LEVVEL

Open Banking Architecture

Secure and Efficient Design for Open Banking

11/1/2018

Author: Chris Rigoni, Phil Mork, Jim Boone, Sonny Werghis, Luke Lallu

© 2018 Levvel.io | www.levvel.io | hello@levvel.io

Overview

What is Open Banking?

Open banking allows a financial institution to create sets of APIs which are consumed by third party organizations to present the data to a consumer within their application or website. This allows aggregators like Mint.com or Credit Karma to create a better and more reliable customer experience. For each of these situations, the customer will grant access to their data through so they are aware and agree to their data being shared.

PSD2 and GDPR

In Europe, open banking is legislated. The second Payment Service Directive (or PSD2) requires banks to provide an API layer that grants access to authorized third parties to account and payment data housed by the banks. While legislating open APIs may seem drastic, PSD2 aims to foster innovation and competition among financial institutions across the industry as well as to streamline payment processing in Europe. It also increases the security and encryption standards for the growing number of FinTechs. Partnerships with FinTechs and others stand to provide a better customer experience and innovation not only from the third parties consuming the APIs, but also from the financial institutions themselves.

The General Data Protection Regulation (GDPR), which is separate from PSD2, concerns European Union citizens, companies operating within the European Union, and any companies outside of the European Union that process data of European Union residents. The regulation provides a set of “digital rights” to EU residents revolving around their consent to use their data, clear communication on how the data will be used, and even parent consent for use of their child’s data. All of these protections are intended to give more control to the consumer over their personal data, as well as ensure the data isn’t misused, exploited, or exposed in a breach.

The Industry Today

A few institutions have invested in open banking to create an extensive offering. Some have created a developer hub for their API offering, which offers a sandbox to experiment with APIs, and creates opportunity for furthering the conversation into becoming an API client. Many also list out their API sets to

© 2018 LEVVEL.IO ALL RIGHTS RESERVED

allow potential clients to easily find what they know they need, as well as spark new ideas for what they could offer their customers.

While some examples fall into the Banking as a Service category, there is a major point that is important to highlight. Previously, a startup FinTech or other company would only gain a charter from a bank, but they would still build backend connections, as well as the front end customer experience. When you think of banking as a service in terms of the direction open banking is currently headed, the startup would simply have to build the front end customer facing experience. They could then utilize the charter to perform banking services and utilize existing connections to the bank to provide additional services such as tokenization (Apple, Samsung, and Google Pay), real-time payments, and others using API sets in an open banking agreement.

There are two main draws to open banking: an alternative to screen scraping, and monetization of the APIs. Most aggregators (companies that offer budgeting and other services in one place using all of your financial accounts) that use banking information (such as account balances, interest rates on credit cards, etc.,) acquire the information by requiring the customer to input their login information for their financial institution. Once entered, the aggregator logs into the consumer account and scrapes the screen to obtain the balance and other data. This has the potential to cause issues with the financial institution's online site, is not secure, and can be an overall poor customer experience. In addition, if the financial institution ever changes their user interface for the customer-facing site, the aggregator could have to update their screen scraping setup in order to account for the changes. In the interim of the update, the customer is unable to access that account within that aggregator offering.

Given that screen scraping can cause issues for both the financial institution, as well as the entity that needs the data, there are definitely motivations to use a more secure and efficient manner to accomplish the same goal. In addition, financial institutions are able to capitalize on the aggregators and other entities that could use the data by monetizing the APIs that are used. This has the potential to create new and potentially lucrative revenue streams within the financial institution. In all of this, the consumer should be at the forefront of considerations, and have control over how and where their data is shared.

© 2018 LEVVEL.IO ALL RIGHTS RESERVED

Key Technical Considerations for US Based Banks

Open Banking API Security

Banks have developed robust security and governance practices around access and use of customer data—both for internal use and for data sharing with partners. These practices have been developed over many years of experience with risk management and regulatory compliance. Traditionally, approaches to data security have been driven by the need to reduce risk and fraud through theft and abuse of sensitive personal and financial data. Such goals have been achieved by protecting tightly controlling access to data.

