Open Banking API Standards and Specifications

Central Bank of Oman

25 04 2024

Table of Contents

1. Introduction

Application Programming Interface (API) is a set of protocols and rules that enable different software applications to communicate to each other and thus enabling seamless flow of information among the applications. The APIs work based on the mechanism of request and response cycle. A request for data is sent by an application through API, which fetches the data and shares back from another application. Benefits of APIs include,

- Improved collaboration
- Monetization of data
- System security
- Cost reduction
- More scalability of the systems
- Fostering innovation

1.1 Open API

Open APIs are defined in various ways, the most common accepted definition is, Open API is publicly available API that provides developers with access for programming to software applications or webservices (mostly proprietary). The Open APIs are the backbone for enabling the Open banking. The Open banking APIs allows third party service providers (TPPs) to access the data from the financial institutions and use it to develop and provide innovation solutions to the end customers.

1.2 Challenges of Open APIs

Some of the major challenges of the Open APIs are,

- **Securit**y: API security is the topmost challenge the organizations are facing as APIs very vulnerable to the attacks and can lead to data leakages easily. Another challenge is there is no single tool that can track and detect the breach automatically. The organizations are advised to implement robust end point security at process, infrastructure, and protocol levels to mitigate the risks.
- **Data privacy:** This refers to unauthorized access to the customer’s personal data. Organizations should have a robust consent management system should be in place along with the complying to the data privacy laws.
- **Regulatory compliance:** Financial institutions and third-party service providers must navigate a complex web of regulatory requirements and standards, often varying by region. For example, in Europe, the General Data Protection Regulation (GDPR) places stringent demands on data privacy and consent, while the Revised Payment Services Directive (PSD2) governs payment services and data sharing. Compliance with these regulations and others is essential to ensure that customer data is handled in a legally sound and ethical manner.

2. Scope and Objectives

The objective of API standards and specification document provides guidelines to all the participants in open banking in Oman.

The scope of this document include introduction to the API specifications based on Open API standard (OAS) 3.1.0, architectural styles of Open API development, data formats requirements, version control and error handling requirements, testing requirements. Along with the above, the documentation also provides insights into API lifecycle management guidelines and API Governance mechanism.

This document should be used as a reference for developing and management of the Open Banking APIs. This document should be used in Oman jurisdiction only.

This document is not,

- User guide for API development and management
- It's objective is not to provide details of the design of the Open APIs, this is left to the participants on how they design, test, and distribute the APIs based on the use cases.
- These are not actual APIs and the associated documentation.

Further, these specifications should be read in conjunction with the all the existing guidelines and mandates published by the Central Bank of Oman (CBO). All ecosystem participants must ensure compliance with all the applicable laws and regulation particularly: -

1. CBO Fintech Regulatory Sandbox Framework
2. BM 1184 – Financial Consumer Protection Regulatory Framework
3. BM 1185 – Rules for Availing Cloud Services by Licensed Institutions
4. BM 1187 – Instructions under law on Combating Money Laundering and Terrorism Financing
5. National Committee on Combating Terrorism financing Decision No 1/2022
6. BM 1191 – Instructions on Digital Onboarding and Electronic KYC
7. BM 1194 – Cyber Security & Resilience Framework
8. CBO's API Policy
9. Any other circulars issued by the CBO

3. Key Entities in the Open API ecosystem

The key entities in the development, consumption and management of the Open APIs should define clear roles and responsibilities for seamless management of the APIs.

