insert failed
OBP-30033	Cannot update card	Update failed
OBP-30034	ViewId not supported	Invalid VIEW_ID
OBP-30035	User customer link not found	Link not found

Error Code	Message	Description
OBP-30036	Cannot create counterparty metadata	Insert failed
OBP-30037	Counterparty metadata not found	Metadata missing
OBP-30038	Cannot create FX rate	Insert failed
OBP-30039	Cannot update FX rate	Update failed
OBP-30040	Unknown FX rate error	FX error
OBP-30041	Checkbook order not found	Order not found
OBP-30042	Cannot get top APIs	Database error
OBP-30043	Cannot get aggregate metrics	Database error
OBP-30044	Default bank ID not set	Property missing
OBP-30045	Cannot get top consumers	Database error
OBP-30046	Customer not found	Invalid CUSTOMER_ID
OBP-30047	Cannot create webhook	Insert failed
OBP-30048	Cannot get webhooks	Retrieval failed
OBP-30049	Cannot update webhook	Update failed
OBP-30050	Webhook not found	Invalid webhook ID
OBP-30051	Cannot create customer	Insert failed
OBP-30052	Cannot check customer	Check failed
OBP-30053	Cannot create user auth context	Insert failed
OBP-30054	Cannot update user auth context	Update failed
OBP-30055	User auth context not found	Invalid USER_ID
OBP-30056	User auth context not found	Invalid context ID
OBP-30057	User auth context update not found	Update not found
OBP-30058	Cannot update customer	Update failed
OBP-30059	Card not found	Card not found
OBP-30060	Card exists	Duplicate card
OBP-30061	Card attribute not found	Invalid attribute ID
OBP-30062	Parent product not found	Invalid parent code
OBP-30063	Cannot grant account access	Grant failed
OBP-30064	Cannot revoke account access	Revoke failed
OBP-30065	Cannot find account access	Access not found

Error Code	Message	Description
OBP-30066	Cannot get accounts	Retrieval failed
OBP-30067	Transaction not found	Invalid TRANSACTION_ID
OBP-30068	Transaction refunded	Already refunded
OBP-30069	Customer attribute not found	Invalid attribute ID
OBP-30070	Transaction attribute not found	Invalid attribute ID
OBP-30071	Attribute not found	Invalid definition ID
OBP-30072	Cannot create counterparty	Insert failed
OBP-30073	Account not found	Invalid routing
OBP-30074	Account not found	Invalid IBAN
OBP-30075	Account routing not found	Routing invalid
OBP-30076	Account not found	Invalid ACCOUNT_ID
OBP-30077	Cannot create OAuth2 consumer	Insert failed
OBP-30078	Transaction request attribute not found	Invalid attribute ID
OBP-30079	API collection not found	Collection missing
OBP-30080	Cannot create API collection	Insert failed
OBP-30081	Cannot delete API collection	Delete failed
OBP-30082	API collection endpoint not found	Endpoint missing
OBP-30083	Cannot create endpoint	Insert failed
OBP-30084	Cannot delete endpoint	Delete failed
OBP-30085	Endpoint exists	Duplicate endpoint
OBP-30086	Collection exists	Duplicate collection
OBP-30087	Double entry transaction not found	Transaction missing
OBP-30088	Invalid auth context key	Key invalid
OBP-30089	Cannot update ATM languages	Update failed
OBP-30091	Cannot update ATM currencies	Update failed
OBP-30092	Cannot update ATM accessibility	Update failed
OBP-30093	Cannot update ATM services	Update failed
OBP-30094	Cannot update ATM notes	Update failed
OBP-30095	Cannot update ATM categories	Update failed
OBP-30096	Cannot create endpoint tag	Insert failed

Error Code	Message	Description
OBP-30097	Cannot update endpoint tag	Update failed
OBP-30098	Unknown endpoint tag error	Tag error
OBP-30099	Endpoint tag not found	Invalid tag ID
OBP-30100	Endpoint tag exists	Duplicate tag
OBP-30101	Meetings not supported	Feature disabled
OBP-30102	Meeting API key missing	Key not configured
OBP-30103	Meeting secret missing	Secret not configured
OBP-30104	Meeting not found	Meeting missing
OBP-30105	Invalid balance currency	Currency invalid
OBP-30106	Invalid balance amount	Amount invalid
OBP-30107	Invalid user ID	USER_ID invalid
OBP-30108	Invalid account type	Type invalid
OBP-30109	Initial balance must be zero	Must be 0
OBP-30110	Invalid account ID format	Format invalid
OBP-30111	Invalid bank ID format	Format invalid
OBP-30112	Invalid initial balance	Not a number
OBP-30113	Invalid customer bank	Wrong bank
OBP-30114	Invalid account routings	Routing invalid
OBP-30115	Account routing exists	Duplicate routing
OBP-30116	Invalid payment system	Name invalid
OBP-30117	Product fee not found	Invalid fee ID
OBP-30118	Cannot create product fee	Insert failed
OBP-30119	Cannot update product fee	Update failed
OBP-30120	Cannot delete ATM	Delete failed
OBP-30200	Card not found	Invalid CARD_NUMBER
OBP-30201	Agent not found	Invalid AGENT_ID
OBP-30202	Cannot create agent	Insert failed
OBP-30203	Cannot update agent	Update failed
OBP-30204	Customer account link not found	Link missing
OBP-30205	Entitlement is bank role	Need bank_id

