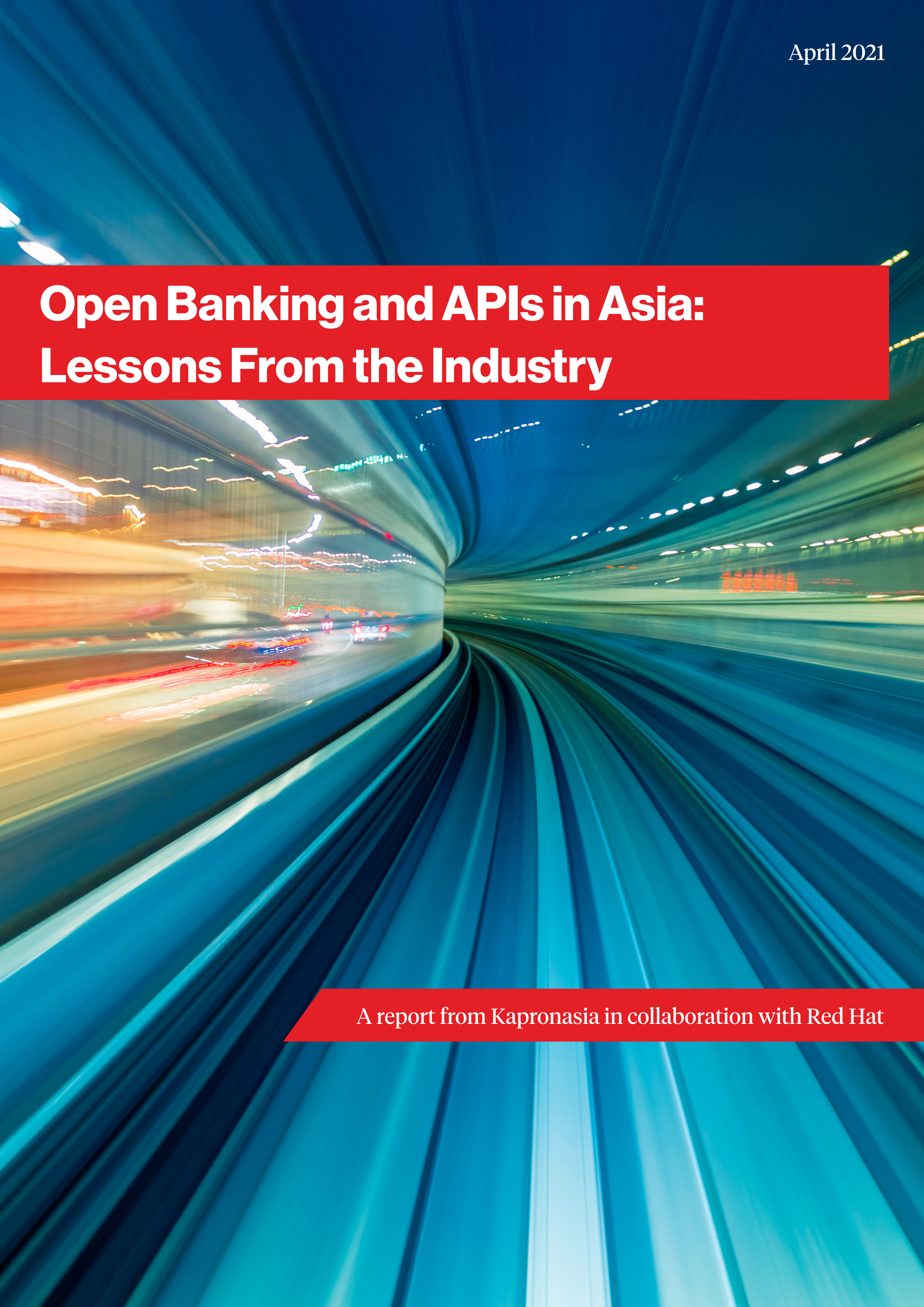

April 2021

Open Banking and APIs in Asia: Lessons From the Industry

A report from Kapronasia in collaboration with Red Hat

Contents

Methodology

For this report, Kapronasia conducted both primary and secondary research in Asia to obtain the most relevant insights from the industry around APIs and Open Banking.

Secondary Research: Sources included but were not limited to, market intelligence reports and studies by industry experts and professional services networks, white papers, educational materials, media articles, and marketing collateral.

Primary Research: Interviews were secured from relevant players across the ecosystem, including financial institutions, fintechs, and industry experts.

Executive Summary

Both regulatory and competitive forces have been making Open Banking a new reality across the region. Banks are now realizing that if they want to keep their existing customers, acquire new ones, and play a greater role in their customers' lives then they must become more customer focused, while offering a broader range of digital products and services. To do this effectively, banks will no longer be able to be vertically integrated institutions and will be required to shift to a distributed Open Banking model to collaborate with third parties. Such a model requires APIs, the digital ports that enable communication between services.