3.1 API Providers

The API providers are the entities that are producing and distributing the APIs and hence the underlying data for consumption. The API providers should adopt best practices of API development and management. This include but not limited to,

- **API Design and Documentation**: Create well-defined and documented APIs that adhere to Open Banking standards and specifications.
- **Security and Authentication**: Implement robust security measures, including authentication and authorization mechanisms, to protect sensitive customer data.
- **Compliance**: Ensure compliance with regulatory requirements, such as data protection and privacy laws, and adhere to open banking standards set by CBO.
- **Monitoring and Logging**: Continuously monitor API usage, track performance metrics, and maintain detailed logs for auditing and troubleshooting and submission of these to CBO on established frequency.
- **Rate Limiting and Throttling**: Implement rate limiting and throttling mechanisms to control API traffic and prevent abuse.
- **Versioning**: Manage API versions effectively to ensure backward compatibility and allow for smooth transitions.
- **Developer Support**: Provide developer support through documentation, forums, and developer portals to assist API consumers in integrating and using the APIs.
- **Incident Response**: Develop and implement an incident response plan to address security breaches or other issues promptly.
- **Data Governance**: Establish data governance policies to handle and protect customer data appropriately.
- **Feedback and Improvement**: Gather feedback from API consumers and continuously improve the APIs based on their needs and suggestions.

3.2 API Consumers

The API consumers are the Third-Party Service Providers, who consume the API distributed and exposed by the API providers in turn the data. Following are some of the practices that should be adopted by the consumers of the API.

- **Security**: Implement secure practices for handling API credentials and sensitive data obtained from the API provider.
- **Compliance**: Ensure compliance with open banking regulations and standards while using the APIs.
- **Monitoring and Analytics**: Monitor API usage, track key performance metrics, and analyse data to optimize usage and costs.
- **Error Handling**: Implement effective error handling and retry mechanisms to handle API failures gracefully.
- **Data Usage**: Use API data responsibly and ethically, respecting user privacy and consent.
- **Documentation Review**: Thoroughly review the API documentation provided by the API provider to understand how to use the API correctly.
- **API Version Management**: Stay updated on API versions and plan for updates or migrations when necessary.

- **Feedback**: Provide constructive feedback to the API provider regarding API functionality and improvements.

3.3 API gateway

The API gateway is the software system used to manage the Open APIs. It should be able to handle all the routing requests, composition, and translation of protocols between the API providers and the TPP applications. The participants, i.e. both TPPs and FIs, implementing the API gateways should have functionalities listed but not limited to,

- **API Gateway Configuration**: Configure and maintain the API gateway to handle API traffic efficiently, including security settings, routing, and load balancing.
- **Security**: Implement security features within the API gateway, such as DDoS protection, Web Application Firewall (WAF), and SSL/TLS termination.
- **Performance Optimization**: Ensure the API gateway is optimized for high availability and low-latency responses to meet API consumers' needs.
- **Analytics and Monitoring**: Provide robust monitoring and analytics tools for both API providers and consumers to track API usage and performance.
- **Rate Limiting and Throttling**: Implement rate limiting and throttling policies as per API provider requirements to prevent abuse.
- **Scalability**: Ensure that the API gateway can scale to handle increased API traffic as needed.
- **Documentation**: Offer documentation and support for configuring and using the API gateway effectively.
- **Compliance**: Support compliance with regulatory requirements by providing features for data protection and access control.
- **Developer Portal**: If applicable, offer a developer portal to help API providers and consumers manage their API interactions.
- **Incident Response**: Be prepared to respond to incidents related to the API gateway promptly and effectively.

3.4 Central Bank of Oman (CBO)

The Central Bank of Oman (CBO) plays an important role in regulating the Open APIs.

- **Provision of Sandbox environment:** CBO should facilitate the testing of the use cases and the Open APIs, by enabling a sandbox and provisioning the data for the testing to all the participants.
- **Audit of APIs:** The CBO may timely conduct audit of the APIs against the set standards and mechanism to ensure all the participants are adhering to the Open banking regulatory requirements.
- **Monitoring Open APIs related metrics:** CBO should monitor the metrics that are established related to API performance to ensure adherence to the KPIs.
- **Advisor to the participants:** CBO should actively be engaged with all the participants on Open banking and advise them on the latest developments in the open banking space, their thoughts and plans for open banking in Oman.

