BLANC LABS

Open Banking Implementation Demystified:

Technical Architecture and Best Practices for Enabling Financial Data Portability

Table Of Contents

Abstract 03

What is Open Banking? 04

- Screen Scraping 05
- Advantages of Open Banking for Customers 07
- Open Banking Implications for Banks and Regulators 08
- FDX API Standards from The Financial Data Exchange 09
- OB Open API Specs 10

Technical Architecture 11

- System Building Blocks 13
  - API Gateway 13
  - Data Grid Layer 13
  - Identity and Consent Manager 14
  - Identity Provider 14
  - Analytics Engine 14
  - API Portal 14
  - Enterprise API Manager 15
  - Logs Aggregation Engine 15
  - OB Optimized Microservices 15
  - Downstream Systems 16
- Cloud Infrastructure Design 17

Open Banking Security 18

Best Practices and Recommendations 19

Conclusion 22

- What's next? 22

About Us 23

Abstract

Over the last few years, there has been a revolution in the way that banks and FinTech's interact with each other. This revolution has resulted in better products, experiences, and services for consumers by enabling tools and mechanisms to exchange financial data securely by leveraging modern industry standards.

This movement towards sharing data is **Open Banking** and presents us with a new operating model where **Open APIs** are used to establish interoperable systems that will allow for innovative financial products.

For any bank, enabling APIs to expose the data of its customers to external parties will always be a challenge that needs to be solved using an engineering approach. When it comes to enabling Open Banking, the required solution is nothing more than a cross-organization **API Integration** project that prioritizes activities and outcomes such as:

1. Conducting a comprehensive review of existing APIs
2. Categorizing/ rationalizing based on capabilities
3. Establishing a governance framework and actively managing the API lifecycle in order to deliver digital services at scale

These activities can be implemented depending on the situation and characteristics of the organization. Although various departments will contribute to this initiative, IT and Architecture teams most often take the bulk of the responsibility. A common theme that arises is whether there is any existing reference architecture or concrete guidelines that can be used to succeed in an Open Banking implementation initiative. Our goal with this whitepaper is to provide technical teams with a practical guide to implementing a successful Open Banking strategy.

In this whitepaper, which is one of a series, we describe the process of defining what **Open Banking** is and a recommended way to address its technical implementation by explaining a high-level solution architecture that is organization agnostic. We will also explore best practices that will help you get valuable insights on your journey to successfully implementing Open Banking.

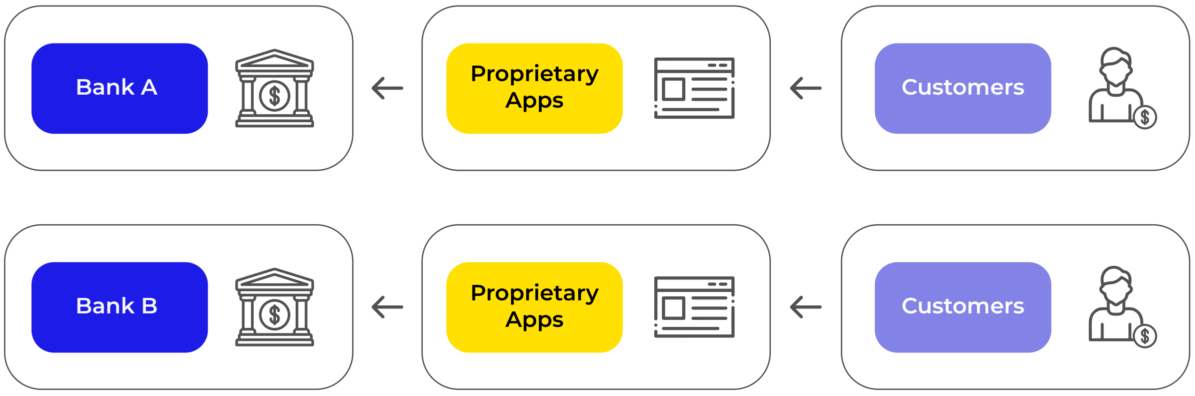
What is Open Banking?

Open Banking meets the modern demands of financial technology by unifying technical infrastructure and regulatory standards. It facilitates an innovative landscape for fintech companies to provide exciting new products that satisfy consumer needs. All these new products and services are exposed through API ecosystems, allowing third parties to integrate and provide a better customer experience. Simply put, Open Banking is “the evolution of the kind of API-based integrations that organizations have been building for decades.”[1]

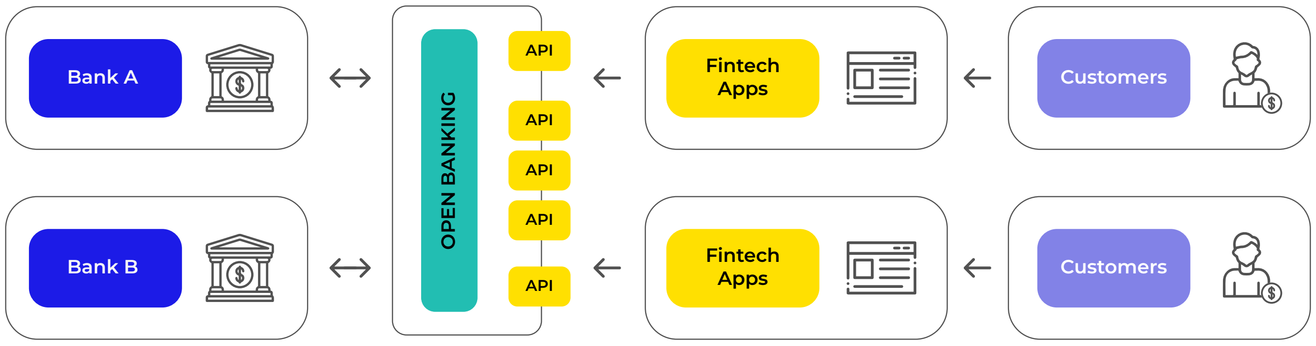
Traditional banking systems lack the capacity to provide users with holistic insight into their financial products across different banks, resulting in issues and complications when customers try to make informed decisions. In this digital age, it is increasingly important for banks to offer solutions which break down data silos and give an overview of a customer’s full financial profile.