Open Banking technology, however, does not fit neatly with many legacy core banking systems, which are currently preventing effective interoperability with Open Banking APIs. Most legacy systems have a complex and inflexible IT architecture, whereas Open Banking demands flexibility, agility, and scalability. Moreover, banks' core systems are transactional in nature while the digital/Open Banking flow is based on a customer-centric journey which requires service orchestration – the sequencing of "microservices" glued together with the appropriate business logic.

Because replacing legacy core systems is an expensive, risky, and lengthy process, many financial institutions (FIs) have been breaking out core system functionalities that are essential to handling the digital customer journey, by adopting a microservices based architecture. This then enables banks to abstract away from their core system platforms and operate with a more modern, scalable infrastructure that sits on top of these legacy systems.

While this approach enables banks to become modular and platform-based in order to effectively collaborate with players in the financial services ecosystem, many FIs have in fact failed to build this abstraction layer correctly. Such an abstraction layer should include a first layer of microservices that abstracts the service away from the core systems. A second layer that orchestrates these base services to turn them into the bank's processes; and a third layer which is the API Gateway, an API management tool that exposes microservices and hosts other functions such as authentication, security policy management, and load balancing. Each layer comes with its own technical difficulties, however, which makes it hard for banks to get Open API Banking right.

At the microservices layer, while banks have often been able to break down their services, many have struggled to create true microservices, which are an order of magnitude more difficult to build. Without these true microservices, banks are unable to have horizontal scalability – the ability to run parallel, distributed processes across the infrastructure to handle the load. The problem occurs both on premises and on the cloud and therefore the limitation is not due to the choice of underlying infrastructure, but rather is an application architecture issue. Such horizontal scaling provides not only resilience, but also real elasticity by enabling applications to utilize the full capabilities of the cloud.

At the orchestration layer, many banks have either neglected to build orchestration entirely and have simply atomized their backends and built microservices exposed by APIs on top of these. Or, where there is orchestration, it will be a technical orchestration, rather than a business process orchestration. Both instances speak to the fact that the end customer is often missed out of these API initiatives.

In the first instance, APIs have been exposed for the sake of it rather than evaluating how these are going to impact consumer behavior. There is also no orchestration to deliver the customer journey. In the second instance, a purely technical orchestration will lack the appropriate business logic to guide the customer journey resulting in a suboptimal customer experience.

At least where external APIs are concerned, banks must always put the customer journey front and center. That means that developers for their part will need to take a customer-centric approach when designing APIs, which will include the need to think carefully about the underlying business process at the orchestration layer that will deliver the customer journey.

In the end, the winners will be the ones that can build a truly differentiated customer offering based on efficient service platform(s), which could be a mix of both internal and external services. It is therefore going to be critical for banks to have the right architecture and organizational design in place to rise to the challenge.

Open Banking in Asia: A brief look at where we are Today

Open Banking is Taking Off in Asia

Open Banking, the system of allowing access and control of consumer banking and financial accounts through third-party applications via open Application Programming Interfaces (APIs), is sweeping across Asia-Pacific. As one might expect from such a diverse region, the roadmap for Open Banking development varies. Some jurisdictions such as Australia, Hong Kong, and Singapore are forging ahead.

Drivers of this shift are both regulatory and commercial. On the regulatory side, regulators in APAC have on the whole taken a "soft" approach, preferring to support Open Banking with specific policies and guidance rather than strict regulations and timelines. Regulators in Japan, South Korea, and Singapore, for example, have created API "playbooks" encouraging API developments and perimeters.

Only Australia and Hong Kong have so far opted for a regulatory-driven approach. In Australia, Open Banking was launched under the Consumer Data Right (CDR) Act introduced to the banking sector in July 2020.[1] In Hong Kong, the Hong Kong Monetary Authority (HKMA) published the "Open API Framework for the Hong Kong Banking Sector" in July 2018, setting out a four-phase approach for banks to implement Open APIs.[2] In contrast to this top-down regulatory approach, in markets such as India, Thailand, and Malaysia, Open Banking has been driven more by bottom-up customer demand.

Competition is a Key Driver

If regulators have been encouraging, it has been competitive forces which have been making Open Banking a new reality across the region. The larger incumbent banks within APAC are feeling competitive pressure from the emergence of digital banks and have understood that they must innovate if they want to avoid losing market share to these new players. Any market share lost now will be harder to recover later as the innovation-driven marketplace becomes more competitive and crowded.

To remain competitive, these larger incumbents are developing and launching their own new products and services, often in collaboration with each other and with FinTechs. According to a payment service provider interviewed for this report, a large East Asian bank came to them to partner because they were losing business to digital banks and realized that they needed to have a product that gave them that feature parity to prevent customer bleed or churn.

The small and mid-market incumbents, meanwhile, are being squeezed from both ends. Like the larger incumbents, they face similar competitive pressures from the new challenger banks, but unlike their larger counterparts they are unable to compete on volume and/or cost. Their approach is more likely to focus on their core relationship business and to partner with a digital bank to get access to low-cost digital services without having to build these themselves.