4. Open Banking API Specifications

The Open API Specification defines a standard, language-agnostic interface to HTTPS APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. This document takes reference of Open API Specification (OAS) standard version 3.1.0.
The participants of the open banking in Oman should refer this section of the document for the Open API specification and try to adhere to the standards established.
The document provides participants with the best practices and requirements to be followed in terms of

- Architecture styles
- API construct
- Data formats
- Error handling
- Testing requirements
- Authentication and security
- API Lifecycle management
- API Governance
- API metrics

4.1 Architecture styles

The Open APIs can be developed using different architectural styles. Based on the industry and global best practice for Open banking, the participants should use Representational State Transfer (REST) as default architectural style for Open API development across all the use cases and services. The participants should adopt global leading standards and practices for architecture and integration, and this should be submitted to CBO for review and approval.

4.2 API Construct

The construct of the Open API based on the Open API Specification (OAS) 3.1.0 is based on the object-oriented objects or arrays of objects that group related key-value pairs. The first set of brackets {} in an Open API document contains all the properties and is called the **"document object"**. While there is some flexibility, Open API documents must adhere to a basic structure. Some high-level sections are mandatory, while others are optional, allowing for variations in Open API specs across different APIs.

The mandatory objects in the structure of the Open APIs are,

Metadata

The metadata for the Open API should include the following.

- Version: The API should be versioned as per the established process by the participants.
- Title: The title should be unambiguous and should state what the API is intended for.
- Description: Details of the API should be included in this section.

Servers

The servers section specifies the API server and base URL. You can define one or several servers, such as production and sandbox.

Paths

The paths section defines individual endpoints (paths) in your API, and the HTTPS methods (operations) supported by these endpoints.

Components

This field in the Open API Specification is an object that contains reusable schemas for request bodies, response schemas, and security schemes. These schemas can be referenced throughout the spec using the reference tags, particularly in the path object.

Security

An object that declares the type of security scheme authorizing requests. A security object is defined globally or overridden by individual operations.

Tags

An object containing metadata that specifies the order in which You should display API resources in the documentation.

External Docs

An object that links to additional documentation, such as user guides.

There are other optional object items which can be used based on the use cases and requirements,

- Contact
- License
- Operation
- Media Type
- Example
- Call back.
- Link
- Discriminator
- OAuth Flows

4.2.1 Request response structure of the Open API

Request Header:

The following header attributes are required to be sent for almost all API calls and the body must have a valid JSON. API specific headers are documented along with respective APIs.

Header Key	**Value**	**Description**
Content-Type	application/JSON	Represents the format of the payload being provided in the request. This must be set to application/JSON, except for the endpoints that support Content-Type other than application/JSON
Authorization	Bearer <*your_access_token*>	Credentials (Generated from /token end point) to be provided to the Authorisation or Resource Server in Bearer

		Authentication Scheme. Required except for /token endpoint.
X-PG-IdempotencyKey	*<Unique_Key>*	Unique request identifier to support idempotency for **POST** Methods. *Must be less than 40 chars.*
X-PG-UserIPAddress	*<User_IP_Address>*	User's/PSU's IP address if they are currently logged in with your application.
X-PG-DeviceUserAgent	*<User-Agent_from user's_device>*	Indicates the user-agent from the device or the browser that the Customer/PSU is using.
X-PG-UserLoginTime	*<Your_User's login Time*	The time when the user/PSU last logged in with your application in ISO-8601 format (YYYY-MM-DDThh:mm:ssZ) ex: 2021-10-13T13:01:15Z

Request Body

If an operation sends a request body, use the request Body keyword to describe the body content and media type. The request body should have the following,

- Name
- Purpose
- Required by (TPP / AA / API PF)
- Description
- Conditions – Optional / Mandatory
- Data Type
- Limitations (if any)

Responses

For each operation, you can define status codes, such as 200 OK or 404 Not Found, and the response body schema. Schemas can be defined inline or referenced.

Response body should also have the response fields along with the following,

- Status
- Status message
- Error message (in case of any errors)

Example responses can also be identified for different content types:

- Success Response
- Status code 200 or 201

Error Response

Error Id.	Data Type	Description
error	string	Unique short error code, enough to understand the error as well as use it in code. New error codes may be added as we introduce new features and enhance functionalities.