With an emphasis on customer data centricity, financial institutions are seeing the benefit of offering customers the flexibility to securely move their information from one institution to another. This will ultimately give customers better control over their financial data. For FinTechs and banks, it will mean new financial products and opportunities for revenue generation.

[1]Eyal Syvan, Axway’s Mr Open Banking



Traditional Banking Operation Model



Open Banking Approach

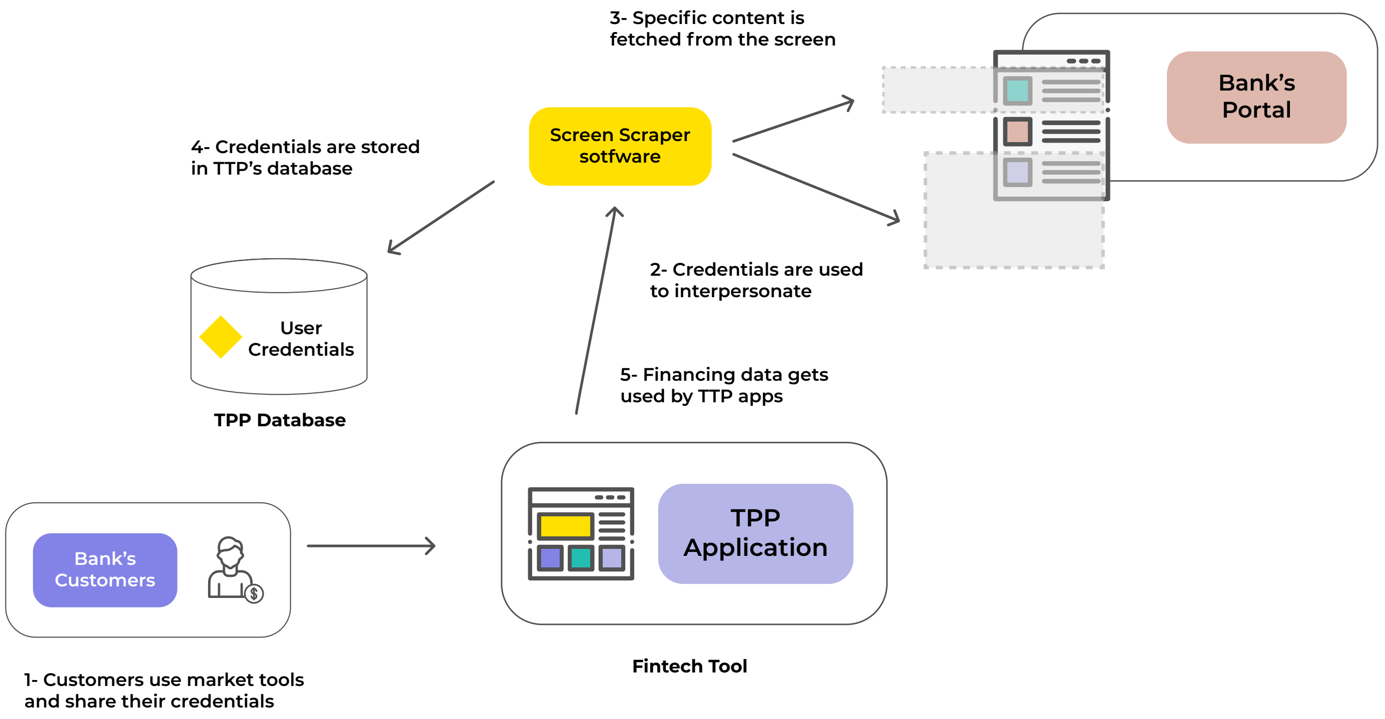
While Open Banking presents new opportunities for Fintechs and financial institutions, it is not the first mechanism that the financial industry has put in place to extract or fetch financial information to be used by third party apps (TPPs). For many years, TPPs have been developing applications that rely on a technique called **screen scraping**—a problematic and potentially risky way to extract data.

Screen Scraping

Screen scraping allows TPPs (third-party providers) to have access to a customer's banking information through the user's credentials (login and password) which are stored in a database. The apps developed by the TPP impersonate the actual customer and fetch as much data as they need by inspecting the HTML content from the bank's portals. This leads to a variety of security issues:

- Regardless of the specific data elements required by the TPP app (say the app just needs your saving account balance), in practice they have access to everything an actual user would after logging in (mortgages, credit card, loans, risks analysis, etc.)
- If for some reason your credentials get stolen by an attacker, they will have the ability to make any transaction, just like the user would
- Once the credentials are shared, users lose control over what providers can see and for how long
- Although the TPP tries to protect customer data, the truth is that sharing credentials (especially the ones that have access to your money) will always make the user vulnerable
- Screen scraping may also be a potential violation of the bank's terms and conditions (many banks have specific conditions on the expected usage of their products, especially when it comes to inter-personal user accounts)

It is important to highlight that the problem here is not the TPPs but the way that financial data is obtained. Third-party applications themselves must deal with multiple challenges including the usage of proper means to keep credentials secured and keeping up with frequent changes by the banks' portals (let's recall that Screen Scraping relies on fixed data elements present in a web page). In this way, screen scraping is both hard to maintain and open to potential risks.