Open Banking requires banks to facilitate more open access to customer data through public APIs. In this scenario, the traditional goals of risk management and regulatory compliance continue to remain paramount. But, they are augmented with additional goals such as open data sharing, ease of data consumption, and customer control of their financial data. As a result, publishing and securing Open Banking APIs present new challenges.

Bank Customer Authentication and Authorization

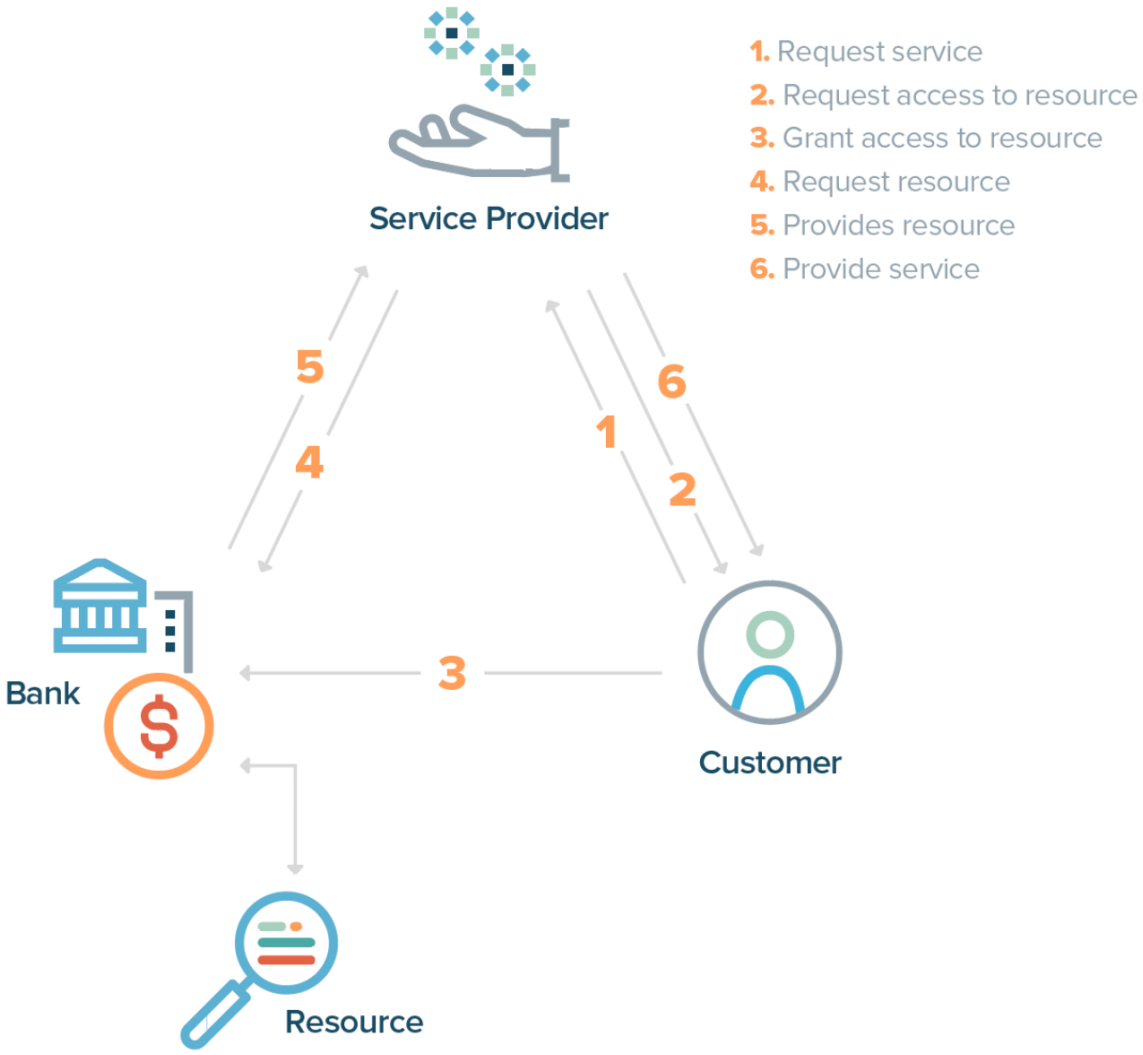
Open Banking empowers customers to choose who their data is shared with, and how much of their data is shared. Banks must facilitate targeted data sharing while continuing to provide the same high degree of security as today.

