The seven make-or-break API challenges CIOs need to address

Keerthi Iyengar, Ling Lau, Srinivas Ramadath, and Vik Sohoni

Application programming interfaces hold great promise for tech modernization—if CIOs develop a coherent strategy in advance.

To compete in the digital age, large enterprises need scalable and flexible IT systems to allow for ongoing experimentation and iteration at speed.[1] As a result, many companies have launched programs to modernize their technology.

Application programming interfaces (APIs) have emerged as a key element of tech modernization at many businesses, including most banks and insurers. With their ability to link systems and data, APIs play a crucial role in making IT systems more responsive and adaptable.

[1] Albert Bollard, Elixabete Larrea, Alex Singla, and Rohit Sood, "The next-generation operating model for the digital world," March 2017, McKinsey.com.

Yet despite the promise, most enterprises have failed to capture the value they initially envisioned from APIs. In many cases, a rush to build APIs without a thoughtful strategy has created a mess, with redundancies, poor maintenance practices, and limited transparency canceling out many of the potential benefits. Some companies have spent years ripping and replacing megasystems, adding APIs in an ad hoc way the entire time, without making any real progress.

In our work with many clients on developing API strategies and building out API portfolios, we've found common themes behind the difficulties companies encounter. One is that IT assumes sole ownership of API programs, so they aren't tightly linked to business goals or an overarching modernization vision. Another reason API programs falter is siloed efforts—for example, efforts focused solely on data or cloud migration—which can generate only incremental value. To achieve the full promise of APIs, companies ultimately have to make significant progress along multiple dimensions.

To avoid these pitfalls and get the full benefit from their API initiatives, CIOs must address seven key challenges.

Challenge #1: Where do I start?

"We're not sure where to start with our API program."
—Digital-transformation leader at a large bank

The flexibility of API means that it can be used almost anywhere in IT to make something better. That range can be overwhelming, and small experiments can take on a life of their own, leading to wasted energies and limited value.

Deciding which APIs to build requires taking a dual perspective: identifying which can enable important customer-facing applications/solutions and which can build a sound technical foundation. Within that context, IT leaders can prioritize API development based on the business's strategy, business and modernization impact, and ability to execute.[2]
For example, a regional bank began with APIs that digitized a key customer journey. It then went on to build APIs that simplified the architecture and drove efficiencies.

Customer journeys are a good place to start thinking about which APIs to build. A focus on journeys can identify which APIs can stitch together disparate IT systems to deliver a seamless end-to-end customer experience. To maximize the impact, tech leaders should also consider three additional ways APIs can provide value.

[2] Keerthi Iyengar, Somesh Khanna, Srinivas Ramadath, and Daniel Stephens, "What it really takes to capture the value of APIs," September 2017, McKinsey.com.

Data access and integration

APIs can expose data across various IT systems, both legacy and modern. They can then be used to extract the data and link them to advanced analytics systems.

Cloud migration

Most large organizations still have significant captive, on-premise infrastructure and have started to migrate to private clouds. A small number are embarking on public-cloud migration to reach the next horizon of benefits. Managing a hybrid cloud environment requires IT leaders to prioritize APIs that can access infrastructure across both private- and public-cloud assets.

Core transformation

APIs can help with transformations because they can connect core IT systems, and the specific data inside them, to digital platforms. APIs can also serve as connective tissue and a synchronization mechanism across both modern and legacy systems that may operate at different clock speeds.

All of these elements are intimately related. Thus figuring out which APIs to build first requires mapping the most important journeys and then noting the required complementary APIs to unlock the needed data, cloud, and core resources. It also requires identifying APIs that will allow the organization to operate in an agile, test-and-learn mode.

Challenge #2: What should my API teams look like?

"We decided to implement APIs using agile but had a hard time making it happen."
—Transformation leader at a large insurer

The teams that build APIs need to employ a dedicated agile operating model. This requires a shift from seeing API building as a set of one-off projects and instead treating each API as a product that will evolve and improve over time based on customer feedback. The two core elements to get right for the operating model are teams with the right skill sets and an organizational structure that supports the business needs of the company.

API teams need a product owner to ensure that they build high-value APIs. They also need an API architect to ensure that they follow technical standards and best practices and that what gets built is aligned with IT's architectural vision—the end state envisaged when the tech-modernization program has been completed. Scrum masters ensure that teams use an iterative, test-and-learn execution model to build APIs quickly. Once the business has a prioritized backlog of high-value APIs, top-priority items get assigned to scrum teams, which then build APIs in one- to two-week sprints. Those APIs are subsequently tested and iterated with users.

Companies can choose from three organizational structures to manage API development. Aligning leadership around one of them is crucial for IT to take advantage of APIs.