Error Code	Message	Description
OBP-30206	Entitlement is system role	bank_id must be empty
OBP-30207	Invalid password format	Password too weak
OBP-30208	Account ID exists	Duplicate ACCOUNT_ID
OBP-30209	Insufficient auth for branch	Missing role
OBP-30210	Insufficient auth for bank	Missing role
OBP-30211	Invalid connector	Invalid CONNECTOR
OBP-30212	Entitlement not found	Invalid entitlement ID
OBP-30213	User lacks entitlement	Missing ENTITLEMENT_ID
OBP-30214	Entitlement request exists	Duplicate request
OBP-30215	Entitlement request not found	Request missing
OBP-30216	Entitlement exists	Duplicate entitlement
OBP-30217	Cannot add entitlement request	Insert failed
OBP-30218	Insufficient auth to delete	Missing role
OBP-30219	Cannot delete entitlement	Delete failed
OBP-30220	Cannot grant entitlement	Grant failed
OBP-30221	Cannot grant - grantor issue	Insufficient privileges
OBP-30222	Counterparty not found	Invalid routings
OBP-30223	Account already linked	Customer link exists
OBP-30224	Cannot create link	Link creation failed
OBP-30225	Link not found	Invalid link ID
OBP-30226	Cannot get links	Retrieval failed
OBP-30227	Cannot update link	Update failed
OBP-30228	Cannot delete link	Delete failed
OBP-30229	Cannot get consent	Implicit SCA failed
OBP-30250	Cannot create system view	Insert failed
OBP-30251	Cannot delete system view	Delete failed
OBP-30252	System view not found	Invalid VIEW_ID
OBP-30253	Cannot update system view	Update failed
OBP-30254	System view exists	Duplicate view
OBP-30255	Empty view name	Name required

Error Code	Message	Description
OBP-30256	Cannot delete custom view	Delete failed
OBP-30257	Cannot find custom view	View missing
OBP-30258	System view cannot be public	Not allowed
OBP-30259	Cannot create custom view	Insert failed
OBP-30260	Cannot update custom view	Update failed
OBP-30261	Cannot create counterparty limit	Insert failed
OBP-30262	Cannot update counterparty limit	Update failed
OBP-30263	Counterparty limit not found	Limit missing
OBP-30264	Counterparty limit exists	Duplicate limit
OBP-30265	Cannot delete limit	Delete failed
OBP-30266	Custom view exists	Duplicate view
OBP-30267	User lacks permission	Permission missing
OBP-30268	Limit validation error	Validation failed
OBP-30269	Account number ambiguous	Multiple matches
OBP-30270	Invalid account number	Number invalid
OBP-30271	Account not found	Invalid routings
OBP-30300	Tax residence not found	Invalid residence ID
OBP-30310	Customer address not found	Invalid address ID
OBP-30311	Account application not found	Invalid application ID
OBP-30312	Resource user not found	Invalid USER_ID
OBP-30313	Missing userId and customerId	Both missing
OBP-30314	Application already accepted	Already processed
OBP-30315	Cannot update status	Update failed
OBP-30316	Cannot create application	Insert failed
OBP-30317	Cannot delete counterparty	Delete failed
OBP-30318	Cannot delete metadata	Delete failed
OBP-30319	Cannot update label	Update failed
OBP-30320	Cannot get product	Retrieval failed
OBP-30321	Cannot get product tree	Retrieval failed
OBP-30323	Cannot get charge value	Retrieval failed

Error Code	Message	Description
OBP-30324	Cannot get charges	Retrieval failed
OBP-30325	Agent account link not found	Link missing
OBP-30326	Agents not found	No agents
OBP-30327	Cannot create agent link	Insert failed
OBP-30328	Agent number exists	Duplicate number
OBP-30329	Cannot get agent links	Retrieval failed
OBP-30330	Agent not beneficiary	Not confirmed
OBP-30331	Invalid entitlement name	Name invalid

| OBP-

12.5 Useful API Endpoints Reference

System Information:

```
GET  /obp/v5.1.0/root                                 # API version info
GET  /obp/v5.1.0/rate-limiting                        # Rate limit status
GET  /obp/v5.1.0/connector-loopback                  # Connector health
GET  /obp/v5.1.0/database/info                       # Database info
```

Authentication:

```
POST /obp/v5.1.0/my/logins/direct                    # Direct login
GET  /obp/v5.1.0/users/current                       # Current user
GET  /obp/v5.1.0/my/spaces                           # User banks
```

Account Operations:

```
GET  /obp/v5.1.0/banks                               # List banks
GET  /obp/v5.1.0/banks/BANK_ID/accounts/private  # User accounts
GET  /obp/v5.1.0/banks/BANK_ID/accounts/ACCOUNT_ID/VIEW_ID/account
POST /obp/v5.1.0/banks/BANK_ID/accounts       # Create account
```