Error Message	string	Descriptive error message that helps debugging and understand the cause for failure. This can change over time and is not safe to use in code.
details	Object [] nullable	An array of error details providing more details around error (like field specific level).
Field Name	string nullable	The actual field name which failed validation or for which the error was thrown.
message	string nullable	Descriptive error message specifying the reason for failure for the field.
Trace Id	string	Unique trace Id that must be shared with payment gateway support team / IT team to help debug when you face any issue.

4.2.2 Data Types

Following are the data types that are generally be used in the open banking API construct, commonly used data types are,

Data Type	Description	Examples
Integer(num)	Numeric data type for numbers without fractions	-707, 0, 707
Floating Point (float)	Numeric data type for numbers with fractions	707.07, 0.7, 707.00
Character (char)	Single letter, digit, punctuation mark, symbol, or blank space	a, 1, !
String (str or text)	Sequence of characters, digits, or symbols—always treated as text	hello, +1-999-666-3333
Boolean (bool)	True or false values	0 (false), 1 (true)
Enumerated type (enum)	Small set of predefined unique values (elements or enumerators) that can be text-based or numerical	rock (0), jazz (1)
Array	List with a few elements in a specific order—typically of the same type	rock (0), jazz (1), blues (2), pop (3)
Date	Date in the YYYY-MM-DD format (ISO 8601 syntax)	28-09-2021
Time	Time in the hh:mm:ss format for the time of day, time since an event, or time interval between events	12:00:59
Datetime	Date and time together in the YYYY-MM-DD hh:mm:ss format	28-09-2021 12:00

Note: For more details on the data standards and specifications, please refer: Data Standards and specifications document.

4.2.3 HTTPS Methods

The HTTPS methods are the keywords for directing the actions for an API to perform. The developers should adhere to correct HTTPS methods.

GET

The GET request method is used to retrieve data from a specified resource, identified by a URL. It's a safe and idempotent operation, meaning it should not modify data on the server.

Example: A customer or a third-party financial app can send a GET request to the bank's API, specifying the account ID. The API responds with the transaction history for that account, allowing the customer to view their past transactions.

POST

POST is used to submit data to be processed to a specified resource. Unlike GET, it can change the server's state and create new resources. It's often used for creating new records or submitting forms.

Example: A customer wants to transfer money from their checking account at Bank A to their savings account at Bank B. The customer's financial app sends a POST request to Bank A's API with the necessary details (amount, recipient account, etc.), initiating the transfer.

PUT

PUT is used to update a resource or create one if it doesn't exist at a specific URL. It completely replaces the existing resource with the new data provided in the request.

Example: A customer wishes to change their notification preferences for transaction alerts. They send a PUT request to the Open Banking provider's API, updating their notification settings to receive alerts via email instead of SMS.

PATCH

PATCH is like PUT but is used to apply partial modifications to a resource. It only updates the specified fields, leaving the rest of the resource intact.

Example: a customer may want to modify their transaction description or add additional information to a transaction record. Using a PATCH request, the customer's banking app sends only the changes to the bank's API, like a new description or tag for a specific transaction, without resending the entire transaction data.

DELETE

DELETE is used to remove a resource from the server at a specified URL. It instructs the server to delete the resource permanently.

Example: If a customer wants to close a bank account in Open Banking, they can initiate the account closure by sending a DELETE request. The request includes the account number, and upon validation, the bank’s API will permanently remove the account from its records.

There are some other methods which intended for certain specific purposes.

HEAD

HEAD is like GET, but it requests only the headers of the resource, not the actual content. It's useful for checking resource metadata or verifying resource existence without downloading the full content.

Example: When a third-party financial application, like a budgeting tool, wants to check if a bank's API is accessible and if it supports certain features without retrieving account data, it can send a HEAD request. The response will contain metadata indicating the API's availability and capabilities.

OPTIONS

OPTIONS is used to request information about the communication options for a resource. It helps a client understand which HTTPS methods and headers are allowed for that resource, enabling safe and efficient interactions.