In this scenario:

- Customers are “Resource Owners”, with their bank data being the resource that they share with trusted third parties.
- Banks, which are the “Resource Servers”, must allow these trusted third parties to access (read) and/or perform actions (modify, update) on these resources depending on the level of permission granted by the resource owner.
- Resource owners must be able to grant permissions to these third parties, or “Clients”, without handing over their security credentials.
- Resource owners must be able to determine what level of permission is to be granted (for example: which data can be read, what actions can be performed, etc) to these third parties.
- Resource owners must be able to revoke permissions granted to third parties at any time.

© 2018 LEVVEL.IO ALL RIGHTS RESERVED

OAuth2.0 has become the industry standard as a security model that supports resource sharing. In this model, applications (such as those provided by third parties) can securely request access to resources (such as bank data) hosted by a server (such as a bank) on behalf of the resource owner (such as customers of a bank). The following figure represents a simplified schematic of the relationship between participants in OAuth2:



© 2018 LEVVEL.IO ALL RIGHTS RESERVED

While OAuth2 does not explicitly address how the resource owner is authenticated, it requires that an "Authorization Server" verify the identity of the resource owner. It is implied that this Authorization Server authenticates the resource owner before access to a resources is granted. In most situations, banks will act as the Authorization Server, the exceptions being when banks have outsourced this function to an external service provider. Banks are free to use their existing security Identity and Access Management (IAM) systems to authenticate the customer (such as using username and password, account ID and pin, multi-factor authentication, etc).

A recent development in this area is Strong Customer Authentication (SCA) required by PSD2. SCA sets stricter requirements for verifying a customer's identity before a financial transaction is initiated. As SCA is more widely adopted, it will significantly increase the security of Open Banking APIs and reduce the incidence of financial fraud. In summary, it requires authentication to use at least two of the following three elements:

- Knowledge: Something the user knows, such as a password, PIN, etc.
- Possessions: Something the use has, such as mobile phone, token, or card.
- Inherence: Something the user is, such as facial recognition, fingerprint, or iris scan.

OAuth2 specifies how third parties may request revocation of permission that was granted to them. This is generally used when the customer uninstalls the third party application or cancels their account with the third party. But, banks must also provide a mechanism for the customer to revoke permission to prevent unauthorized use by rogue third parties. This can be as simple as removing an entry (called token) corresponding to a third party from an internal database.

Data Governance

As Open Banking use cases evolve and new service models become available to customers, banks must balance data privacy and protection with customer control and transparency. Existing data classification and governance models may be inadequate or too restrictive.

Two service models that have clearly emerged from banking regulations in UK and EU include Account Information Services and Payment Initiation Service. Banks that support these services models must take a closer look at data required to fulfill each of these services. Data that was previously classified as

© 2018 LEVVEL.IO ALL RIGHTS RESERVED

sensitive, such as account numbers, account balances and transaction details, may be made available for sharing with third parties that provide these services. In this context, banks could enhance existing governance policies by applying the following recommendations:

- Identify the minimal set of data required to fulfill the business requirement that each API supports: This requires a clear and unambiguous definition of the business requirement, which is ideally achieved by means of standard specifications published and governed in partnership across the industry. Regulatory initiatives such as PSD2 in EU also provide a mechanism to define such services.
- Limit access to sensitive data at individual customer level only: In order to minimize data loss through fraudulent activity, banks must avoid making customer data available in bulk via public APIs. Customer data that is made available via public APIs should require clear and specific consent from the individual customer.
- Provide aliases or tokens in lieu of account numbers to minimize risk of fraudulent activity: Account numbers are sensitive information that can be misused when they fall into the wrong hands. Banks must consider replacing account numbers with single-use tokens or aliases when made available via Open Banking APIs. Single-use tokens could be used to facilitate payment services with reduced risk of man-in-the-middle attacks. For APIs that provide account information only, an alias would be a suitable replacement for an account number.