Transaction Operations:

```
GET  /obp/v5.1.0/banks/BANK_ID/accounts/ACCOUNT_ID/VIEW_ID/transactions
POST /obp/v5.1.0/banks/BANK_ID/accounts/ACCOUNT_ID/VIEW_ID/transaction-request-
types/TYPE/transaction-requests
```

Admin Operations:

```
GET  /obp/v5.1.0/management/metrics            # API metrics
GET  /obp/v5.1.0/management/consumers          # List consumers
POST /obp/v5.1.0/users/USER_ID/entitlements   # Grant role
GET  /obp/v5.1.0/users                         # List users
```

12.8 Resources and Links

Official Resources:

- Website: https://www.openbankproject.com
- GitHub: https://github.com/OpenBankProject
- API Sandbox: https://apisandbox.openbankproject.com
- API Explorer: https://apiexplorer-ii-sandbox.openbankproject.com

Standards:

- Berlin Group: https://www.berlin-group.org
- UK Open Banking: https://www.openbanking.org.uk
- PSD2: https://ec.europa.eu/info/law/payment-services-psd-2-directive-eu-2015-2366_en
- FAPI: https://openid.net/wg/fapi/
- Open Bank Project: https://apiexplorer-ii-sandbox.openbankproject.com

Community:

- RocketChat: https://chat.openbankproject.com
- Twitter: @openbankproject

Support:

- Issues: https://github.com/OpenBankProject/OBP-API/issues
- Email: contact@tesobe.com
- Commercial Support: https://www.tesobe.com

12.9 Version History

Major Releases:

- v5.1.0 (2024) - Enhanced OIDC, Dynamic endpoints
- v5.0.0 (2022) - Major refactoring, Performance improvements
- v4.0.0 (2022) - Berlin Group, UK Open Banking support
- v3.1.0 (2020) - Rate limiting, Webhooks
- v3.0.0 (2020) - OAuth 2.0, OIDC support
- v2.2.0 (2018) - Consent management
- v2.0.0 (2017) - API standardization
- v1.4.0 (2016) - Early Release

Status Definitions:

- **STABLE:** Production-ready, guaranteed backward compatibility
- **DRAFT:** Under development, may change
- **BLEEDING-EDGE:** Latest features, experimental
- **DEPRECATED:** No longer maintained

Conclusion

For the latest updates visit Open Bank Project GitHub or contact TESOBE. **This Document Version:** 0.2 **Last Updated:** October 29 2025 **Maintained By:** TESOBE GmbH **License:** AGPL V3

6. Authentication and Security

6.1 Authentication Methods

6.1.1 OAuth 1.0a

Overview: Legacy OAuth method, still supported for backward compatibility

Flow:

1. Request temporary credentials (request token)
2. Redirect user to authorization endpoint
3. User grants access
4. Exchange request token for access token
5. Use access token for API requests

Configuration:

```
# Enable OAuth 1.0a (enabled by default)
allow_oauth1=true
```

Example Request:

```
GET /obp/v4.0.0/users/current
Authorization: OAuth oauth_consumer_key="xxx",
                     oauth_token="xxx",
                     oauth_signature_method="HMAC-SHA1",
                     oauth_signature="xxx",
                     oauth_timestamp="1234567890",
                     oauth_nonce="xxx",
                     oauth_version="1.0"
```

6.1.2 OAuth 2.0 / OpenID Connect

Overview: Modern OAuth2 with OIDC for authentication

Supported Grant Types:

- Authorization Code (recommended)
- Implicit (deprecated, for legacy clients)
- Client Credentials
- Resource Owner Password Credentials

Configuration:

```
# Enable OAuth2
allow_oauth2_login=true

# JWKS URI for token validation (can be comma-separated list)
oauth2.jwk_set.url=http://localhost:9000/obp-oidc/jwks,https://www.googleapis.com/oauth2/v3/certs

# OIDC Provider Configuration
openid_connect_1.client_id=obp-client
openid_connect_1.client_secret=your-secret
openid_connect_1.callback_url=http://localhost:8080/auth/openid-connect/callback
openid_connect_1.endpoint.discovery=http://localhost:9000/.well-known/openid-configuration
openid_connect_1.endpoint.authorization=http://localhost:9000/auth
openid_connect_1.endpoint.token=http://localhost:9000/token
openid_connect_1.endpoint.userinfo=http://localhost:9000/userinfo
openid_connect_1.endpoint.jwks_uri=http://localhost:9000/jwks
openid_connect_1.access_type_offline=true
openid_connect_1.button_text=Login with OIDC
```

Multiple OIDC Providers:

```
# Google
openid_connect_1.client_id=xxx.apps.googleusercontent.com
openid_connect_1.client_secret=xxx
openid_connect_1.endpoint.discovery=https://accounts.google.com/.well-known/openid-configuration
openid_connect_1.button_text=Google

# Keycloak
openid_connect_2.client_id=obp-client
openid_connect_2.client_secret=xxx
openid_connect_2.endpoint.discovery=http://keycloak:8080/realms/obp/.well-known/openid-configuration
openid_connect_2.button_text=Keycloak
```

Authorization Code Flow:

```
1. Authorization Request:
GET /auth?response_type=code
    &client_id=xxx
    &redirect_uri=http://localhost:8080/callback
    &scope=openid profile email
    &state=random-state

2. Token Exchange:
POST /token
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code
&code=xxx
&redirect_uri=http://localhost:8080/callback
&client_id=xxx
&client_secret=xxx

3. API Request with Token:
GET /obp/v4.0.0/users/current
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...
```