The digital banks, meanwhile, also have an interest in Open Banking. Being startups, these are primarily driven by customer acquisition and so a big hook for them to gain the kind of customers they want to onboard, open accounts, and monetize the customer relationship is to be able to make the business case that their product is the cheapest, fastest, and/or most transparent. These too then will be incentivized to partner with organizations that can provide them with that edge, enabled by Open Banking.

Should Open Banking become more regulated within the region and as Open APIs come to the fray, the competitive pressure from digital aspirants, e-wallet providers, and account aggregators is going to intensify, further driving banks to innovate to keep customers engaged on their platforms. Organizations that are not adequately prepared will face higher costs of tactical change and find themselves at a significant disadvantage as the marketplace rapidly evolves.

1 "What Is the Consumer Data Right?" Office of the Australian Information Commissioner, Accessed March 28, 2021. https://www.oaic.gov.au/consumer-data-right/what-is-the-consumer-data-right/.

2 "Open Application Programming Interface for the Banking Sector." HK Monetary Authority, May 4, 2020. https://www.hkma.gov.hk/eng/key-functions/international-financial-centre/fintech/open-application-programming-interface-api-for-the-banking-sector/. EMEA Centre for Regulatory Strategy, "Open Banking around the World." Deloitte, Accessed March 28, 2021. https://www2.deloitte.com/tw/en/pages/financial-services/articles/open-banking-around-the-world.html.

The Need for a Comprehensive Open Banking Integration Strategy

Banks Must Shift to Customer-Centric Processes

To meet the demands of digital/Open Banking and build a truly differentiated customer offering banks will have to move from a monolithic, legacy architecture to a cloud-native, microservices-based architecture. Such a shift will also require a new organizational structure, from traditional waterfall methods to more agile approaches.

The problem for banks, however, is how to replace their core banking systems with something more modern and scalable?

One solution for banks is to adopt a Domain Oriented Microservices Architecture, while keeping their existing legacy, core systems. This will allow banks to abstract away from their core system platforms and operate with a more modern, scalable infrastructure that will sit on top of these legacy systems. Done right, such architecture will enable banks to become modular and platform-based in order to effectively collaborate with players in the financial services ecosystem.

A Domain Oriented Microservices Architecture solution that enables banks to abstract away from their core systems, migrating critical customer-facing functionality and data to new service-based platforms, typically involves three-layers:

- A first layer of microservice function that banks need to build on top of their core system. This abstracts the service away from the core system and breaks it down into its core functions, but these functions do not give the bank their business process. They give banks access to data and enable them to trigger simple, unit-level transactional operations, i.e., to create an account, book loan, etc, but without the business process layer around them.

- A second layer that orchestrates these base services to turn them into the bank's processes. A proper business orchestration manages sequencing, while gluing services and logic together. Often banks will atomize their backend, exposing microservices via APIs, but will then fail to build the business orchestration layer on top that combines/aggregates these different atomic services into something more like a business API or a business service. Or, where there is orchestration, it is a technical orchestration rather than a business orchestration.

- A third layer which is the API Gateway that acts as a reverse proxy to accept all API calls, aggregate the various services required to fulfill them, and return the appropriate result. API gateways usually also handle common tasks that are used across a system of API services, such as user authentication/authorization, rate limiting, and statistics.

It is important for banks to decide who manages the API Gateway and to establish governance rules around it and the APIs that it hosts. There should be a governance body that rather than prescribes API standards; guides and defines standardized practices for API implementation that applies across the entire organization. Implementers, for their part, should then understand and apply standards where they fit, while documenting and reviewing any deviation.

Moreover, while the prescription of API standards would be too rigid, a free for all would invite inconsistency. To thread this needle, firms might want to consider using an "intelligence scaffold." This is a framework for API design which includes what must be included, what should be included, and what would be nice to have included in the design of the API. It is then up to organizations or groups within organizations to decide how far to take "musts," what are "should" elements, and what are "nice to have" elements.

Sham Arora, CIO, Global Head of Enterprise Technology at Standard Chartered says that the Bank runs a tight governance around APIs that acts as an API barometer. For example, an API Governance committee looks at how fit an API is for the external client in terms of whether it embodies quality with regards to everything essential for API health: The availability, performance, and functional correctness of not just the API endpoint, but also the business transactions that the API supports.

Digital/Open Banking Implementation Challenges and Solutions

Legacy Core Systems

Open Banking technology often does not fit neatly with many legacy core banking systems, which are currently preventing effective interoperability with Open Banking APIs. Most legacy systems have a complex and inflexible IT architecture, whereas Open Banking demands flexibility, agility, and scalability. Moreover, banks' core systems are transactional in nature while the digital/Open Banking flow is based on a customer-centric journey which requires service orchestration – the execution of the operational and functional processes involved in an end-to-end service.