Example: Before establishing a connection with a bank's API, a developer working on a fintech application can send an OPTIONS request to the API endpoint. This helps the developer understand which authentication methods (e.g., OAuth 2.0) and HTTPS methods (e.g., GET, POST) are supported by the bank's API, ensuring secure and compliant interactions.

TRACE

TRACE is used to perform a diagnostic test by echoing back the received request to the client. It's rarely used in production but can be valuable for debugging and understanding how a request is processed by intermediary servers.

Example: During the development and testing of an Open Banking API integration, a developer may use TRACE requests to trace the journey of their HTTPS requests as they pass through intermediary servers and proxies. This can assist in diagnosing any unexpected modifications or issues in the request/response flow.

CONNECT

CONNECT is used to establish a network connection to a resource, typically through a proxy server. It's crucial for secure connections, such as HTTPS, where it helps establish a secure tunnel for encrypted communication.

Example: Especially for secure communication, a CONNECT request can be used to establish a secure tunnel through a proxy server for encrypted data transmission. This is crucial for ensuring the confidentiality and integrity of sensitive financial data, such as when accessing account details via HTTPS.

4.2.4 Open API Testing requirements

Open API testing should be carried out thoroughly by all the participants prior to commercializing the APIs. All the participants should follow testing best practices like Test driven development. The participants should also try to automate the testing process and use tools like Postman, Swagger etc. for automation. Open API testing should be carried out for evaluating Data consistency, API security and API performance. Clear metrics should be established for various types of testing and should be shared with the CBO.

Open API testing requirements prescribed but not limited to are,

API Documentation

The documentation provides guidelines of the functionalities that are being tested. It should also provide details of the available endpoints, expected request-response and error codes. The development team should make sure to document comprehensively for ease of testing.

Endpoint URLs

These are the specific URLs to which the request are directed to. Testing should cover all available endpoint URLs for comprehensive testing.

HTTPS method

All the types of HTTPS methods should be tested for intended actions. Testing these would also ensure efficiency of the APIs being tested.

Input Payload

When sending a request to an API, often, specific data or parameters need to be included, known as the input payload. This data can vary in structure and format, depending on the API's design. API Testers must be aware of the expected input payloads for different endpoints to ensure that the API can process the provided data correctly and return the desired response.

Expected Output

Once a request is made to an API, it responds with data, known as the output. This output can come in various formats, such as JSON or XML. The tester should test the expected output format to verify that the API is returning data in the correct structure and that the data itself is accurate and relevant.

The APIs should undergo different testing to fully comply with the requirements.

- Validation testing
- Integration testing
- Functional test
- Load testing
- Penetration testing
- Security testing
- Fuzzy testing

This phase should also include the comprehensive assessment for the security vulnerabilities. The participants should adhere to the cybersecurity guidelines provided in Workstream 7 i.e. Information security framework document (section 3.1.2, 3.1.4,) for API security readiness. Some of the API security areas to be taken care are,

- Excessive Data exposure
- Authentication and authorization
- Consent management
- Coding hygiene
- Proper access mechanisms
- Documentations

- Inadequate monitoring practices

4.2.5 Open API Authentication and security requirements

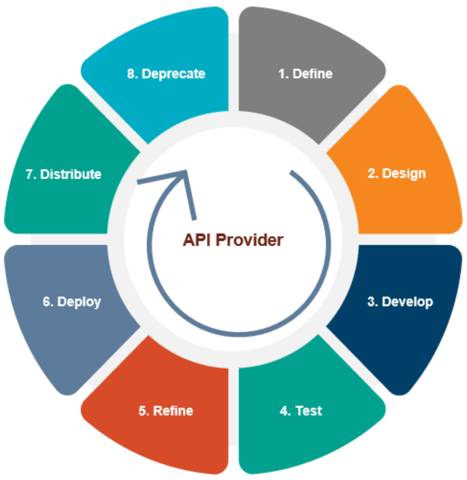
Please refer to the Open banking security standards documentation.