6.1.3 Direct Login

Overview: Simplified authentication method for trusted applications

Characteristics:

- Username/password exchange for token
- No OAuth redirect flow
- Suitable for mobile apps and trusted clients
- Time-limited tokens

Configuration:

```
allow_direct_login=true
direct_login_consumer_key=your-trusted-consumer-key
```

Login Request:

```
POST /my/logins/direct
Authorization: DirectLogin username="user@example.com",
                          password="xxx",
                          consumer_key="xxx"
Content-Type: application/json
```

Response:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "consumer_id": "xxx",
  "user_id": "xxx"
}
```

API Request:

```
GET /obp/v4.0.0/users/current
Authorization: DirectLogin token="eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
```

6.2 JWT Token Validation

Token Structure:

```
{
  "header": {
    "alg": "RS256",
    "typ": "JWT",
    "kid": "key-id"
  },
  "payload": {
    "iss": "http://localhost:9000/obp-oidc",
    "sub": "user-uuid",
    "aud": "obp-api-client",
    "exp": 1234567890,
    "iat": 1234567890,
    "email": "user@example.com",
    "name": "John Doe",
    "preferred_username": "johndoe"
  },
  "signature": "..."
}
```

Validation Process:

1. Extract JWT from Authorization header
2. Decode header to get `kid` (key ID)
3. Fetch public keys from JWKS endpoint
4. Verify signature using public key
5. Validate `iss` (issuer) matches configured issuers
6. Validate `exp` (expiration) is in future
7. Validate `aud` (audience) if required
8. Extract user identity from claims

JWKS Endpoint Response:

```
{
  "keys": [
    {
      "kty": "RSA",
      "use": "sig",
      "kid": "key-id-1",
      "n": "modulus...",
      "e": "AQAB"
    }
  ]
}
```

Troubleshooting JWT Issues:

Error: OBP-20208: Cannot match the issuer and JWKS URI

- Verify `oauth2.jwk_set.url` contains the correct JWKS endpoint
- Ensure issuer in JWT matches configured provider
- Check URL format consistency (HTTP vs HTTPS, trailing slashes)

Error: OBP-20209: Invalid JWT signature

- Verify JWKS endpoint is accessible
- Check that `kid` in JWT header matches available keys
- Ensure system time is synchronized (NTP)

Debug Logging:

```
<!-- In logback.xml -->
<logger name="code.api.OAuth2" level="DEBUG"/>
<logger name="code.api.util.JwtUtil" level="DEBUG"/>
```

6.3 Consumer Key Management

Creating a Consumer:

```
POST /management/consumers
Authorization: DirectLogin token="xxx"
Content-Type: application/json

{
  "app_name": "My Banking App",
  "app_type": "Web",
  "description": "Customer-facing web application",
  "developer_email": "dev@example.com",
  "redirect_url": "https://myapp.com/callback"
}
```

Response:

```
{
  "consumer_id": "xxx",
  "key": "consumer-key-xxx",
  "secret": "consumer-secret-xxx",
  "app_name": "My Banking App",
  "app_type": "Web",
  "description": "Customer-facing web application",
  "developer_email": "dev@example.com",
  "redirect_url": "https://myapp.com/callback",
  "created_by_user_id": "user-uuid",
  "created": "2024-01-01T00:00:00Z",
  "enabled": true
}
```

Managing Consumers:

```
# Get all consumers (requires CanGetConsumers role)
GET /management/consumers

# Get consumer by ID
GET /management/consumers/{CONSUMER_ID}

# Enable/Disable consumer
PUT /management/consumers/{CONSUMER_ID}
{
  "enabled": false
}

# Update consumer certificate (for MTLS)
PUT /management/consumers/{CONSUMER_ID}/consumer/certificate
```

6.4 SSL/TLS Configuration

6.4.1 SSL with PostgreSQL

Generate SSL Certificates:

```
# Create SSL directory
sudo mkdir -p /etc/postgresql/ssl
cd /etc/postgresql/ssl

# Generate private key
sudo openssl genrsa -out server.key 2048

# Generate certificate signing request
```

```
sudo openssl req -new -key server.key -out server.csr

# Self-sign certificate (or use CA-signed)
sudo openssl x509 -req -days 365 -in server.csr -signkey server.key -out
server.crt

# Set permissions
sudo chmod 600 server.key
sudo chown postgres:postgres server.key server.crt
```

PostgreSQL Configuration (`postgresql.conf`):

```
ssl = on
ssl_cert_file = '/etc/postgresql/ssl/server.crt'
ssl_key_file = '/etc/postgresql/ssl/server.key'
ssl_ca_file = '/etc/postgresql/ssl/ca.crt'  # Optional
ssl_prefer_server_ciphers = on
ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'
```

OBP-API Props:

```
db.url=jdbc:postgresql://localhost:5432/obpdb?
user=obp&password=xxx&ssl=true&sslmode=require
```

6.4.2 SSL Encryption with Props File

Generate Keystore:

```
# Generate keystore with key pair
keytool -genkeypair -alias obp-api \
  -keyalg RSA -keysize 2048 \
  -keystore /path/to/api.keystore.jks \
  -validity 365

# Export public certificate
keytool -export -alias obp-api \
  -keystore /path/to/api.keystore.jks \
  -rfc -file apipub.cert

# Extract public key
openssl x509 -pubkey -noout -in apipub.cert > public_key.pub
```

Encrypt Props Values:

```
#!/bin/bash
# encrypt_prop.sh
echo -n "$2" | openssl pkeyutl \
  -pkeyopt rsa_padding_mode:pkcs1 \
  -encrypt \
  -pubin \
  -inkey "$1" \
  -out >(base64)
```

Usage:

```
./encrypt_prop.sh /path/to/public_key.pub "my-secret-password"
# Outputs: BASE64_ENCODED_ENCRYPTED_VALUE
```

Props Configuration:

```
# Enable JWT encryption
jwt.use.ssl=true
keystore.path=/path/to/api.keystore.jks
keystore.alias=obp-api

# Encrypted property
db.password.is_encrypted=true
db.password=BASE64_ENCODED_ENCRYPTED_VALUE
```

6.4.3 Password Obfuscation (Jetty)

Generate Obfuscated Password:

```
java -cp /usr/share/jetty9/lib/jetty-util-*.jar \
  org.eclipse.jetty.util.security.Password \
### 12.5 Complete API Roles Reference

OBP-API uses a comprehensive role-based access control (RBAC) system with over
**334 static roles**. Roles control access to specific API endpoints and
operations.

**Note:** All roles can be dynamically listed using the `/obp/v5.1.0/roles`
endpoint.

**Last Updated:** 2025-10-29

#### Role Naming Convention

Roles follow a consistent naming pattern:
- `Can[Action][Resource][Scope]`
- **Action:** Create, Get, Update, Delete, Read, Add, Maintain, Search, Enable,
Disable, etc.
- **Resource:** Account, Customer, Bank, Transaction, Product, Card, Branch, ATM,
etc.
- **Scope:** AtOneBank, AtAnyBank, ForUser, etc.

#### Common Role Patterns

**System-Level Roles** (requiresBankId = false):
- Apply across all banks
- Examples: `CanGetAnyUser`, `CanCreateBank`, `CanReadMetrics`

**Bank-Level Roles** (requiresBankId = true):
- Scoped to a specific bank
- Examples: `CanCreateCustomer`, `CanCreateBranch`, `CanGetMetricsAtOneBank`

#### Key Role Categories

**Account Management:**
- CanCreateAccount
- CanUpdateAccount
- CanGetAccountsHeldAtOneBank
- CanGetAccountsHeldAtAnyBank
- CanCreateAccountAttributeAtOneBank
- CanUpdateAccountAttribute
- CanDeleteAccountCascade
- CanCreateAccountAttributeDefinitionAtOneBank
- CanDeleteAccountAttributeDefinitionAtOneBank
- CanGetAccountAttributeDefinitionAtOneBank
- CanUpdateAccountAttribute
- CanGetAccountApplications
- CanUpdateAccountApplications
- CanGetAccountsMinimalForCustomerAtAnyBank
- CanUseAccountFirehose
- CanUseAccountFirehoseAtAnyBank
- CanSeeAccountAccessForAnyUser

**Customer Management:**
- CanCreateCustomer
- CanCreateCustomerAtAnyBank
- CanGetCustomer
- CanGetCustomers
- CanGetCustomersAtAnyBank
- CanGetCustomersMinimal
- CanGetCustomersMinimalAtAnyBank
- CanGetCustomerOverview
- CanGetCustomerOverviewFlat
- CanUpdateCustomerEmail
- CanUpdateCustomerNumber
- CanUpdateCustomerMobilePhoneNumber
- CanUpdateCustomerIdentity
- CanUpdateCustomerBranch
- CanUpdateCustomerData
- CanUpdateCustomerCreditLimit
```

```
   - CanUpdateCustomerCreditRatingAndSource
   - CanUpdateCustomerCreditRatingAndSourceAtAnyBank
   - CanGetCorrelatedUsersInfo
   - CanGetCorrelatedUsersInfoAtAnyBank
   - CanCreateCustomerAccountLink
   - CanDeleteCustomerAccountLink
   - CanGetCustomerAccountLink
   - CanGetCustomerAccountLinks
   - CanUpdateCustomerAccountLink
   - CanCreateCustomerAttributeAtOneBank
   - CanCreateCustomerAttributeAtAnyBank
   - CanCreateCustomerAttributeDefinitionAtOneBank
   - CanGetCustomerAttributeAtOneBank
   - CanGetCustomerAttributeAtAnyBank
   - CanGetCustomerAttributesAtOneBank
   - CanGetCustomerAttributesAtAnyBank
   - CanGetCustomerAttributeDefinitionAtOneBank
   - CanUpdateCustomerAttributeAtOneBank
   - CanUpdateCustomerAttributeAtAnyBank
   - CanDeleteCustomerAttributeAtOneBank
   - CanDeleteCustomerAttributeAtAnyBank
   - CanDeleteCustomerAttributeDefinitionAtOneBank
   - CanCreateCustomerAddress
   - CanGetCustomerAddress
   - CanDeleteCustomerAddress
   - CanCreateCustomerMessage
   - CanGetCustomerMessages
   - CanDeleteCustomerCascade
   - CanUseCustomerFirehoseAtAnyBank