Online performance is another issue for core systems which have been traditionally optimized for batch performance. It is not that legacy core systems are necessarily slow, they are not. Modern enterprise-grade core systems can easily manage a 10,000 transaction per second (TPS) event if they are having to process a batch file. The issue for the core system is more having to manage a 10,000 TPS event in real-time online, if it is coming from 10,000 people that all want to check their accounts at once. In this case the digital interface is likely going to face an unacceptably slow delay in receiving a response.

To manage the shift towards Open Banking, core banking systems need restructuring. However, core banking platforms are notoriously difficult to manage and to upgrade. Furthermore, banks are reluctant to touch them given that they are the hearts of the bank. If things go wrong, the bank could lose millions. While banks have tried to put multiple fixes in place, the backend of their core system is still a COBOL based banking system. They have been able to orchestrate it so that they can talk to APIs, but this is still a very suboptimal solution.

System Elasticity and Single Point of Failure

Banks looking to integrate their services with external APIs have to be able to provision for demand fluctuations, which are often difficult to predict. That means that their systems have to be resilient to cope with these sudden surges and any additional services that the bank may want to add around their core systems. If a bank's systems fail while delivering a service which is integrated with a third-party, it is very difficult for a bank to rectify the issue with the end user and a bad customer experience can ultimately damage the bank's reputation. Banks therefore need to find a way for their backend and core banking systems to cope with the high volumes and network traffic that API queries can create.

Where legacy systems typically struggle is not so much with vertical scaling – increasing or decreasing computing resources, as banks can typically just add computing power to an existing system. Rather, what

banks struggle with is horizontal scaling – where a sequential piece of logic is broken down into smaller pieces so that they can be executed in parallel across multiple machines. To understand the benefits of horizontal scaling we need to look at resilience, single point of failure, impact, and elasticity.

Starting with resilience, the problem with a monolithic core system is not so much that it is typically not resilient. Core systems have in fact been historically extremely stable, mostly because traditionally these were largely left alone with little change occurring on the system. However, because these monoliths are all or nothing, when they do fail, everything fails, as the DBS outage back in 2010 illustrated.[3] The impact to the bank is therefore huge and the penalty from the regulator for not recovering your services within the allocated timeframe is substantial.

Digital/Open Banking revolves around the addition of services for a bank's customers, however as each additional service is added around the core system and the number of sessions increase, the risk of an individual service and therefore all the services going down increases. Such a monolithic core system introduces then what is known as a single point of dependency or single point of failure. While the risk of the core system failing may be small, the impact of a failure is enormous.

This then brings us back to horizontal scalability. If you can add parallel processes, which can be distributed across multiple machines, it will decrease the risk of a single service going down as the load can be distributed. In addition, parallel machines can continue to offer a service even if it fails on a single machine. That makes recovery easier and less costly in terms of potential penalties from the regulator. But the other major benefit is that if a service does go down, it will not then bring down all your other services as it would on a monolithic core system.

Achieving horizontal scalability, is however, architecturally challenging. That is because guaranteeing atomicity across a highly distributed system is difficult. Atomicity prevents updates to the database occurring only partially. Such updates must either succeed completely or fail completely. The alternative, leaving every system to update asynchronously, introduces a high risk of errors happening. To ensure atomicity, consistency across systems therefore must be enforced. Satisfying this property across a highly distributed system is hard.[4]

However, horizontal scaling is ultimately superior in terms of business process, recovery, availability, and system elasticity. In the end, even the best mainframe in the world eventually reaches full capacity. With the advent of digital banking services, it is critical that a core banking system today can be in the position where if it needs more capacity it can just provision for more resources on the private or public cloud, which horizontal scaling allows. That is real elasticity.

Design Issues

Pseudo Microservices Preventing Scaling Through Microservice Architecture

For banks adopting a Domain Oriented Microservices Architecture type solution to overcome the constraints of their core systems, a key challenge is not simply breaking down their core system services but building a true microservices architectural layer. Not only is doing so a challenge, but not doing so properly also brings with it its own issues.

Let us start with why this is an issue if not done properly. Often banks have been able to break down their services and deploy each of these to be technically self-contained within a single container and container image, which can be deployed automatically. But what the platform then cannot do, which a true microservices layer can, is to then take this container and create *n* stateless instances of the process which can run in parallel, from a common and consistent data store.

3 Thomson Reuters, "Singapore Bank Suffers Massive IT Failure." Reuters, July 7, 2010. https://www.reuters.com/article/urnidgns852573c4006938804825775800 0f69f1-idUS213136691820100706.

4 In database systems, atomicity is one of the ACID (Atomicity, Consistency, Isolation, Durability) transaction properties intended to guarantee data validity despite errors, power failures, and other mishaps. See Wikipedia, "ACID (Atomicity, Consistency, Isolation, Durability)." Wikipedia, March 22, 2021. https://en.wikipedia.org/wiki/ACID.

However, building a true microservices architectural layer is technically an order of magnitude more challenging than just breaking down your services – but it is also the reason why such microservices favor horizontal scaling. That is because building true microservices requires banks to think about their data store, which also must be broken down and distributed for horizontal scaling to work.