4.3 API Lifecycle Management

An Application Programming Interface (API) lifecycle defines various stages in the management of the APIs. The Open API lifecycles varies for the generator and provider of the API service and to the consumer of the API. Open APIs are provided by the ASPSPs like banks and financial institutions (FIs) who enable data provisioning to the Third-party service providers like FinTech. All the stakeholders should be aware and follow the lifecycle management to ensure standardization and ease integration amongst various stakeholders.

4.3.1 API Provider Lifecycle

The Provider (producer and distributor) lifecycle is divided into 8 phases. The provider should follow these phases to ensure the APIs are developed, distributed, and managed effectively and efficiently.



Phase 1: DEFINE

The precursor to this step is the use case and clear business requirements. Once, the business requirements are clear, then the providers of the Open APIs that would enable the functionality should define the functionalities of each of the APIs that would be serving the business requirement. At this stage, API governance framework also should be set up including owners, platforms, and tools for developers ex: GitHub. This phase should be diligently carried out to build effective APIs.

Phase 2: DESIGN

This phase involves the design of the API itself including how an API would deliver the functionality expected by the end user. It includes adherence to the API design principles, API governance principles and defines the mechanism to fetch and expose the data to the integrated systems. Some of the best practices for a good API design entail:

- Understanding the use case and functionality in depth
- Defining the API specifications
- Listing down all the assumptions
- Documenting the API along with the specifications.

A well-designed API should provide standardization of API design patterns like following the Open API specifications standard, naming conventions, capitalization, and punctuation. There should be a proper workflow designed for the functionality that is being fulfilled using the APIs.

Phase 3: DEVELOP

This is the stage where coding of the API happens. Developers should collaborate for development and use source code repositories Ex: GitHub, CVS or Apache SVN to keep track of issues and feedback/review of the codes. There should be an appropriate version control tool in place for code change management. It is recommended to use DevOps as default practice for development, testing and deployment of the codes in a secure manner.

Phase 4: TESTING

API testing is a critical step in checking the functionality of the APIs. It is recommended to use automation wherever possible to test the APIs. Test design, Test cases and acceptance criteria must be developed in the design phase itself, which can be traced during the testing phase

Please refer to the testing requirements in section 4.2.4

Phase 5: REFINE

This phase involves refinement of the Open APIs based on the testing outcome. The APIs should be modified to accommodate the test results and fix all the bugs / vulnerabilities before retesting.

Phase 6: DEPLOY

In this phase, the APIs are published in the development, testing and production environment. Before deployment the developers should make sure that all the required test cases are passed. If there are any exceptions, they should be recorded and released as part of the release note for the requirements. There should be a service / Open API catalogue to be developed by the providers of the API.

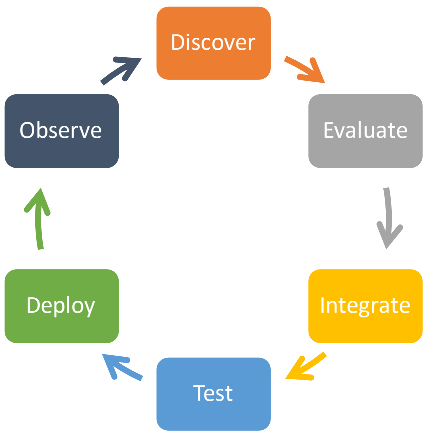
Phase 7: DISTRIBUTE

In this phase, the developers should publish the tested APIs on the portals. The Open API catalogue should include details such as APIs being offered, tags, use cases they serve etc. Once the APIs are distributed, the TPPs can discover the APIs, establish, and contract with the API providers and start consuming the same.

Phase 8: DEPRECATE
All the Open APIs that are no longer used within the ecosystem should be archived and deprecated. Before deprecating the APIs, the provider should make sure that the existing functionalities are not hampered with the API's deprecation. The unused APIs should be archived immediately and deprecated within 6 months from the date of archiving.