   **Transaction Management:**
   - CanCreateAnyTransactionRequest
   - CanGetTransactionRequestAtAnyBank
   - CanUpdateTransactionRequestStatusAtAnyBank
   - CanCreateTransactionAttributeAtOneBank
   - CanCreateTransactionAttributeDefinitionAtOneBank
   - CanGetTransactionAttributeAtOneBank
   - CanGetTransactionAttributesAtOneBank
   - CanGetTransactionAttributeDefinitionAtOneBank
   - CanUpdateTransactionAttributeAtOneBank
   - CanDeleteTransactionAttributeDefinitionAtOneBank
   - CanCreateTransactionRequestAttributeAtOneBank
   - CanCreateTransactionRequestAttributeDefinitionAtOneBank
   - CanGetTransactionRequestAttributeAtOneBank
   - CanGetTransactionRequestAttributesAtOneBank
   - CanGetTransactionRequestAttributeDefinitionAtOneBank
   - CanUpdateTransactionRequestAttributeAtOneBank
   - CanDeleteTransactionRequestAttributeDefinitionAtOneBank
   - CanCreateHistoricalTransaction
   - CanCreateHistoricalTransactionAtBank
   - CanDeleteTransactionCascade
   - CanCreateTransactionType
   - CanGetDoubleEntryTransactionAtOneBank
   - CanGetDoubleEntryTransactionAtAnyBank
```

```
**Bank Resource Management:**
- CanCreateBranch
- CanCreateBranchAtAnyBank
- CanUpdateBranch
- CanDeleteBranch
- CanDeleteBranchAtAnyBank
- CanCreateAtm
- CanCreateAtmAtAnyBank
- CanUpdateAtm
- CanUpdateAtmAtAnyBank
- CanDeleteAtm
- CanDeleteAtmAtAnyBank
- CanCreateAtmAttribute
- CanCreateAtmAttributeAtAnyBank
- CanGetAtmAttribute
- CanGetAtmAttributeAtAnyBank
- CanUpdateAtmAttribute
- CanUpdateAtmAttributeAtAnyBank
- CanDeleteAtmAttribute
- CanDeleteAtmAttributeAtAnyBank
- CanCreateFxRate
- CanCreateFxRateAtAnyBank
- CanReadFx
- CanDeleteBankCascade
- CanCreateBankAttribute
- CanGetBankAttribute
- CanUpdateBankAttribute
- CanDeleteBankAttribute
- CanCreateBankAttributeDefinitionAtOneBank
- CanCreateBankAccountBalance
- CanGetBankAccountBalance
- CanGetBankAccountBalances
- CanUpdateBankAccountBalance
- CanDeleteBankAccountBalance

**User & Entitlement Management:**
- CanCreateUserCustomerLink
- CanCreateUserCustomerLinkAtAnyBank
- CanGetUserCustomerLink
- CanGetUserCustomerLinkAtAnyBank
- CanDeleteUserCustomerLink
- CanDeleteUserCustomerLinkAtAnyBank
- CanCreateEntitlementAtOneBank
- CanCreateEntitlementAtAnyBank
- CanDeleteEntitlementAtOneBank
- CanDeleteEntitlementAtAnyBank
- CanGetEntitlementsForOneBank
- CanGetEntitlementsForAnyBank
- CanGetEntitlementsForAnyUserAtOneBank
- CanGetEntitlementsForAnyUserAtAnyBank
- CanGetEntitlementRequestsAtAnyBank
- CanDeleteEntitlementRequestsAtAnyBank
- CanCreateUserAuthContext
```

```
  - CanCreateUserAuthContextUpdate
  - CanGetUserAuthContext
  - CanDeleteUserAuthContext
  - CanCreateUserInvitation
  - CanGetUserInvitation
  - CanRefreshUser
  - CanSyncUser
  - CanReadUserLockedStatus
  - CanCreateResetPasswordUrl

  **Consumer & API Management:**
  - CanCreateConsumer
  - CanGetConsumers
  - CanEnableConsumers
  - CanDisableConsumers
  - CanUpdateConsumerName
  - CanUpdateConsumerRedirectUrl
  - CanUpdateConsumerLogoUrl
  - CanUpdateConsumerCertificate
  - CanSetCallLimits
  - CanReadCallLimits
  - CanDeleteRateLimiting
  - CanReadMetrics
  - CanGetMetricsAtOneBank
  - CanSearchMetrics
  - CanGetConfig
  - CanGetConnectorMetrics
  - CanGetAdapterInfo
  - CanGetAdapterInfoAtOneBank
  - CanGetDatabaseInfo
  - CanGetSystemIntegrity
  - CanGetCallContext