To understand why, one must understand that to build true microservices, banks must also stop their reliance on what is known as "session stickiness." This is a solution whereby the requests for a particular session are routed to the same physical machine that serviced the first request for that session. Horizontal scaling does not allow for this as that would mean that a user's sessions could not be serviced from elsewhere within the bank's application software.

To therefore build true microservices and to enable a user's sessions to be serviced from elsewhere banks must adjust both how they program and the application logic. Everything related to the user session itself needs to be managed inside the data cache, not inside the application process. The data cache itself is then replicated across the bank's stack to enable it to be accessed by any number of microservices running on different virtual machines or real servers. A function of horizontal scaling.

If you cannot break down and distribute your data store in this way, then you will not have genuine microservices and that means you will not be able to have horizontal scaling. Because doing so is so challenging banks are therefore advised to only break down those core system services that are really facing issues of scale into these genuine microservices.

If banks were to take their applications and rearchitect them to be truly cloud-native based on real microservices they would be able to better utilize the infrastructure that they have on premises and see savings in terms of infrastructure cost. That is because banks currently need to provision in additional capacity requirements to account for surges in demand for their digital services. By not building genuine microservices to enable horizontal scaling, which utilizes the cloud's elasticity, the banks must provision for this extra capacity on premises.

Currently, on average, most of a bank's applications use only a fraction of the infrastructure provisioned to them. That means that most of the money being spent on infrastructure, on average, is not being used. If banks could move to more cloud-native applications, increasing their on premises infrastructure utilization from, for example, 30% to 50%, that could represent millions in net savings a year in terms of infrastructure cost.

However, banks that do not have the proper microservices architecture will not be able to do horizontal scaling, which then defeats the object of using cloud, as the platform is then not able to fully utilize the cloud's capability. The process will end up running almost identically on the cloud as it does on premises and critically, banks will not be able to use the elasticity of the cloud, which is its biggest attraction. They will still have to allocate a dedicated resource on premises in case the process needs to consume it.

Some of the more mature organizations in APAC have realized this and are now being more selective in terms of which workloads they want to put in the cloud. They want to see that there will be a direct benefit in running on the cloud model and that they are not just moving to the cloud for the sake of it.

Customer Centric Process Design

When the Monetary Authority of Singapore (MAS) published its "API Playbook" in November 2016, many local banks rushed to build APIs on every system they had to be able to claim bragging rights for being the fastest bank to externalize their APIs or the bank to offer the most external APIs. However, despite the numerous APIs exposed by banks in Singapore, their actual utilization remains comparatively low.[5]

The main problem, which is a general industry issue, is that most APIs are not closely associated to a business process understood by a third-party. This makes them hard to consume. The APIs are often not designed with customer flow in mind, but rather the backend system in mind.

This is the crux of the issue and highlights both the versatility of APIs but also the failure of banks to unleash their full potential. By taking a customer-centric approach to API design, banks can leverage their power to deliver a memorable customer experience. To do this well, however, developers must think in terms of channel and customer journey.

Banks' developers need to be able to articulate the customer journey across each of their channels. To do that, banks need to understand the context for the customer interaction and what the channels are used for, which will help define each channel's data and action needs. For each use case, the customer journey must also be consistent across the channels. With that in mind, once banks have understood the customer journey and what it takes to deliver it, they can then design the API to consistently deliver the data or actions required to service the application or person interacting with the customer in that channel to deliver that journey.

5 Eroglu, Hakan, "The Asia-Pacific Way of Open Banking Regulation." Finextra, June 20, 2019. https://www.finextra.com/blogposting/17396/the-asia-pacific-way-of-open-banking-regulation

To that end, Mr. Arora of Standard Chartered says that the Bank has undertaken an "API first" approach to deliver the right business solutions. He says:

> *"Before creating any functionality, we think of exposing the corresponding capability or interface via APIs. For example, while building our state-of-the-art payment system, we thought beforehand what APIs will be invoked by external and internal clients, like payment processing or payment initiation etc. The building blocks of functionality then followed as per the interface defined during the API design phase."*

Ultimately, the customer journey is a process, which a single microservice is not going to be able to deliver. Instead, the customer journey is going to be served by an amalgamation of microservices, guided by the appropriate business logic. Building out and orchestrating those microservices to deliver that customer journey is no easy task. Indeed, Carmela Castelao, Head of Global Open Banking Program and Jose Navarro, Strategy Global Open Banking at BBVA spoke of the challenge of orchestration, asking:

> *"Which services are required to open an account and in what order? That may differ for each bank, as they will have different granularity of services, but bottom line, it is about making sure that the relationship front to back is the correct one."*

Sandeep Malhotra, EVP of Products & Innovation at Mastercard also agrees that delivering the correct customer journey is difficult. He says that:

> *"Microservices need to be combined and made into a composite service to fulfill an experience. But that is the problem with "bite-sized" microservices. Together they form a 'meal', but what if the first course is good and the second is bad? That will lead to an overall bad 'dining' experience."*

Instead, Mr. Malhotra says, "Banks should design micro-experiences rather than microservices, ensuring that each course stands out on its own."