4.3.2 API Consumer Lifecycle

Consumer of the Open APIs from the providers are the TPPs that are providing innovative services to the end customers. The consumers should first identify the APIs that are required for their use cases. Based on the contract with the API providers, they need to establish a initial process for connecting to the provider's API portal and the end point.



Phase 1: DISCOVER

In the initial phase of the API consumer lifecycle, the primary focus should be on discovering the appropriate external APIs that align with the project's objectives. This involves a comprehensive evaluation of the project's requirements and identifying potential areas where external APIs could add value. Extensive research is conducted to explore the available APIs that are relevant to the project's domain. This research process entails delving into the API landscape, studying documentation, and gaining insights into the capabilities and features offered by different APIs. The goal is to gain a thorough understanding of how each API operates and whether it is suitable for integration into the project.

Phase 2: EVALUATE

Once a set of potential APIs is identified, the evaluation phase comes into play. In this phase the consumers should perform a meticulous assessment of each API against a range of criteria. These criteria include factors such as the functionality provided by the API, its security measures, scalability options, and associated costs. Furthermore, the compatibility of the APIs with the existing technical architecture of the project is carefully analysed. In this phase, a structured evaluation process helps in determining which APIs align most effectively with the project's strategic goals and technological requirements.

Phase 3: INTEGRATE

The participants should develop a integration strategy outlining the seamless integration of the identified Open APIs into the overall ecosystem. The consumer should connect to the

Provider's portal and end point to gain access to the Open APIs. The strategy should also encompass designing the integration architecture, establishing data flows between the project and the APIs, and creating the necessary code structures and configurations to enable robust connectivity.

Phase 4: TEST

Rigorous testing should be conducted to ensure that the integrated APIs function flawlessly in various scenarios. Comprehensive test cases should be formulated to cover a wide range of potential interactions.

Various testing to be covered are unit testing to validate the individual components, integration testing to verify the API interactions, and end-to-end testing to assess the overall functionality. The testing phase also addresses scenarios involving edge cases, error handling, and exception management to ensure the API integrations are resilient and reliable.

Phase 5: DEPLOY

Once the APIs pass the testing phase, they are prepared for deployment to production environments. This phase involves a meticulous transition of the API integrations from development to production settings. All necessary security measures are put in place, including encryption protocols and robust authentication mechanisms, to ensure the protection of sensitive data. The deployment process adheres to established protocols and follows best practices to ensure a seamless and secure transition of the API integrations into the live environment.

Phase 6: OBSERVE

Following the deployment, the observation phase must be initiated. This phase revolves around continuous monitoring and analysis of the integrated APIs in real-world usage. Monitoring tools and logging mechanisms are set up to track various metrics such as API usage patterns, response times, and error rates. This ongoing observation provides valuable insights into the performance of the APIs and helps identify any potential issues that may arise during actual usage. The insights gathered during this phase are used to optimize the API integrations, enhance their performance, and swiftly address any emerging challenges.

5. API Governance

Governing open banking APIs within the confines of an open banking ecosystem is a multifaceted task that demands the establishment of a comprehensive framework. This framework serves the critical purposes of maintaining security, ensuring compliance with regulations, and fostering interoperability among the various entities that participate in this complex financial landscape. Some of the important activities involved in effectively governing open banking APIs within the open banking paradigm.

Clear roles and responsibilities should be developed for various participants of the open banking ecosystem (detailed as in section 3 of the document). Along with these, appropriate governing bodies should be established within the organization to enable efficient and effective open banking adoption. These bodies would be responsible for taking decisions related to use cases, Open APIs design, Testing etc.

- **Open Banking Steering Committee**

  This steering committee should be an apex body for taking all the decision related to the Open banking implementation. Ideal constitution of the committee should be a mix of business, technical and support functions of the institutions along with the representation from CBO. Business members plays a crucial role in overseeing and shaping the strategic direction of the initiative and would be responsible for defining the business objectives, policies, and compliance standards for open banking APIs. It ensures that the APIs align with the broader business goals and meet regulatory requirements. The roles to be included in the Business Governance typically consist of representatives from various business units, legal experts, compliance officers, and senior management who can provide insights into the business aspects of open banking.