Centralized

A centralized group develops most APIs for both business and technology units. This model can build a lot of APIs quickly, but the central group needs to work closely with local business or IT units to ensure that the right APIs are prioritized and that they're adopted after they've been built.

Decentralized

A set of dedicated API factory teams, embedded in different business and technology units, works cross-functionally to build and consume APIs. In this model, APIs are more likely to be aligned with business needs, but enforcing standards and best practices can be difficult.

Hybrid

Under a hybrid model, a centralized API factory builds foundational APIs. Individual business and technology units also contribute to the API catalog, following the taxonomy and standards set by the central factory and using foundational APIs built by the central factory.

Challenge #3: What do I track?

"I need to demonstrate in-year benefits with our API program."
—Transformation officer at a large e-tailer

Companies often use a variety of different and inappropriate metrics to assess API performance. Teams' activities should be assessed using a set of common agile metrics: direct business value for customer-facing APIs and reuse/reduction of technical debt for back-end APIs (Exhibit 1). Other potential metrics include developer adoption, contribution to simplifying architecture or reducing infrastructure costs, and data-specific key performance indicators (KPIs).[3]

Challenge #4: How do I ensure my APIs actually deliver business value?

"How do I ensure the APIs we build enable business value and agility?
—Chief digital officer at a large telco

APIs historically were classified as middleware that integrated and exchanged data among multiple systems. Because most API efforts were housed in the IT group, the taxonomy used to classify them was usually technical and nonintuitive. This prevented business stakeholders from engaging in API design and prioritization.

Leading organizations today are defining their API taxonomy in a way that creates a common language that's understood by both business and technical

[3] Keerthi Iyengar, Somesh Khanna, Srinivas Ramadath, and Daniel Stephens, "What it really takes to capture the value of APIs," September 2017, McKinsey.com.

EXHIBIT 1 **Sample metrics for API teams**

			Metric	Description	Measurement
Common metrics	Cost	1	Cost to develop API	Average cost to develop API (factory-dev budget per API)	Calculation: factory-dev budget over # of APIs built
Common metrics	Cost	2	Cost to run API	Average operating costs per API (includes maintenance, issue resolution)	Calculation: opex over # of APIs built
Common metrics	Quality	3	Incidents per API	Average number of issues/bugs that get reported per API once launched	Calculation: issues reported over # of APIs built
Common metrics	Quality	4	Code quality	Number of errors the automated scan returns when looking for vulnerabilities	Automated scan results
Common metrics	Quality	5	Automated test coverage	Measure of how much automation on test cases is being executed	Automated test results
Common metrics	Output	6	APIs launched	Total number of APIs delivered to enable services	
Common metrics	Output	7	Active use	Percentage of APIs built that has been integrated with an app/service and in use	Calculation: # of integrated APIs over # of APIs built
Service API impact		8	Reduction of technical debt	Percentage of middleware services that use APIs	API gateway
Service API impact		9	Reuse of API	Average number of apps or services integrated per API	Calculation: # of integrated services over # of APIs built
Product API impact		10	Monetization	Revenue/profit generated per API	Calculation: revenue generated by APIs over # of APIs built

units. The key is to distinguish between APIs that directly serve the business (where business input is crucial) versus those that are technical enablers. We have seen a sound taxonomy reduce API analysis time by 50 to 75 percent, increase adoption by 30 to 50 percent, and increase value realization by 25 to 50 percent.

A solid taxonomy allows businesspeople to have conversations with IT staff about which APIs directly drive customer experiences and which are part of the infrastructure that enables delivery of those experiences (Exhibit 2).