3- Specific content is fetched from the screen
Bank's Portal
4- Credentials are stored in TTP's database
Screen Scraper sotfware
2- Credentials are used to interpersonate
User Credentials
TPP Database
5- Financing data gets used by TTP apps
Bank's Customers
TPP Application
Fintech Tool
1- Customers use market tools and share their credentials

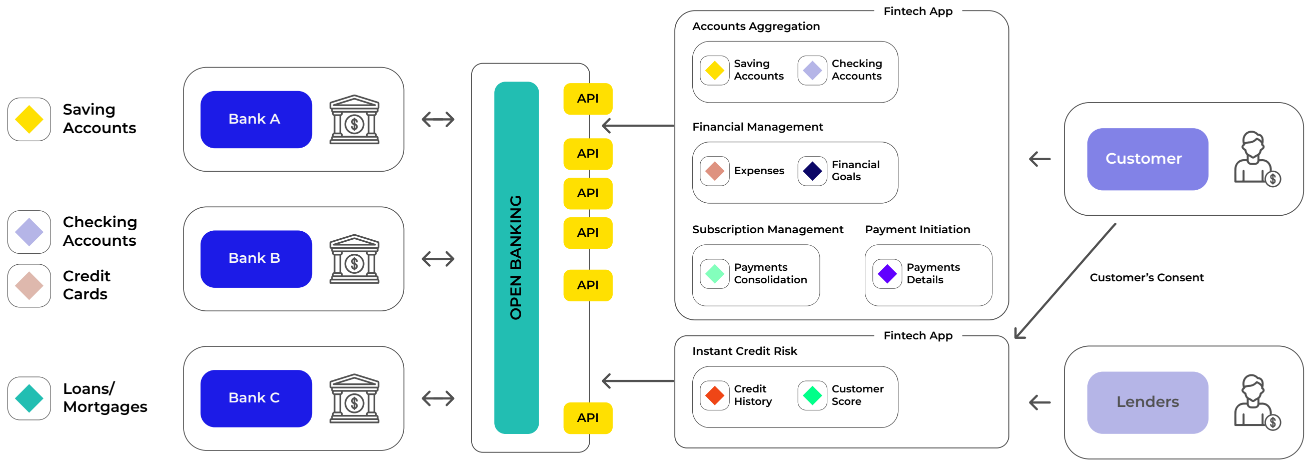
Screen Scraping Operation Model

This is where Open Banking differs from screen scraping. By promoting the development of secure and regulated APIs to make consumer's financial data available, banks and FinTechs are working with regulatory bodies to create standards (communication protocols) to link banks with third-party apps. This movement is clearly establishing a financial network that can securely exchange information.

Advantages of Open Banking for Customers

Banking and FinTech providers' use of customer data has enabled a more efficient, interconnected process. But what does this mean to the consumer? Through smarter data usage, customers have access to new applications that promise convenience, security and control over their financial experience like never before:

- **Account aggregation:** Allows customers to get an overview of their various accounts across different banks.
- **Personal Finance Management:** Provides the customers with a complete overview of their financial situation (including a monthly spending budget).
- **Instant Credit Risks:** Allows lenders to gain an almost instantaneous overview of an applicant's credit history (especially useful to speed-up credit applications).
- **Subscription Management:** Detects all the recurring payments from the customer and shows them in one interface.
- **Taxes Preparation:** Banks can communicate with accounting platforms to retrieve tax data, which will allow them to realize operational efficiencies to reduce call volumes during the tax season.
- **Payment Initiation:** Think of this as a simplification of the payment experience, where there will be multiple payment options within the same payment interface; additionally, payments will be processed in minutes or hours versus days.
- **And more...**

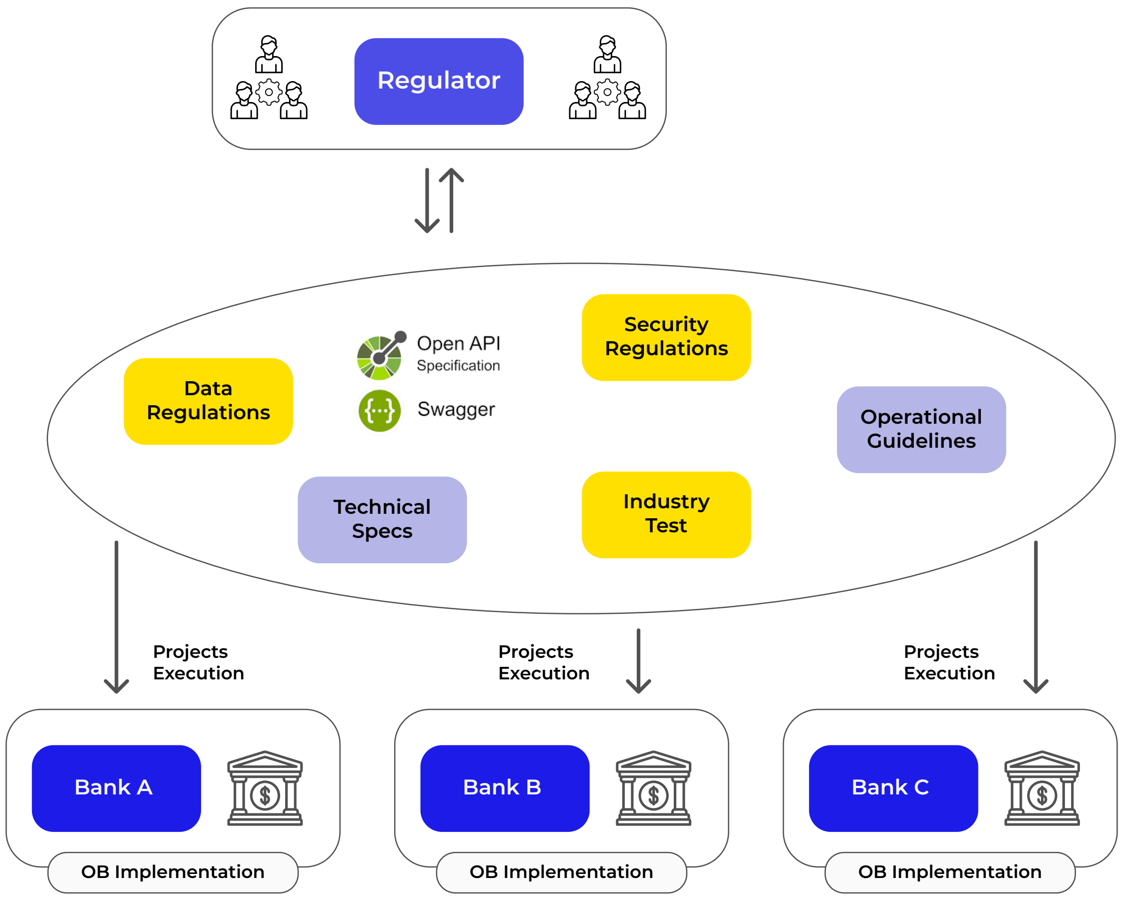


Open Banking Opportunities

It's really important to emphasize that Open Banking's intent is not simply exposing our financial information to a provider to access it. Any action conducted around a customer's data (such as bank balance, transaction data, payment initiation, etc.) can only take place with their specific consent and will ultimately be strictly regulated by the corresponding financial authority of each country.