Technical members focus on the technical aspects of open banking API governance and would be responsible for solutioning of the use cases and establishing the technical standards, architecture, and security protocols for the APIs. It oversees the development and maintenance of the API infrastructure, ensuring that it remains secure, scalable, and interoperable. The roles within the technical governance often include software architects, security experts, developers, and IT infrastructure specialists who can guide the technical implementation of open banking APIs. Collaboration between business and technical members is essential to achieve a successful and compliant open banking ecosystem.

Along with the governing body, there should be a proper governance mechanism should be set up. This should include,

- **Dispute resolution committee**
  There should be dispute resolution committee set up within the organization encompassing members from technical team and business owners to resolve any dispute that may arise internally and with the external partners. If the dispute cannot be resolved mutually, then the team should apprise CBO of the dispute and potential impact.

6. Open API Metrics

It is recommended that both API providers and consumers should constantly monitor the API performance. The monitoring should be classified into qualitative and quantitative metrics.

The API Providers should develop a detailed KPIs and dashboard for monitoring the performance to the APIs. Some of the key things to be considered are,

- Monitor the KPIs regularly and publish the results on a periodic basis to CBO through reports. Any critical anomaly should be reported immediately.
- Record individual API performance metrics with at most 10-minute intervals between data points. Any interval missed is assumed to be downtime except within a period coinciding with a scheduled change or maintenance activity.
- All the KPIs must be defined at the transactional level without need of token, keys etc. for ease of use.

Some of the technical metrics (not limited to) for consideration includes,

- Uptime > 99.9%
- Latency should be as close to Zero.
- Errors (per minute)
- Requests (per minute)
- API calls (per minute)
- API call drops (per minute)
- API Retention

- Top consumers per API
- Unique API consumers.

Indicative levels of performance expected by the API providers and the consumers,

- Availability > 98.5%
- API Processing time < 3 secs
- API success rate > 97%

All the stakeholders are expected to develop incident response plan for mitigating any risks. Indicative response time for various incidents should be classified as operational, system related and performance related.

All the parties involved in the production and consumption of the Open APIs should make use of a certified and licensed API monitoring tools.

7. Glossary

- **AISP** – Account Information Service Provider. AISPs are entities that are authorized to access financial information from different banks and financial institutions with the user's consent.
- **API** – Application Programming Interface
- **ASPSP** – Account Servicing Payment Service Provider. ASPSPs are entities that provide and maintain payment accounts for customers. These accounts are often accessed by third-party providers (TPPs), such as AISPs (Account Information Service Providers) and PISPs (Payment Initiation Service Providers), through open banking APIs (Application Programming Interfaces) with the customer's consent. ASPSPs are the banks and financial institutions. The ASPSPs and banks and financial institutions are used interchangeably. In Workstream 4 i.e. Open banking regulatory framework, the terms referred to ASPSPs are banks and financial institutions.
- **BAuth** – Basic Authentication
- **BFSI** – Banking, Financial Services and Insurance
- **CI/CD** – Continuous Integration and Continuous Deployment
- **DDoS** – Distributed Denial of Service
- **DevOps** – Development Operations
- **DevSecOps** – Development, Security and Operations
- **FI** – Financial Institutions
- **HTTPS** – Hypertext Transfer Protocol - Secured
- **JSON** – JavaScript Object Notation
- **JWT** – JSON Web Tokens
- **KPI** – Key Performance Indicator
- **OAS** – Open API Specifications
- **PISP** – Payment Initiation Service Provider. PISPs are entities that facilitate online payments on behalf of consumers or businesses by initiating transactions directly from their bank accounts. These transactions are often initiated through open banking APIs (Application Programming Interfaces) with the user's consent.
- **TPP** – Third Party Provider
- **UI** – User Interface
- **URL** – Uniform Resource Locator
- **YAML** – Yet Another Markup Language
- **XML** – Extendible Markup Language