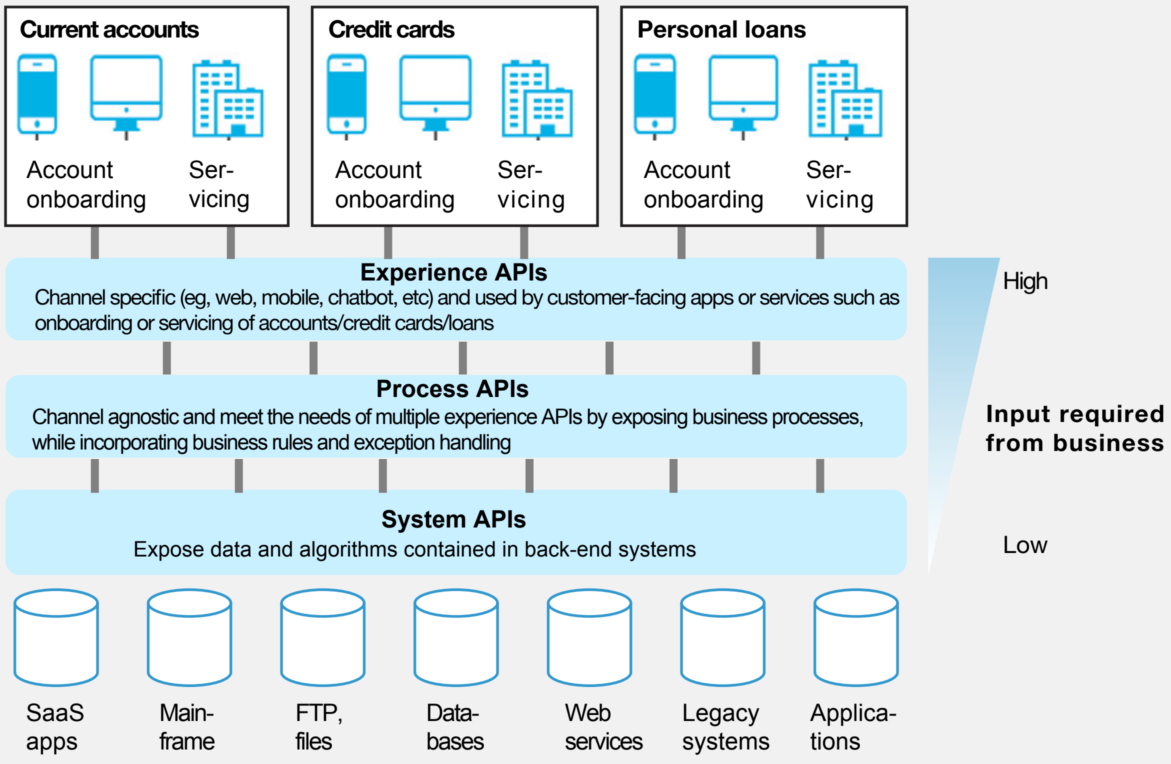
Challenge #5: What technology tools do we need?

"A dizzying array of API-related technologies are available. Which ones should we use or buy?"
—Architect lead at a global bank

IT leaders can easily get lost in the proliferation of API-related technologies, leading to wasted resources and poor performance. By making two specific decisions first, they can help narrow the field of technology options.

The first key technical decision is choosing the right application-development framework/platform for building APIs. It is important to work with business leaders and list the right use cases for the APIs, based on technology feasibility (back-end readiness, for example), the business rules around them, and error conditions to be handled. The IT organization then needs to establish data formats for the APIs and design an API reference architecture that is modular, flexible, and extensible. This will prevent creation of unnecessary technical layers and technology debt. Engineering leaders should also establish API guiding principles, educate product owners on the benefits of good versus bad APIs, and provide tool kits and API catalogs so that developers adhere to best practices and APIs are reused.

EXHIBIT 2 **Sample API taxonomy used by a large bank**


Current accounts
Account onboarding
Ser-vicing
Credit cards
Account onboarding
Ser-vicing
Personal loans
Account onboarding
Ser-vicing
Experience APIs
Channel specific (eg, web, mobile, chatbot, etc) and used by customer-facing apps or services such as onboarding or servicing of accounts/credit cards/loans
Process APIs
Channel agnostic and meet the needs of multiple experience APIs by exposing business processes, while incorporating business rules and exception handling
System APIs
Expose data and algorithms contained in back-end systems
SaaS apps
Main-frame
FTP, files
Data-bases
Web services
Legacy systems
Applica-tions
High
Input required from business
Low

The second big technology decision is around the gateway and the API developer portal that serves as the front door for an organization's API activities. The API gateway houses all the APIs, captures transaction analytics, and does data caching. It is important to decide who manages the gateway and to establish governance rules around it. Some companies select a centralized API platform. While it can be beneficial for a single team to be responsible for the gateway, that team can also become a bottleneck. A cloud-based API gateway and developer portal allows for separate units to be responsible for different sets of APIs and also allows developers to access all APIs in one place. This enables decentralized governance based on a set of published standards that apply across the entire organization.

When deciding whether to insource or outsource API development, the rule of thumb is that APIs should be kept in-house when they can create distinctive capabilities. APIs that don't enable differentiation are better acquired from outside. For example, many banks rely on external vendors for payments APIs.

Challenge #6: Who owns which APIs?

"My business should not bear the up-front investment in an API platform, especially if I cannot prioritize my own projects."
—Business-unit leader at a global retailer

Who gets to define which APIs get built—and who pays for their development—is an important and, at times, contentious issue.

Some organizations have business units define and pay for customer-facing APIs, while the technology group builds them. This works well when a clearly defined digital strategy is already in place.