  **Dynamic Resources:**
  - CanCreateDynamicEndpoint
  - CanGetDynamicEndpoint
  - CanGetDynamicEndpoints
  - CanUpdateDynamicEndpoint
  - CanDeleteDynamicEndpoint
  - CanCreateBankLevelDynamicEndpoint
  - CanGetBankLevelDynamicEndpoint
  - CanGetBankLevelDynamicEndpoints
  - CanUpdateBankLevelDynamicEndpoint
  - CanDeleteBankLevelDynamicEndpoint
  - CanCreateSystemLevelDynamicEntity
  - CanGetSystemLevelDynamicEntities
  - CanUpdateSystemLevelDynamicEntity
  - CanDeleteSystemLevelDynamicEntity
  - CanCreateBankLevelDynamicEntity
  - CanGetBankLevelDynamicEntities
  - CanUpdateBankLevelDynamicEntity
  - CanDeleteBankLevelDynamicEntity
  - CanCreateDynamicResourceDoc
  - CanGetDynamicResourceDoc
```

```
- CanGetAllDynamicResourceDocs
- CanUpdateDynamicResourceDoc
- CanDeleteDynamicResourceDoc
- CanReadDynamicResourceDocsAtOneBank
- CanCreateBankLevelDynamicResourceDoc
- CanGetBankLevelDynamicResourceDoc
- CanGetAllBankLevelDynamicResourceDocs
- CanUpdateBankLevelDynamicResourceDoc
- CanDeleteBankLevelDynamicResourceDoc
- CanCreateDynamicMessageDoc
- CanGetDynamicMessageDoc
- CanGetAllDynamicMessageDocs
- CanUpdateDynamicMessageDoc
- CanDeleteDynamicMessageDoc
- CanCreateBankLevelDynamicMessageDoc
- CanGetBankLevelDynamicMessageDoc
- CanDeleteBankLevelDynamicMessageDoc
- CanCreateEndpointMapping
- CanGetEndpointMapping
- CanGetAllEndpointMappings
- CanUpdateEndpointMapping
- CanDeleteEndpointMapping
- CanCreateBankLevelEndpointMapping
- CanGetBankLevelEndpointMapping
- CanGetAllBankLevelEndpointMappings
- CanUpdateBankLevelEndpointMapping
- CanDeleteBankLevelEndpointMapping
- CanCreateMethodRouting
- CanGetMethodRoutings
- CanUpdateMethodRouting
- CanDeleteMethodRouting
- CanCreateConnectorMethod
- CanGetConnectorMethod
- CanGetAllConnectorMethods
- CanUpdateConnectorMethod
- CanGetConnectorEndpoint
- CanCreateSystemLevelEndpointTag
- CanGetSystemLevelEndpointTag
- CanUpdateSystemLevelEndpointTag
- CanDeleteSystemLevelEndpointTag
- CanCreateBankLevelEndpointTag
- CanGetBankLevelEndpointTag
- CanUpdateBankLevelEndpointTag
- CanDeleteBankLevelEndpointTag
- CanGetAllApiCollections
- CanGetApiCollectionsForUser
- CanReadResourceDoc
- CanReadStaticResourceDoc
- CanReadGlossary

**Consent Management:**
- CanGetConsentsAtOneBank
- CanGetConsentsAtAnyBank
- CanUpdateConsentStatusAtOneBank
- CanUpdateConsentStatusAtAnyBank
- CanUpdateConsentAccountAccessAtOneBank
- CanUpdateConsentAccountAccessAtAnyBank
- CanUpdateConsentUserAtOneBank
- CanUpdateConsentUserAtAnyBank
- CanRevokeConsentAtBank

**Security & Compliance:**
- CanAddKycCheck
- CanGetAnyKycChecks
- CanAddKycDocument
- CanGetAnyKycDocuments
- CanAddKycMedia
- CanGetAnyKycMedia
- CanAddKycStatus
- CanGetAnyKycStatuses
- CanCreateRegulatedEntity
- CanDeleteRegulatedEntity
- CanCreateRegulatedEntityAttribute
- CanGetRegulatedEntityAttribute
- CanGetRegulatedEntityAttributes
- CanUpdateRegulatedEntityAttribute
- CanDeleteRegulatedEntityAttribute
- CanCreateAuthenticationTypeValidation
- CanGetAuthenticationTypeValidation
- CanUpdateAuthenticationTypeValidation
- CanDeleteAuthenticationValidation
- CanCreateJsonSchemaValidation
- CanGetJsonSchemaValidation
- CanUpdateJsonSchemaValidation
- CanDeleteJsonSchemaValidation
- CanCreateTaxResidence
- CanGetTaxResidence
- CanDeleteTaxResidence

**Logging & Monitoring:**
- CanGetTraceLevelLogsAtOneBank
- CanGetTraceLevelLogsAtAllBanks
- CanGetDebugLevelLogsAtOneBank
- CanGetDebugLevelLogsAtAllBanks
- CanGetInfoLevelLogsAtOneBank
- CanGetInfoLevelLogsAtAllBanks
- CanGetWarningLevelLogsAtOneBank
- CanGetWarningLevelLogsAtAllBanks
- CanGetErrorLevelLogsAtOneBank
- CanGetErrorLevelLogsAtAllBanks
- CanGetAllLevelLogsAtOneBank
- CanGetAllLevelLogsAtAllBanks

**Views & Permissions:**
- CanCreateSystemView
- CanGetSystemView
- CanUpdateSystemView
- CanDeleteSystemView
- CanCreateSystemViewPermission
- CanDeleteSystemViewPermission