Data that has been accessed by third party service providers on behalf of the customer are generally copied and stored in systems that banks have no control over (such as in the cloud or a data center used by the service provider). This poses yet another challenge in terms of data security and privacy. Banks cannot prevent the abuse of this data since it no longer has control of it. Customers are sometimes unaware of the sensitivity of financial data. More frequently, they do not attach the same degree of importance to data as banks do. Banks must act to reduce their liability in such situations by educating customers on the risk of sharing sensitive data and maintain clear audit trails of data accessed by third parties.

Threat Protection

Open banking increases the need for banks to guard against external and internal threats. As the number of interfaces through which data can be accessed

© 2018 LEVVEL.IO ALL RIGHTS RESERVED

grows, the type and intensity of external threats will also grow. Existing threat protection techniques that are currently being employed by banks must be extended to cover APIs. Some examples of external threats against APIs include:

- Denial of Service (DoS): API endpoints are prime targets for DoS attacks. Ensure that these APIs are protected by a robust secure gateway and are proxied by local and global load balancers.
- Data Loss: Monitor API usage and data flow, using Data Loss Prevention tools, to protect against accidental or intentional data loss.
- Misuses and Abuse: Establish quotas and rate limits by using an API gateway to ensure that third parties do not misuse or abuse APIs (such as by collecting large amounts of data for analytics, running load tests and stress tests, etc.)

Bank Digital Identity

Customers are increasingly using non-bank service providers to aggregate and consolidate their financial data from multiple sources, including their banks or credit unions, credit card issuers, investment banks, etc. As open banking evolves, the types of services that customers can utilize from these non-bank entities will become more valuable and more popular. Banks have an opportunity to play a critical role as the identity provider in this ecosystem. Bank customers could use their bank identity as a portable “digital identity” as they consume services provided by non-bank entities.

To make digital identity portable it must be accessible to multiple parties. This can be achieved by using a centralized model with its management delegated to a trusted industry body. However, this introduces a single point of failure in the system. Additionally, any such centralized entity would become a natural hacker-magnet. A distributed identity management model, where many parties collect and share information with each other, is an alternative approach that mitigates the risks of a centralized model. Many startups are developing solutions based on Blockchain technology that would provide a secure distributed identity management platform.

A Blockchain-based platform would also allow bank customers to store their data-sharing preferences along with their identity in a transparent and secure manner. For instance, a customer may declare that a non-bank service provider use their bank identity to authenticate them, and use their account balance and spending habits from specific financial institutions to help with budgeting. The customer would have clear visibility into who has access to their data and

© 2018 LEVVEL.IO ALL RIGHTS RESERVED

be able to make changes or revoke permissions quickly and easily. This would accelerate innovation and adoption of novel ways of delivering and consuming financial services.

Bank-driven digital identity is a developing initiative that is gaining traction outside the US in places such as Canada and Australia. More broad industry-driven or government-driven digital identity initiatives in which banks participate are also available in Netherlands and Finland. As this theme gains wider adoption, banks should consider how it intersects with their open banking strategy and how it can be leveraged to foster innovation in financial services.

Open Banking API Modeling

Most banks have implemented a normalized domain model to facilitate data-sharing between internal systems. Examples of this include the IFX model, the IFW model from IBM, and the Terradata financial services data model. As anyone who has worked with any of these models knows, they are very complex by design to include all types of data that might be shared between internal bank systems. The models are also completely defined with XML schemas, and the data is typically exposed via an ESB using SOAP services that are defined with WSDLs. As a former producer and consumer of these services, the SOA world as we know it is hard to beat when checking all of the enterprise architecture checkboxes for schema validation, service cataloging, compliance, versioning, security, and so forth.

However, if we fast forward to the Open API world, the de facto standard across the internet is now RESTful services with resource-based JSON object models. This is completely different from the SOAP protocol and XML schema-based object models. It's impractical to consider ripping and replacing enterprise legacy models with a new resource based REST resource models, so what's to be done? If we simply expose our object based SOAP model via RESTful services using an API gateway conversion service, then the returned data payloads would be overkill for what our RESTful clients need. We have seen several SOAP based services that return over 500 fields of data that is nested ten layers deep. It was painful to consume that data in applications within the enterprise—imagine exposing that much data via RESTful APIs, not to mention the security concerns it would create? So, what is the solution?