Some organizations, however, are in the process of defining their digital strategy and thus are not yet ready to build customer-experience APIs at scale. If that's the case, the IT group should manage and fund process APIs and lower-level system APIs. By using this approach, the IT organization can build lower-level APIs that will already be in place when the digital strategy is ready.

The best practice is joint ownership of APIs, particularly when cross-functional digital teams are building experience APIs to enhance customer journeys. In cases like these, business units have ownership of and pay for experience APIs, while the technology group owns the process and system APIs needed at the bottom of the stack.

In all cases, a council or board should oversee the API portfolio over the entire life cycle (Exhibit 3). API councils are often made up solely of IT staff but should include representatives from the businesses as well. The council monitors the performance of APIs and decides which new ones to add and which existing ones to sunset. It also oversees agile funding of API projects, with money released gradually at first, as APIs meet specific milestones, and later, as business impact has been achieved. The council also continually reprioritizes projects in the queue, while pushing to encourage API reuse.

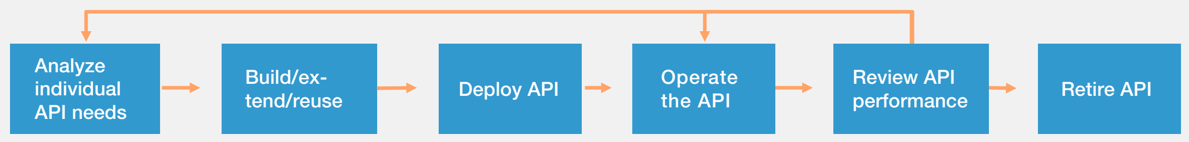
As part of its work, the council must tackle API adoption. That means that any API project needs to have a clear adoption plan to be approved. These are critical to avoid the "build it and they will come" trap. Adoption plans should be developed by business and marketing groups working closely with IT. Self-service APIs that will be exposed externally to customers or partners need to align with the relevant technical, legal, and marketing specs and also be easy to use. Internal APIs should have enough documentation and be sufficiently evangelized within the organization to promote adoption and reuse.

Challenge #7: Where is the talent to do all this going to come from?

"Talent became a bottleneck." —API center-of-excellence lead

EXHIBIT 3

Agile API life cycle with continuous improvement



Map individual **API requirements against road map.** Cross-reference API need against **existing catalog.**

Publish all APIs through **discovery portal** (in road map and nice-to-have requests).

Design, develop, and test APIs. **Where possible, reuse** and extend existing APIs instead of building net new.

Publish API to **gateway**, provide documentation, and communicate to user **community**.

Maintain and fix the API as necessary. **Provide support** to project teams that are leveraging the API.

Monitor during operation:

- Security
- Usage
- Business value
- Metering and throttling

Each sprint, **API metrics** are reviewed to **determine next steps** (continue operation, modify, retire), integrating learnings from monitoring.

New versions may need to be published, following this **same life cycle.**

When APIs become **obsolete** or require-ments change significantly, **retire current version** and support any partner teams to **migrate**.

There are two sets of decisions companies need to make on talent.

Which new skills to acquire

Companies embarking on an API program typically need to acquire seasoned practitioners with expertise in re-architecting existing infrastructure into streamlined, API-enabled systems that can be more flexibly tested and deployed. This means bringing in experts who can access monolithic cores with microservices that use open-source tools. These experts should be familiar with API DevOps processes and also be able to iterate continuously on the API strategy, development, and life cycle–maintenance activities required to drive impact. It is crucial to have practitioners with such skills in-house, since they form the backbone of a company's API approach and need to work closely with existing IT and business leaders.

Which in-house capabilities to enhance

There are two sides to building in-house capabilities: educating business leaders on the value of APIs and upskilling existing IT architects and developers so they can operate in a modern API engineering environment. Business leaders need to understand the "what" of APIs—that is, use cases that explore what is possible—as well as the "how"—the investments that will be involved, the resources that will be required to execute, and the adoption levels that will be needed to capture the full value. IT architects and developers should be well versed in the different kinds of APIs that will be needed and learn about modern open-source technologies like NodeJS and documentation tools like Swagger. They should also gain facility with core tools used in API engineering environments, such as API gateway and cloud-native technologies.

◆◆◆

APIs are no longer a matter of technology strategy alone. Today, they are also a key part of any company's business strategy. By attending thoughtfully to the seven key challenges, organizations can unlock the potential of APIs and accelerate their pathway to tech modernization. ◆

Keerthi Iyengar is an associate partner in McKinsey's Sydney office, **Ling Lau** is a partner in the New Jersey office, and **Srinivas Ramadath** is an associate partner in the Chicago office, where **Vik Sohoni** is a senior partner.