Open Banking Implications for Banks and Regulators

By participating in this innovative Open Banking initiative, banks can acquire new customers more seamlessly. Thanks to the available transactional and statement data, they will have a better understanding of consumer financing behaviors (e.g., risk evaluation and long-term value). Meanwhile, regulators must confront numerous challenges such as security regulations for trustworthiness. Many countries are still formulating their Open Banking regulations and each nation will have its own version of the rules as there is a lack of a global standard.



Open Banking Regulatory Model

From a technical standpoint, banks are expected to provide the development of Open Banking APIs that follow the relevant Open API Specs. These specifications define the message structure of the data to be exchanged, including elements such as HTTP headers, Methods, URIs, Practices, etc., and the means to expose them using the right infrastructure so that services can be offered in a high available and secure way. Let us dive deeper into this in the following sections.

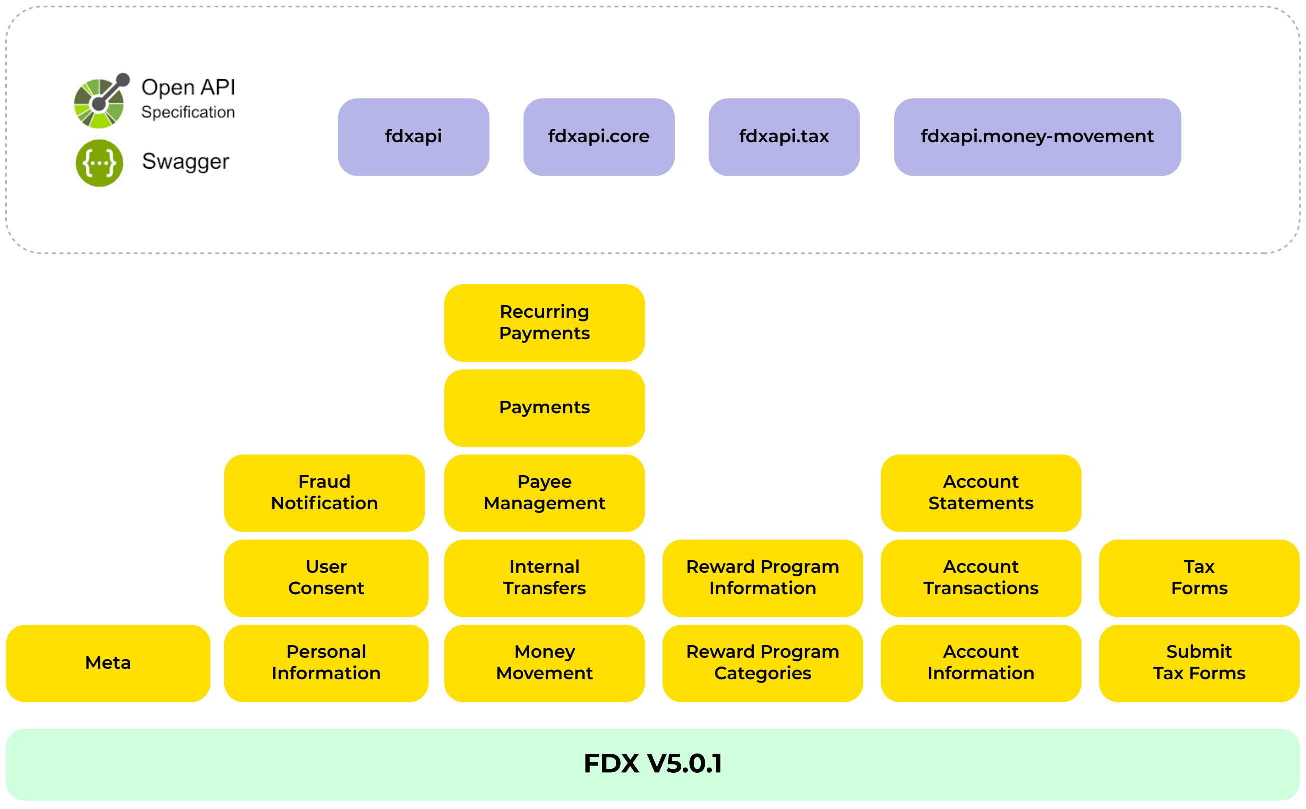
FDX API Standards from The Financial Data Exchange

FDX (Financial Data Exchange) is a non-profit industry standards body operating in the US and Canada whose purpose is unifying the financial services ecosystem around a common, interoperable set of technical standards for user-permissioned financial data sharing. Over the last few years this organization has quickly establishing itself as the de facto North American Open Banking standard.
Today, FDX is used to share over 65 Million consumer records across the North American financial ecosystem, and their membership includes most major US banks and FinTech companies. From the FDX implementer standpoint, the most important step is to get the technical certification as this is the means to demonstrate full conformance to the standards on in-market products. Any company looking for this certification must ensure compliance with the following:

- Data Completeness
- FDX API Conformance
- FDX Security Profile Conformance
- Automated Recipient (app) Registration Support
- Capability and Metrics Endpoints
- UX Consent Principles
- FDX Registration Integration
- FDX Monitored Performance and Availability
- Statements Support

OB Open API Specs

Open API specifications are a key element in the overall regulatory work done by FDX. In the current version, there are standardized domain/operational areas such as Personal Information, Accounts Transactions and Money Movement, based on the common needs of the market, and all these resources have been organized as below:



FDX API for Open Banking

Since these resources have been specified using OAS (Open API specification), the APIs to be exposed must be REST endpoints which have to be protected with different levels of security involving mainly transport and message-level techniques. It is up to each implementer to decide what technology (programming language/framework) or infrastructure approach (on-premises, cloud, hybrids, brands, etc.) to put in place as part of the implementation plan.