This is when to open the Gang of Four Design Pattern book and find the facade

© 2018 LEVVEL.IO ALL RIGHTS RESERVED

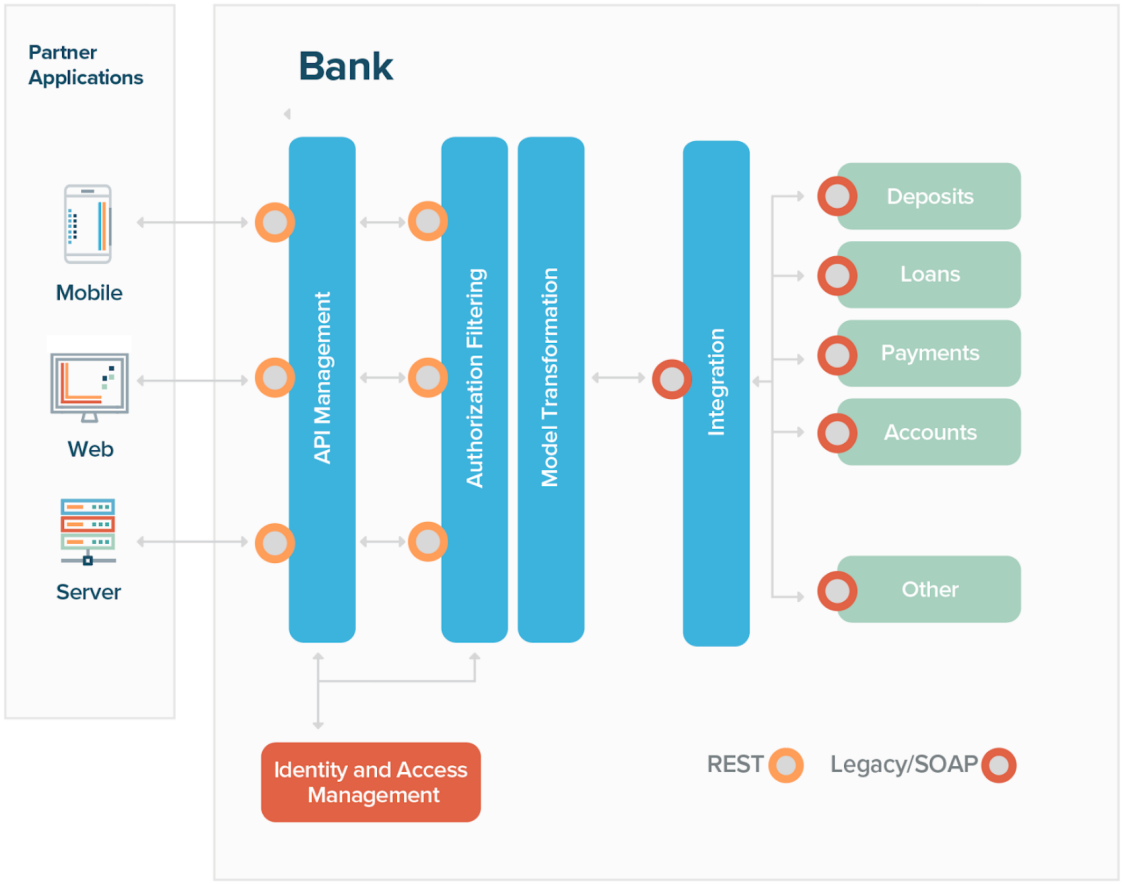
design pattern, which looks like a perfect solution to this problem. The facade pattern is similar to the adapter pattern, except it includes a simplification of the adapted system as well. What's needed is a facade that will expose the complex enterprise-level data as simple RESTful-based API services. The solution sounds simple, but as always, the devil's in the details.

The complex data structures need to somehow be exposed in a much simplified form while dynamically trimming the amount of data that is returned. The data must be dynamically filtered so that only data relevant to the business use case and the client's authorization credentials are returned. Then the resulting data needs to be mapped from our object model into a resource-based API model before it can be exposed to RESTful clients. Simple enough, right?