The representative of an Australian digital bank that we spoke to for this report also highlighted the challenge of orchestration, saying that the structure of the data assets and mapping of data sets is the Bank's biggest challenge:

> *"Often data needs to be applied across multiple systems on the same single API request. When a partner requests the Bank to create a customer account on our platform, the orchestration of data, entitlements, and permissions through the use of multiple microservices, and the sequencing of those calls throughout our ecosystem is a challenge."*

The Bank's representative says that they must populate that data across their core banking system, into their customer records, and into their financial records. They also have to register all users across their security systems, and they need to securely validate their external partners and vendors. Orchestrating this correctly, the representative says, has been one of their biggest difficulties.

Such orchestration, however, often does not get properly addressed. Without a business process orchestration layer which not only sequences a number of microservices correctly, but also contains the business logic to provide a genuinely digital business process, banks will not be able to deliver that end-to-end customer journey, nor will they be able to create a digital bank.

Afterall, without business process orchestration, banks would struggle to do digital onboarding and the experience would be suboptimal at best. Furthermore, while many banks claim to have an orchestration layer, this usually refers to a purely technical orchestration. That is quite different from a business process orchestration, which contains the business logic that allows your business process to be digital.

Mr. Malhotra of Mastercard concurs. Mastercard has recently studied designing a super-app lifestyle application (Banking-as-a-Service) for banks, an effort that has required the company to understand how to orchestrate services coming in from both the banking and the non-banking domains. Mr. Malhotra says that business orchestration has been the primary and biggest challenge, and it has to be approached separately from the technical orchestration.

He says that there is often a disconnect in terms of pure technical orchestration and business orchestration, in part, because banks have built their technical and workflow systems in an evolutionary manner, creating multiple platforms over time. The most effective way to pull them together is to orchestrate around a desired micro experience for the consumer. "Defining the goal as an experience rather than a service more naturally catalyzes the digital transformation of systems and processes required to deliver a refined user journey," says Mr. Malhotra.

If banks want to digitalize and open up their services, they should consider following a process-centric API development strategy. Here the business process should be built first, followed by the APIs, otherwise the business process and therefore the customer journey will be suboptimal. To that end, at least for external APIs, banks must always put the customer journey front and center. That means that FIs should look at the API initiative not purely from the technology perspective but more from the end customer perspective.

The representative from the digital bank, however, says that that business process orchestration to deliver on the end customer journey is particularly challenging when you are building Banking-as-a-Service. The representative says:

> *"It is easier if there is only one app, one set of customers, and one context, but our challenge was how do you satisfy many partners and their customers in the context of the problems that they are trying to solve and the context of what those partners are trying to achieve?"*

Reliance on Third Parties to Deliver a Consistent Quality of Service

One of the challenges with Open Banking and working with external partners is that sometimes delivering the required customer journey is dependent on the quality of services provided by your partners. It only takes one of those services to be bad to have a knock-on effect on the customer journey that you are looking to deliver, which goes back to Mr. Malhotra's dining experience analogy.

Sticking then with Mastercard, the following highlights the potential issues that could arise when consuming external partners' services. The company works with Tesco Bank in the UK, to enable Tesco Bank consumers to come through Tesco Bank's platform to pay their credit card balance from any HSBC, Barclays, RBS, or Lloyds bank accounts that they have access to. The service displays the customer's account information across these banks (aggregation) and enables the customer to determine which accounts they want to use to then make a payment towards their credit card bill.

The challenges for Mastercard in delivering a good customer journey here are severalfold:

- **Availability:** All services need to be available at all times. What happens if, for example, HSBC's system is up, but the Lloyds' system is down for regular maintenance? This would result in a suboptimal user experience.
- **Data integration:** Not only must the exchange of data be seamless, but it is also important to know who has access to the data and where the request is coming from – has it come from a domestic or an overseas server? Knowing the provenance of the request is important in order to comply with GDPR and other local regulations.
- **Cross-dependencies:** Availability becomes even more complicated when you are relying on several services from each partner. In our example above, what happens if I can see my account information from HSBC, but I am unable to make a transfer from my account there. The problem here is a result of the different systems involved in delivering the overall service that Mastercard is offering. The account information display is served by a microservice coming from a shadow account on one system, whereas debiting happens on an ATM system. One system may be up, while the other may be down, meaning that you therefore cannot complete the entire experience.
- **Latency:** The other issue in completing an entire experience relates to the different polices applicable to different systems, which mainly revolve around latency. What if one system takes longer to complete a service than another? That will impact the overall performance of the service, which impacts the customer's experience – which again takes us back to the dining analogy.