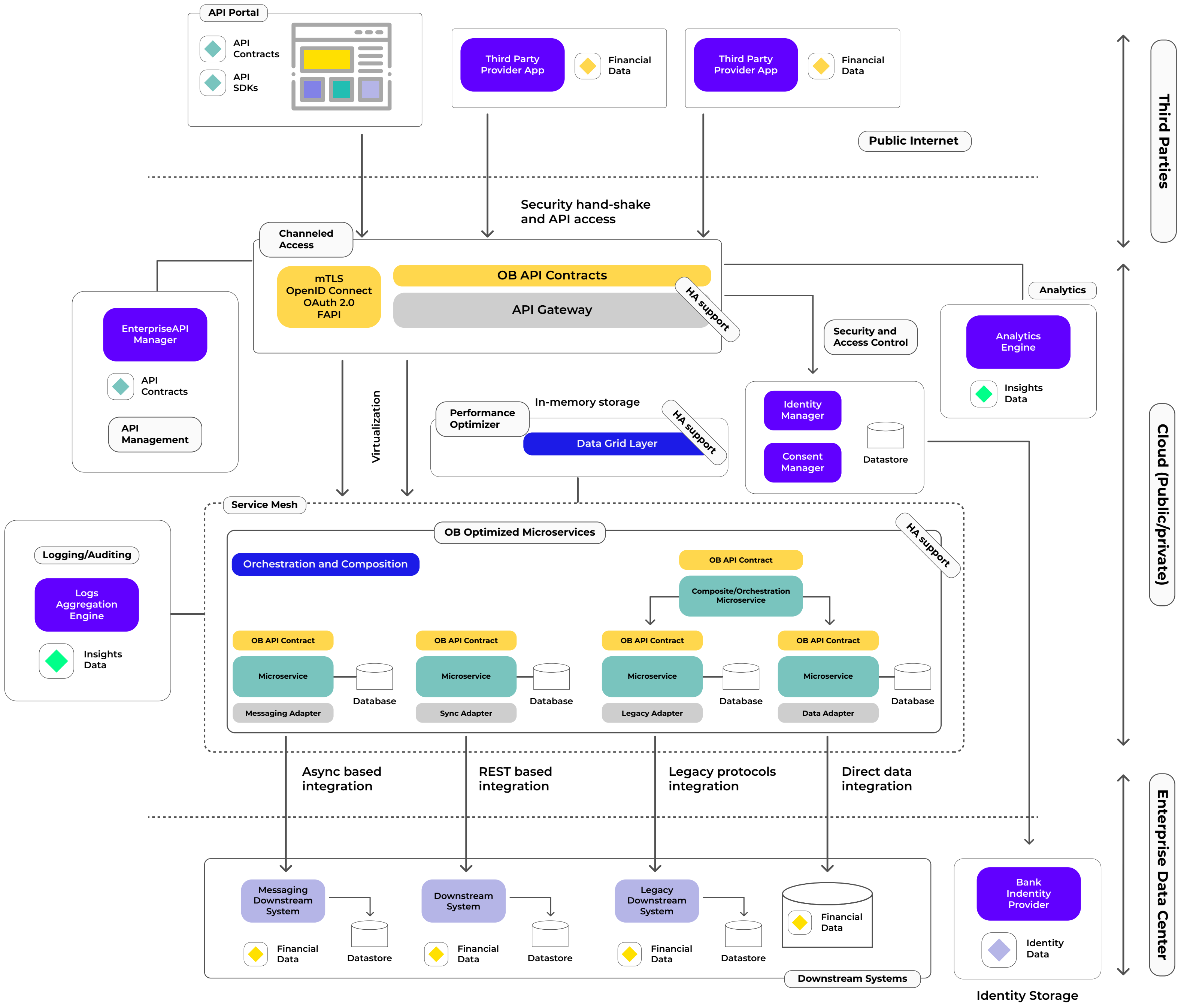
Technical Architecture

For most financial institutions the idea of adopting financial standards such as Open Banking (or even Open Finance) comes with some amount of reengineering of the existing operational applications and infrastructure services. In fact, the main driver for a transition of this magnitude will always be a modern architectural style complemented with the design techniques that are battle proven. **Microservices Architecture** is quickly becoming an effective reference model in the platform banking ecosystem due to its inherent flexibility and characteristics.

Depending on the bank's technical situation and current platform banking business models, the level of support required to meet this transition will vary. Here are the possible scenarios:

Banks with Modern Architecture (Banking Core)	Banks with Legacy Core
This category refers to organizations with modern cores, typically with **service-oriented** and **mature API-based architectures**, in general they already leverage cloud deployment-based models.	These are generally banks that still rely on **monolithic** applications with multiple point-to-point integrations. For some cases these solutions are running in traditional mainframes or regular on-prem infrastructure.

Regardless of the scenario, by considering an enriched microservices-based architecture these organizations will enable a critical driver for their platform's evolution, and all of this is due to the nature of this design style which allows greater integration flexibility (enabling rapid delivery and consumption of new capabilities while minimizing risk) through a deliberate approach with short- and long-term goals.



Technical Architecture (Logical View) for Open Banking

System Building Blocks

The intent of this reference architecture is to capture the most important elements needed to support a fully functional Open Banking implementation. A core aspect of the solution is the enablement of the existing Operational System by introducing an **Orchestration and Composition layer** on top of them (essentially this represents and adaptation of your existing application services such as Customer or Accounts platforms). Such a layer is nothing more than a combination of well-defined microservices optimized to respond to Open Banking needs following the corresponding tech specs. With that said lets deep dive into each building block to define their purpose and context:

API Gateway

The API gateway will be our end-virtualization tool responsible for the exposure of the APIs. It sits between a client and our backend services. This building block acts as a reverse proxy to process all API calls, intercept the traffic, perform aggregation when needed (to fulfill complex requirements) and return the appropriate result. Additionally, this component will help the organization understand how people use the OB APIs since it comes usually with modern monitoring tools.