**Cards:**
- CanCreateCardsForBank
- CanGetCardsForBank
- CanUpdateCardsForBank
- CanDeleteCardsForBank
- CanCreateCardAttributeDefinitionAtOneBank
- CanGetCardAttributeDefinitionAtOneBank
- CanDeleteCardAttributeDefinitionAtOneBank

**Products & Fees:**
- CanCreateProduct
- CanCreateProductAtAnyBank
- CanCreateProductAttribute
- CanGetProductAttribute
- CanUpdateProductAttribute
- CanDeleteProductAttribute
- CanCreateProductAttributeDefinitionAtOneBank
- CanGetProductAttributeDefinitionAtOneBank
- CanDeleteProductAttributeDefinitionAtOneBank
- CanCreateProductFee
- CanGetProductFee
- CanUpdateProductFee
- CanDeleteProductFee
- CanDeleteProductCascade
- CanMaintainProductCollection

**Webhooks:**
- CanCreateWebhook
- CanGetWebhooks
- CanUpdateWebhook
- CanCreateSystemAccountNotificationWebhook
- CanCreateAccountNotificationWebhookAtOneBank

**Data Management:**
- CanCreateSandbox
- CanSearchWarehouse
- CanSearchWarehouseStatistics
- CanCreateDirectDebitAtOneBank
- CanCreateStandingOrderAtOneBank
- CanCreateCounterparty
- CanCreateCounterpartyAtAnyBank
- CanGetCounterparty
- CanGetCounterpartyAtAnyBank
- CanGetCounterparties
- CanGetCounterpartiesAtAnyBank
- CanDeleteCounterparty
- CanDeleteCounterpartyAtAnyBank
- CanAddSocialMediaHandle
- CanGetSocialMediaHandles
- CanUpdateAgentStatusAtOneBank
- CanUpdateAgentStatusAtAnyBank
```

Scopes:

- CanCreateScopeAtOneBank
- CanCreateScopeAtAnyBank
- CanDeleteScopeAtAnyBank

Web UI:

- CanCreateWebUiProps
- CanGetWebUiProps
- CanDeleteWebUiProps

Viewing All Roles

Via API:

```
GET /obp/v5.1.0/roles
Authorization: DirectLogin token="TOKEN"
```

Via Source Code: The complete list of roles is defined in:

- `obp-api/src/main/scala/code/api/util/ApiRole.scala`

Via API Explorer:

- Navigate to the "Role" endpoints section
- View role requirements for each endpoint in the documentation

Granting Roles

```
# Grant role to user at specific bank
POST /obp/v5.1.0/users/USER_ID/entitlements
{
  "bank_id": "gh.29.uk",
  "role_name": "CanCreateAccount"
}

# Grant system-level role (bank_id = "")
POST /obp/v5.1.0/users/USER_ID/entitlements
{
  "bank_id": "",
  "role_name": "CanGetAnyUser"
}
```

Special Roles

Super Admin Roles:

- `CanCreateEntitlementAtAnyBank` - Can grant any role at any bank
- `CanDeleteEntitlementAtAnyBank` - Can revoke any role at any bank

Firehose Roles:

- `CanUseAccountFirehoseAtAnyBank` - Access to all account data
- `CanUseCustomerFirehoseAtAnyBank` - Access to all customer data

Note: The complete list of 334 roles provides fine-grained access control for every operation in the OBP ecosystem. Roles can be combined to create custom permission sets tailored to specific use cases.

12.6 Roadmap and Future Development

OBP-API-II (Next Generation API)

Status: In Active Development

Overview: OBP-API-II is a leaner tech stack for future Open Bank Project API versions with less dependencies.

Purpose:

- **Aim:** Reduce the dependencies on Liftweb and Jetty.

Development Focus:

- Usage of OBP Scala Library

Migration Path:

- Use OBP Dispatch to route between endpoints served by OBP-API and OBP-API-II (both stacks return Resource Docs so dispatch can discover and route)

Repository:

- GitHub: `OBP-API-II` (development branch)

OBP-Dispatch (API Gateway/Proxy)

Status: Experimental/Beta

Overview: OBP-Dispatch is a lightweight proxy/gateway service designed to route requests to OBP-API or OBP-API-II or OBP-Trading instances.

Key Features:

- **Request Routing:** Routing based on Endpoint implementation.

Use Cases:

1. **API Version Management:**

    - Routing to new OBP implementations.

Architecture:

```
Client Request
     │
     ▼
┌────────────────┐
│  OBP-Dispatch  │
│    (Proxy)     │
└────────┬───────┘
         │
         │
    ┌────┼────┬────────┐
    │    │    │        │
    ▼    ▼    ▼        ▼
┌──────┐ ┌──────┐ ┌──────┐ ┌──────┐
│OBP-  │ │OBP-  │ │OBP-  │ │OBP-  │
│API 1 │ │API 2 │ │Trading │API N │
└──────┘ └──────┘ └──────┘ └──────┘
```

Deployment:

```bash
# Build
cd OBP-API-Dispatch
mvn clean package

# Run
java -jar target/OBP-API-Dispatch-1.0-SNAPSHOT-jar-with-dependencies.jar
```