Data Issues

Data and its consumption lie at the heart of digitalization and Open Banking. However, many incumbents still struggle with dirty data, which manifests in a multitude of different forms – duplicate data, incomplete data, inaccuracies, or inconsistencies. The patchwork of legacy systems that most banks operate has employees entering data into separate data silos which often rely on separate systems. The result is duplicate data which is often in different formats. Nearly half of banks told researchers in a survey that creating an integrated, consistent view of their data across the organization is their largest data management challenge.[6]

Addressing these issues with dirty data is critical for API developers and for the entities that rely on the accuracy and consistency of such data. Take, for example, digital onboarding. If there are duplicate records of the same customer, where is the API developer meant to take the customer's data from? There is no single source of data for that customer. Banks should therefore take the opportunity when building their Open Banking layer to standardize their data flows to one single source.

6 PYMNTS, "Banks Kept On Their Toes By Dizzying Data Management Regulations." PYMNTS.com, December 5, 2017. https://www.pymnts.com/news/b2b-payments/2017/wolters-kluwer-bank-data-management-regulation/.

However, because of the expense involved in building a single layer which would require synchronizing/ sanitizing all existing data, banks tend to opt for cheaper, easier alternatives. Take identity management for example. When introducing a single sign-on (SSO), instead of authenticating against a central data source where customer credentials are stored, banks will simply introduce yet another data store in the digital layer, which the SSO can authenticate against. The problem with this solution is that it does not clean up other sources of information on the customer that might exist, including the dirty sources. Banks, therefore, need to go further and have a unified data layer that provides a single source of truth on the bank's customers. Having such a unified layer mitigates against new applications being forced to integrate with one of the many, disparate identity data silos.

Security Risks/Lapses

With 76% of today's banks citing customer data security and privacy as their top Open Banking concern, protecting the integrity of external APIs is therefore critical.[7] Such protection is going to require incorporating security at the infrastructure level with a multi-pronged approach.

Mr. Malhotra of Mastercard emphasizes that your security and privacy must be first-rate. Trust, he says, is a critically important element in differentiating players in Open Banking. This is why Open Banking developments are often accompanied by in-market data privacy and API standardization regulations – to ensure a baseline of consumer trust in order to drive adoption of solutions at scale. The winning players will both meet and strive to exceed those standards.

7 https://www.hydrogenplatform.com/blog/what-is-open-banking

With security being paramount, Mastercard has a product called Open Banking Protect that makes sure the Third-Party Provider (TPP) is who they say they are. In this case the TPP is Tesco Bank but tomorrow it could be Google. Tesco Bank calls Mastercard's Open Banking Connect API, which checks that it is a valid TPP by seeing whether it has the right security certificates to do an "Individual Auth." Mastercard then applies further forensics to see whether the request is coming in over a public or a private internet and which IP it is coming in from.

Once the user has been authenticated, Mastercard checks to see if the user has passporting rights to reach these end customers. Mr Malhotra says, "this is not simply a question of security. Mastercard has rules around who has access to what data, what services they have subscribed to, and the data that consumers have provided consent to." For example, he says, "there could be a request from a valid TPP, such as a hypothetical Tesco Bank in Singapore, but for access to data that they do not have passporting rights to use."

Developer Tooling to Enable Third-Party Development

Developer Portals

It is important for banks to understand that APIs are an ecosystem and not just a set of endpoints that they put out there. The Developer Portal is an essential medium that banks can use to communicate with developers, while providing them with the tooling that they need to help them succeed in leveraging their APIs. It enables the bank to say to the developer, this is our catalogue of APIs, this is how you can report to an API, this is the full documentation about the API, this is how you can test it, here are some software development kits (SDKs) and sample code, here is a sandbox that will let you play around with it all, and here is where you can get help. It is also important that banks provide proper and relevant data for testing. Often sandboxes do not include enough sample data use-cases to validate APIs.

Mr. Arora of Standard Chartered concurs that Developer Portals play a critical role, saying:

> *"To boost API usage and penetration, they need to be made discoverable internally and externally via respective Developer Portals, which allows developers to try them out and enables a smooth transition into the production environment that accelerates adoption."*

Ms. Castelao and Mr. Navarro of BBVA however say that once a bank has successfully "front to back" digitalized their products, it becomes critical to deeply understand the customer's needs to guide them to the right solution rather than leaving the customer to search for the solution by themselves. This requires an expert sales team, with a specific focus in APIs.

They say that in this regard when BBVA publishes its APIs, they put together use cases to highlight how the Bank has solved problems for other companies in the past. That then enables developers to see how a problem that may be similar to theirs has been solved. It also creates an environment for discussion. It enables developers to say, "I want something similar, but I want it like this, and I also have this other problem." The Bank then has its experts that know what the Bank can/could provide. It is therefore not only about helping the customer correctly use the APIs that the Bank offers, but it is also about creating new services on top of what there already is.

Arvie de Vera, SVP & Head of the Fintech Group at Union Bank of the Philippines (UnionBank) agrees that helping the customer is key. The Bank was the first local bank in the country to expose its banking functionalities via APIs and promote Open Banking and collaboration. The Bank has a dedicated API team that proactively pitches and explains the different API products to potential clients. This helps product discoverability for potential new API solutions for emerging business problems or Open APIs that can be productized by the client. Mr. de Vera says that the Bank's platform is tailored to the needs of users and is focused on the overall customer experience. The API Developer Portal uses a Netflixlike recommendation algorithm to suggest relevant APIs to users based on the customer's industry and the APIs that they have already subscribed to.[8] Mr. de Vera says, "comprehensive yet simple to use business and technical documentation is crafted per API so that partners can jumpstart their integration then and there."