Any API exposed by this device is in practice a compliant REST Endpoint that follows the OAS specs defined by FDX, it means that behind it, any service can be wired regardless of its technical interface (that’s the part where we can play around with virtualization), in addition to this, since it is the first “interfacing” point for callers, it must be the place to enforce security by implementing modern policies such as MTLS, Open ID Connect, OAuth 2.0 and FAPI.

Data Grid Layer

Its main goal is storing information in memory to achieve very high performance (and increase resiliency of the data in the event of server failure), the Data Grid Layer will represent our management system for objects that are shared across multiple microservices. One of the most significant benefits it provides is the incredible low response time, predictable scalability, and very high throughput.

By design, this component must support continuous availability and information reliability by using redundancy (which implies having in place a sophisticated approach to keep copies of information synchronized across multiple servers).

Each microservice interfacing with this layer must determine whether storing data is appropriate based on the nature of the scenarios, especially as the data stored in it is basically cached information that may be outdated.

Identity and Consent Manager

Identity and Consent Manager is responsible for the authentication and authorization required to ensure that OB APIs are secure. It is a framework of policies/technologies to ensure that the right users have the appropriate access to financing resources (backend APIs), as part of the access control, security profiles such
as FAPI are put in place.

This building block also provides an administration view for your organization that allows you to search and manage your customers' consents.

Identity Provider

The Identity Provider is an enterprise service that stores and manages digital identities.
It is mostly used for storing the identities of the final consumers/users that are interacting with the OB APIs throughout the third-party apps.

This building block is exclusively accessed by the Identity and Consent Manager, and it is considered as one of the (core) existing banking systems.

Analytics Engine

This data analytics component includes visualization tools (such as reports and dashboards) and business intelligence (BI) systems, as it is fed with the data collected from the API Gateway (usually stored as a data lake). Data scientists and other technical users can build analytical models that allow bank areas to not only understand their past operations, but also forecast what is happening in the Open Banking ecosystem.

API Portal

The portal is responsible for delivering the Developer Experience to third party developers.
It acts as a bridge between Open Banking APIs and consumers by providing information about the APIs at every stage.

API portal will allow educating developer communities about the constraints and aspects needed to call the exposed Open Banking APIs, this component communicates with the API Gateway to provision applications and associated credentials.

Enterprise API Manager

API Manager is the main driver of the API lifecycle, and will track the enablement of Open Banking APIs from their creation, testing, documentation, publishing and discovery.

In addition, the API manager will act as the management plane of the API Gateway by managing the policies to be applied on it (security, rate limiting, enablement, etc.)

Logs Aggregation Engine

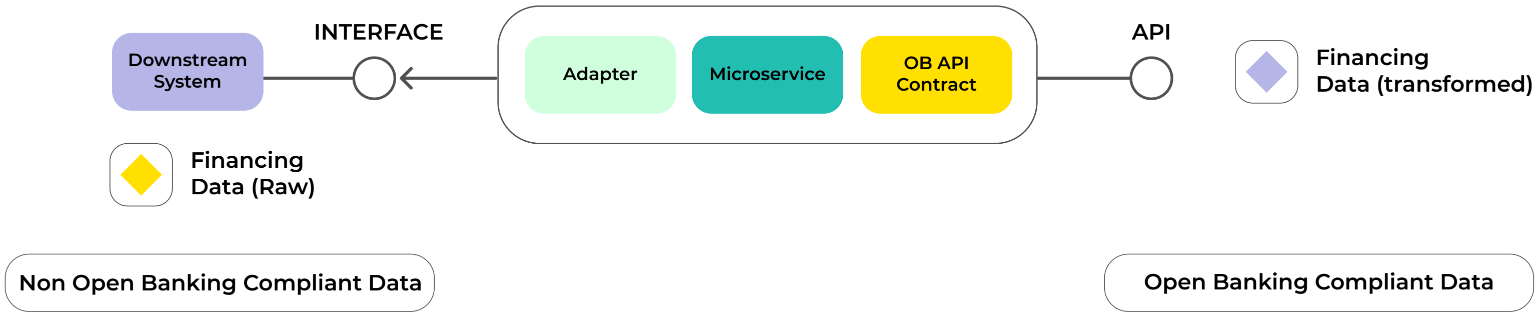
This component is responsible for collecting, standardizing, and consolidating log data from across all the OP Optimized Microservices to facilitate streamlined log analysis.

By capturing event log files from the microservice layer, IT teams will be able to convert their log files into actionable insights in real-time or near real-time.

OB Optimized Microservices

It's a combination of multiple microservice-based applications that expose REST endpoints following the OAS specs defined by FDX. These endpoints are directly consumed by the API Gateway (which is responsible for virtualization and policy enablement).

Think of these applications as custom, highly scalable adapters that interface with the existing Core Banking Systems to enable the data stored by them so that this is available for consumption from outside the bank in a compliant format.


INTERFACE
Downstream System
Adapter
Microservice
OB API Contract
API
Financing Data (transformed)
Financing Data (Raw)
Non Open Banking Compliant Data
Open Banking Compliant Data

OB Optimized Microservice

These applications will become the core of the solution and will present the right conditions for introducing orchestration and composition. Depending on the nature and technical interface exposed by the existing core banking systems, different strategies can be put in place. Here are some expected cases:

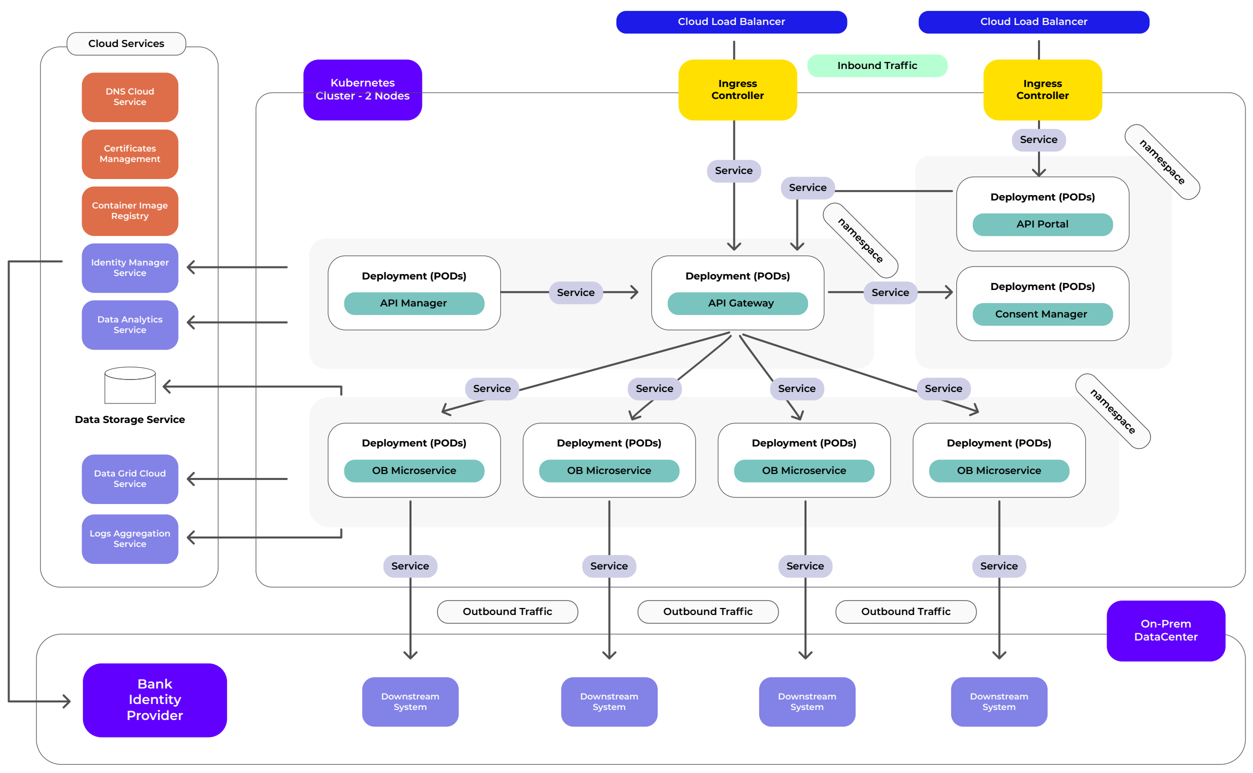
- REST Based Integration: Ideal for downstream systems exposed as REST APIs
- Legacy Protocols Integration: Indented for scenarios where services with legacy protocols are present (TCP/IP Sockets, Mainframes, RPC style Web Services, etc.)
- Async Based Integration: Ideal to integrate systems that rely on messaging (IBM MQ, JMS, Kafka, etc.)
- Direct Data Integration: Intended for scenarios where the data can be directly accessed from a repository (Databases, file systems, etc.)

Downstream Systems

Represents the existing core banking systems that store financing information such as credit, customer info, transactions and account data. Our solution does not seek to displace or deprecate these systems but enable them in the Open Banking Ecosystem by leveraging the Open Banking Optimized Microservices.

Cloud Infrastructure Design

Cloud infrastructure is a key element of the Open Banking Solution, since it brings us scalability, flexibility, cost saving and fast time to market, among others. Having the capability of spinning up new instances or retiring them in seconds will be crucial for our technical plan. That's why choosing an approach based on a Container Orchestration Engine such as Kubernetes is the ideal way to go. In this way, the core elements (Open Banking optimized microservices, API Gateway/Manager/Portal and Consent Manager) of the architecture can be deployed as scalable Kubernetes PODs.



Infrastructure view based on Cloud for Open Banking

However, this doesn't mean that all the services and capabilities required by Open Banking are necessarily PODS in the Kubernetes cluster. Many cloud providers come with prebuilt capabilities that are perfect fit for our needs, here a list of the most common cases:

- Identity Manager Service
- Log Aggregation Service
- Data Grid/Cache Service
- Logs Aggregation
- Data Storage

Open Banking Security

Security is a key consideration in our Open Banking ecosystem and from the FDX standpoint, the work to be done is definitively quite complex to be explored in a single section. But for simplicity, the most important detail we must be aware of is that every incoming request to our API Gateway must be accompanied by a JWT token which is in turn obtained by leveraging OAuth 2.0 and OPEN ID Connect flows.

Each Open Banking client sends requests using HTTP GET, POST or PUT (DELETE is not allowed) that include an OAuth token in the Authorization header following a structure as the one below:

GET /accounts HTTP/1.1
Host: my-openbanking-gateway.com
Authorization: Bearer w0mcJylss-Affgyrbvbvyrtyty6re34dssaa=
Accept: application/json
Accept-Charset: UTF-8 Accept-Encoding: gzip

The data provided as a response by an Open Banking API is limited to what the user could see using his standard login and the scopes referred in the OAuth token. As per specification, these tokens can be Bearer or MAC. But that is not all. Open Banking relies on both JSON Web Signatures and JSON Web Encryption to ensure end-to-end consistency and protect each piece of data. To take advantage of these security mechanisms both the data provider and data recipients should have RSA key pairs and the public keys should be available as signed certificates on a JWKS endpoint.

Best Practices and Recommendations

Although you have been walked through the most relevant aspects in our architectural journey for Open Banking, we are still missing the “best practices” part that complements your particular strategy. Particularly for Open Banking, these practices are a multifaceted and expansive. But here are our top nine recommendations you shouldn’t miss in your endeavor towards Open Banking:

1. Leverage existing platforms as much as possible

Especially when it comes to projects at this scale, a typical doubt that appears is whether to purchase an enterprise platform that fits with the building blocks required in the solution or develop them from scratch. Since the technical complexity is high, when possible use open source or purchase toolsets for any need that is not your actual core business (instead of trying to implement custom application services), there are competitive vendors (such as Axway) with all of the packages needed in the architectural plan that can integrate seamlessly with your existing ecosystem.