Maybe, maybe not. One of the biggest unknowns is what sort of JSON resource model to map the Enterprise data to. Should a custom resource model be written and hope that that clients will consume the data model? Should a standard model be adopted? The problem is that there are currently no standard Open Banking JSON API data models. What's left is doing what most people do: networking with industry friends or searching the internet for what other organizations have already done to help solve this problem.

So far, it's known that if the facade design pattern is implemented, then a resource-based API model suitable for RESTful services needs to be defined on the client side of the facade. Also needed is to transform the data returned from the schema-based object models into the JSON resource model all while allowing for filtering the data based upon authorization credentials. The following figure illustrates this:

© 2018 LEVVEL.IO ALL RIGHTS RESERVED



It is seen that partner applications access the APIs through an API management platform. There are many of them out there to choose from, both commercial and open source. The APIs exposed by the API gateway must be back-ended by logic that implements authorization filtering as well as model transformation of the data received via the ESB or integration bus.

From the enterprise side of the facade, many of the large players in the financial model domain space are attempting to expose their models via RESTful services. These efforts might simplify accessing and transforming the enterprise model, but some way to prune the model data to meet the business case and authorization requirements is needed. It is likely that banks will still need to customize the JSON payloads regardless of what the domain model vendors do.

On the client side of the facade, a resource-based API model that is consumable by partner applications is needed. This is the same problem faced by European banks implementing the PSD2 initiative. They also need a resource model, so they are a good source for understanding how organizations have approached the solution to this problem.

© 2018 LEVVEL.IO ALL RIGHTS RESERVED

There is also an open-source project called the OpenBankProject which has defined a suite of over 140 RESTful APIs and a JSON data model that is currently used by some FinTechs. It's worth checking out this project to get a feel for what a resource-based banking model model looks like from a client perspective.

One of the most important challenges is the selection of an API Management platform to support the Open Banking APIs if the bank doesn't already have one. An internet-facing API gateway is becoming increasingly necessary—not only to support Open Banking APIs, but also to support interacting with external parties such as the Zelle network.

You could write your own, but there are plenty of great commercial and open-source options to choose to provide a faster time to market. Levvel has worked closely with Apigee, a very robust API management platform that Levvel highly recommends. The choice of the correct API management platform is critical since this component typically implements most, if not all, of the security components.

Another challenge is going to be adopting or creating a RESTful API suite that exposes just the right amount of data that needs to be exposed for each use case. It is a significant amount of work to write an API suite from scratch, so Levvel recommends surveying the existing approaches both within Europe and the U.S. to see if there are any open banking API suites that have gained traction. Once again, you may want to take a look at the OpenBankProject mentioned earlier.

The final challenge is going to be transforming the SOA object model into a RESTful API resource model. There are plenty of object mapping libraries out there that could be used to do this, but the mappings could be very tedious to define and implement. In order to ensure that only the desired data is exposed, the mapping of this data must be done carefully and correctly.

Data payload filtering may also be a challenge since you will need to interface with the authorization system, and again, map which data fields should be available for each authorization credential or role. Note that none of these challenges are insurmountable, and as the market develops and matures, more and more solution options are likely to become available.

© 2018 LEVVEL.IO ALL RIGHTS RESERVED

Open Banking API Onboarding and Management

Some banks and other financial institutions have joined the API economy by publishing and monetizing APIs that allow developers to access resources and services provided by the bank. At a high level, APIs can be categorized into a few different types: internal, partner, and public (or open). Internal APIs are those built by banks to allow bank applications to communicate with each other. Partner APIs are APIs banks use to integrate with partners such as vendors, payment networks, etc., but are restricted to use by only those specific partners. Public APIs (also referred to as Open APIs) are APIs built for developers outside the bank to leverage.

All APIs require onboarding and management by the API provider, but Open APIs may present some challenges to banks that have only been accustomed to internal or partner APIs. In this section, some key aspects of publishing and managing APIs will be briefly discussed.