Documentation

Because documentation or a lack thereof can be a major obstacle to preventing API utilization and penetration, it is important that we drill down a bit further to see why that is. Ultimately, poor documentation can lead to time delays and buggy implementations. Documentation therefore needs to be part of the API design process, whereby the developer needs to think about how they are documenting the API as they are building it.

Because developers do not know who will be consuming the API – it could be experienced users or someone completely new to programing – they should always keep the end user in mind and provide as much information as possible, including where to get help. Documentation must also be in synch with any changes to the API, so how it is going to be maintained should also be a part of the API design process.

Given its importance it is therefore remarkable how often documentation is either out of date, incomplete, inaccurate, or just not useful. Poor documentation comes in myriad forms: Endpoints of the API are left out, there is no information on the data that needs to be sent, and/or there is no information on the response in terms of what data you will receive, what it represents, and how it should be handled.

In fact, it is this lack of information on the response which is often the key weakness in documentation. Specifically, the error messages that you can expect to receive and how to handle these. Good API design should therefore incorporate standard exceptions, whose handling should also be standardized and explicitly defined in the documentation.

Ultimately, the process of the API needs to be clearly communicated to the developer in the documentation so that it is easy to understand. The bottom line is that while writing a good API is very complex which makes it difficult to standardize, banks can standardize the data format and behaviors of an API. Banks can also standardize their documentation and handling of exceptions. From a developer's point of view, they will then know how the API is going to behave.

8 developer.unionbankph.com

Conclusion

Disruptive forces across the region are forcing banks to open up their platforms in order to consume other players' services or enable the external consumption of their services by other players. Digitalization is also forcing banks to think about how they can offer their services online. Ultimately, this is being driven by competition and customer expectations, which are pressuring banks to understand the customer better while offering more personalized and contextualized services.

For banks to get this right, they are going to have to move away from a centralized integration approach to a distributed Open Banking model. This will require a shift from a monolithic, legacy architecture to a cloud-native, microservices-based architecture. Such a shift will also require a new organizational structure, from traditional, waterfall methods to agile and DevOps approaches.

That is because legacy core systems do not fit well with digital/Open Banking requirements. However, upgrading core systems is an expensive, risky, and lengthy process. Building a more modern, scalable infrastructure that sits on top of these legacy systems provides a solution that enables banks to become modular and platform-based in order to effectively collaborate with players in the financial services ecosystem.

However, this abstraction layer is difficult to build and to get right. Banks will want to think about creating a true microservices layer to enable horizontal scaling. While such horizontal scaling is difficult to achieve, it does provide resilience, helping to mitigate against disastrous system failures, and will enable the bank to leverage the elasticity provisioned by the cloud. That in turn will allow banks to save substantially on infrastructure costs.

Banks must also ensure that APIs have been designed with the end customer in mind rather than the backend system in mind. That will require articulating the customer journey across all channels of the bank to understand the context for the customer interaction and what the channels are used for. Delivering that customer journey will then require a business process orchestration layer that is not simply able to sequence a number of microservices correctly, but also contains the business logic to provide a genuinely digital business process.

Getting Open API Banking right is challenging and will also require banks to think about how they work with external third parties, manage their data, and security. Importantly, if banks want to ensure that their external APIs are utilized, they will also need to work closely with consumers through their Developer Portals to ensure that these are finding the right solutions. They will also want to work with these developers to add further services where required. Finally, banks must ensure that their documentation includes as much information as possible so that the developer understands the API process, its behavior, and how to handle exceptions.

By having the correct architecture and organizational design in place, banks can leverage digitalization and Open Banking to engage with their customers in ways that reach far beyond what they have been able to do to date. That is ultimately how banks are going to retain customers and win new ones and emerge as victors in the new normal.

Red Hat

Red Hat is the world's leading provider of enterprise open source software solutions, using a community-powered approach to deliver reliable and high-performing Linux, hybrid cloud, container, and Kubernetes technologies. Red Hat helps customers integrate new and existing IT applications, develop cloud-native applications, standardize on our industry-leading operating system, and automate, secure, and manage complex environments. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500. As a strategic partner to cloud providers, system integrators, application vendors, customers, and open source communities, Red Hat can help organizations prepare for the digital future.

Please visit https://www.redhat.com

Kapronasia

Kapronasia is a leading provider of market research covering fintech, banking, payments, and capital markets. From our offices and representation in Shanghai, Hong Kong, Taipei, Seoul, and Singapore, we provide clients across the region the insight they need to understand and take advantage of their highest-value opportunities in Asia and help them to achieve and sustain a competitive advantage in the market.

Please visit https://www.kapronasia.com