2. Enable automatic Security Scans for Open Banking APIs

Open Banking is all about exposing APIs to interact with external parties and inherently this practice may carry to raise potential vulnerabilities, this is why Open Banking adopters can (and must) frequently harden and refine their security infrastructure and resources to face future attacks properly. Nowadays there are multiple alternatives to secure APIs across their entire lifecycle. Preferably, look for choices that rely on intelligent technologies like artificial intelligence (AI) and machine learning (ML). Define periodic jobs to run your security scan and keep its rules up to date.

3. Don’t set your Open Banking architecture in stone

“Architectures set in stone”, in the Open Banking world is not convenient at all since it’s constantly evolving to address modern needs. Be flexible and consider “changes” as a natural event in the process. Design your Open Banking ecosystem to allow changes causing minimal impact. Avoid premature optimizations.

4. Accelerate your delivery process by relying on CI/CD

Having the capability to ship software quickly and efficiently is more important now than ever. Open Banking architecture itself is complex by nature and demands the setup of various software components that require a mechanism to facilitate an effective process for getting services to production faster by ensuring an ongoing flow of new features and bug fixes. Removing traditional roadblocks must be a priority while keeping the DevOps approach of aligning development and operations teams via the most efficient delivery method.

5. Keep isolation as a "must" for your architecture building blocks (Docker/Kubernetes)
We have already mentioned that Open Banking is made up of different components that cover a variety of responsibilities, and we have gone over the importance of each. This is why leveraging a solution for high availability and solution encapsulation is another key aspect in the endeavor towards Open Banking. Technologies such as Docker containers and Kubernetes further enable the foundation of your architecture.

Since resilience is a must, you should ensure that your infrastructure is robust by design and hence your OB services will be highly available, your goal should be getting to the point where your Open Banking APIs remain online even if some of the components go offline.

6. Define an API catalog with all the Open Banking capabilities
The more organized your APIs and resources are the better. This is something you can gain by bringing an API catalog which (among others) should provide you with a 360-degree view of the APIs in your ecosystem, not only the ones exposed to the outside but the ones that operate in the inside of your architecture. Elements that will be tracked include the version, contract type (OAS, protobuf, graphql, wsdl, etc.) physical location, purpose, authors, dependencies, security policies, proxies, and consumers.

Empower your teams to take advantage of this resource and include it in the software development life cycle so that it's maintained properly.

7. Take advantage of the data ingested by your APIs
In this modern era, companies need to make informed decisions about their products, services or business strategies through the data their systems collect. APIs are by far the ideal mechanism to gather such data (especially if we consider the number of expected transactions in an Open Banking solution), data analysts can help sort through this data but that's not enough, make sure your architecture leverages the proper building blocks (analytics engine, log aggregators and API Gateway) to manage and exploit each piece of data gathered by your system, this will be the only way your organization will be in a position to draw conclusions, make predictions, and drive informed decisioning.

8. Opt for a Cloud Infrastructure whenever possible

Many organizations across the world are relying more and more on cloud deployment services, even in situations where sensitive data must be moved from traditional data centers (on prem) to the cloud. Cloud computing models provide organizations with realistic flexibility in the sense that you can quickly scale resources such as memory, CPU and storage up to meet business demands without having to invest in physical infrastructure. Always keep in mind that Open Banking (as with any other solution) has special scalability needs that would be hard to satisfy with a non-cloud model.

9. Rely on abstractions as the main enabler of your existing capabilities

The architecture proposal reviewed in this document doesn't imply you must refactor your core systems to expose the expected data. It just presents a model in which an adapters layer enables your existing services to be used as compliant Open Banking resources and that's possible due to the old but solid principle of abstraction. This protection layer allows you to mitigate impacts with existing consumers (dependencies) while preparing for Open Banking enablement. It also introduces routines such as transformations, routing and content enriching.

Conclusion

In this whitepaper, we have illustrated a potential solution architecture approach (based on Logical and Cloud views) required to implement an Open Banking-based solution by addressing different concerns around the technical domain and the common challenges behind the implementation of this initiative. During this journey, we explored what Open Banking is and why it's important in the modern financing industry. We also reviewed the way these problems have been faced in the past by using techniques such as screen scrapping and the risks it comes with.

One important aspect we explored is the level of standardization and regulation offered by organizations such as FDX (Financial Data Exchange), which is the most recognized standard across North America (currently being widely implemented by recognized financial institutions). These artifacts were explored in the System Building Blocks section. Although sharing financial data between different entities may sound risky, Open Banking regulations and standards offer a trust model that guarantees security from end to end.

No Open Banking providers can interact with APIs without meeting the security regulations that are enforced by the local financial authority.

As part of the benefits of Open Banking, customers will have more control over their data, instead of being more vulnerable to security related problems.

What's next?

Open Banking is complex and requires a considerable amount of effort to reap the rewards. For more comprehensive guidance on how to capitalize on this technology, watch out for additional materials from **Blanc Labs**.

About Us

Blanc Labs enables the financial future of tomorrow. We innovate, advise, and execute technology solutions that prepare enterprises for the future. To help companies rapidly deliver on their digital initiatives, Blanc Labs has developed expertise, technology accelerators, and bespoke solutions in a wide variety of applications, including financial services, enterprise productivity, and customer experience.

Blanc Labs also offers technology advisory services in Open Banking through readiness workshops, technology strategy and roadmaps, and an Integration Centre of Excellence. These services help our clients take full advantage of opportunities such as Banking as a Service, payments, open APIs, and ecosystem integration.

Blanc Labs serves the Americas through operations in Toronto, New York, Bogota, and Buenos Aires.

To learn more about how Blanc Labs is building a better future, visit www.blanclabs.com

Open Banking Implementation Demystified: Technical Architecture and Best Practices for Open Banking Solutions

Blanc Labs
67 Yonge St #1501
Toronto, ON M5E 1J8
Canada

Worldwide Inquiries:
Phone: +1 (647) 994-5465

blanclabs.com