Developer Onboarding

While APIs enable new services that are targeted at customers of financial services, banks must recognize that their primary audience is the developer community. Facilitating and streamlining consumption of APIs by developers is critical to monetizing them. Developer registration can be manual or automated. Though automated registration is preferred for mature, well-established API products, manual registration may be an option when starting out for banks that are not mature in this area.

In general, the development community wants the onboarding process to be as streamlined as possible, and ideally for it to be self-service for the sandbox environment. Since developer adoption is critical, strive to make the developer onboarding as streamlined and automated as practical for your situation while still satisfying all security and compliance requirements.

Then, have a more rigorous approval process before a developer or third party is granted API access to the production environment. As the market matures and governance models are established, automated registration with identity verification could become the norm.

Sandboxes or Test Environment

© 2018 LEVVEL.IO ALL RIGHTS RESERVED

Sandboxes are environments that developers can use to test and validate the API. Such environments generally mimic the characteristics of the API in production by providing simulated responses. Developers can use these responses to validate how the API works during the development of applications that use these APIs. Sandboxes also provide simulated error responses that allow developers and testers to ensure that their application correctly handles API errors and failures, both due to invalid input and system or network errors.

Documentation and Support

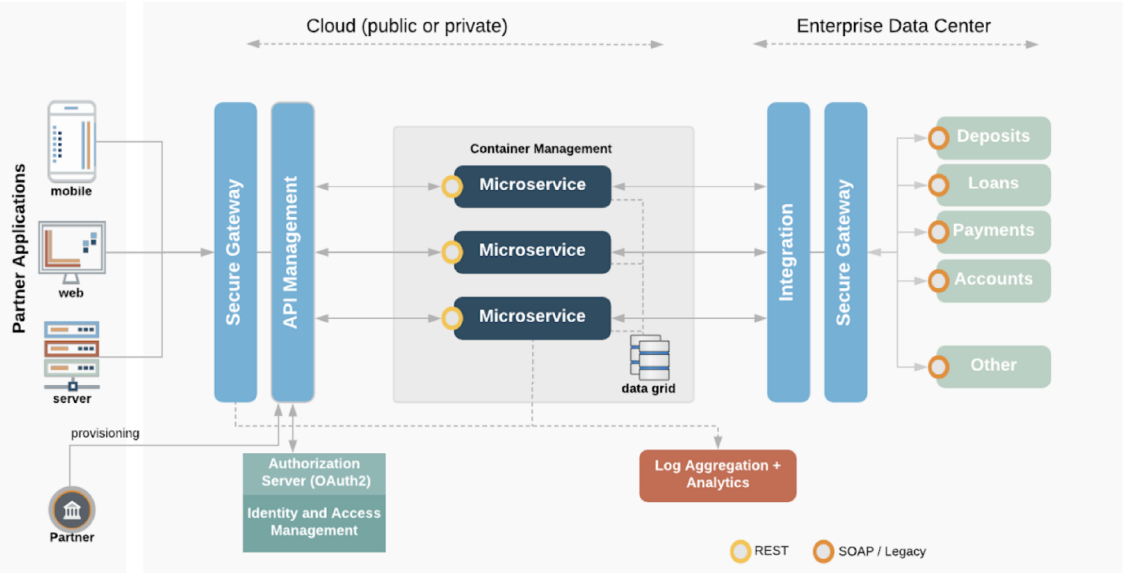
Clear documentation and support are essential to successful adoption of APIs by the developer community. Many API products see poor uptake because of the lack of guidance on their usage. API documentation must be readily available online, and should be comprehensive enough to ensure that developers have a clear understanding of the capabilities and limitations of the API. Banks can provide API support by creating online user communities and forums. Searchable documentation and knowledge base that includes frequently asked questions and answers reduces the need for live support and significantly eases adoption of APIs by developers. Banks should be dedicated to maintaining and improving the documentation along with the API itself.

Monitoring and Analysis

Monitoring of Open Banking APIs is no different than any other public or internal bank APIs. Active monitoring provides visibility into usage, performance and uptime. It surfaces issues that can impact customers, and enables easier and proactive troubleshooting. More importantly, given the sensitive nature of financial data exchanged via Open Banking APIs, monitoring and analysis can expose potential or actual data leakage and data theft. Data governance and security policies previously discussed can be enforced via monitoring and analysis. Additionally, monitoring and analysis provides insights into customer usage of bank resources. This information can be used to enhance customer experience and accelerate monetization of the APIs.

Putting It All Together—Architecture Recommendation

© 2018 LEVVEL.IO ALL RIGHTS RESERVED



A high level logical architecture to support Open Banking API is shown above. This section will cover technical recommendations for various components of this architecture.

Integration Pattern

Integration Patterns that are widely used in traditional Service Oriented Architecture (SOA) and Enterprise Application Integration (EAI) domains are well suited to APIs—specifically, the facade pattern which can be used, “provide a simple interface to a complex subsystem” . To implement this pattern, start with defining the API to support the use case. Next, develop and validate the API using stubs that represent back-end systems. Finally, integrate with back-end systems using data transformation, aggregation, and protocol mediation as needed. API facades can be implemented as microservice, which is discussed in the next section.

Microservices

Microservices are loosely coupled, fine grained, lightweight, and independently deployable components of an application. Some of the benefits of adopting Microservices for implementing Open Banking APIs include flexibility of deployment, agility to roll out changes quickly, scalability through loose coupling, and availability through redundancy.

© 2018 LEVVEL.IO ALL RIGHTS RESERVED

Container Management

A complementary technology to consider when implementing APIs as Microservices is containerization, which reinforces some of the benefits of Microservices such as deployment flexibility and reliability. Popular containerization management stacks, such as Docker, Kubernetes, and CloudFoundry, offer a variety of deployment options to comply with technical standards established by the bank's enterprise architecture. They can be deployed either in a public cloud (with security provided via a virtual private cloud) or in an on-premise private cloud environment in the bank's own data center.

API Management

Banks that develop Open Banking APIs must publish, secure, and manage them over their lifecycle. This is best accomplished using available API Management tools that facilitate "creating and publishing web APIs, enforcing their usage policies, controlling access, nurturing the subscriber community, collecting and analyzing usage statistics, and reporting on performance." These tools frequently provide the following capabilities:

- Visual API design tools to quickly define, assemble, and publish APIs
- Configuration and enforcement of access control, permissions, and policies
- Configuration of licensing, billing, and payment options to facilitate monetization
- Subscription management, socialization, and community development
- Event monitoring and usage analytics

Secure Gateway

- Purpose-build Secure Gateways provide additional layers of security over that offered by API management tools. This could include:
- Transport Layer Security (TLS) enforcement with encryption/decryption and certificate management
- Input validation
- Content-aware threat protection including Data Loss Prevention
- Intrusion detection and prevention
- Throttling, size limiting, and rate limiting

© 2018 LEVVEL.IO ALL RIGHTS RESERVED

Authentication and Authorization

As previously discussed, Open Banking requires banks to authorize partner applications to act on behalf of their customers. This requires banks to identify the customer (authentication) and get their permission to share data with a partner (authorization). A number of Identity and Access Management (IAM) tools incorporate OAuth2, which is a protocol that facilitates this scenario. Banks may be able to leverage their existing security infrastructure for this purpose.

Analytics and Monitoring

Open Banking API architecture must include log aggregation and analytics to facilitate operational functions such as root cause analysis, performance management, security intelligence, and monitoring. Log aggregators provide connectivity to many application and infrastructure components from which logs can be collected. These logs can be forwarded to an analytics engine for normalization, enrichment, and correlation of data from which operational and business insights can be derived.

About Levvel

Our team is made up of leaders, thinkers, designers, and builders whose collective experience spans many industries and companies of all sizes. We're avid fans of technology but even bigger fans of solving business problems. An understanding of the difference between an idea and a solution is just one of the things that sets us apart.

Our Payments practice combines enterprise payments experience, technical expertise, and a pulse on emerging technologies to ensure our clients create the right approach the first time. Our team has broad, strong industry relationships and expertise across the payments ecosystem.

hello@levvel.io
980.278.3065

© 2018 LEVVEL.IO ALL RIGHTS RESERVED