apigee

Google Cloud

Inside the API Product Mindset

PART 1

Creating World-Class Developer Experiences

2

3

4

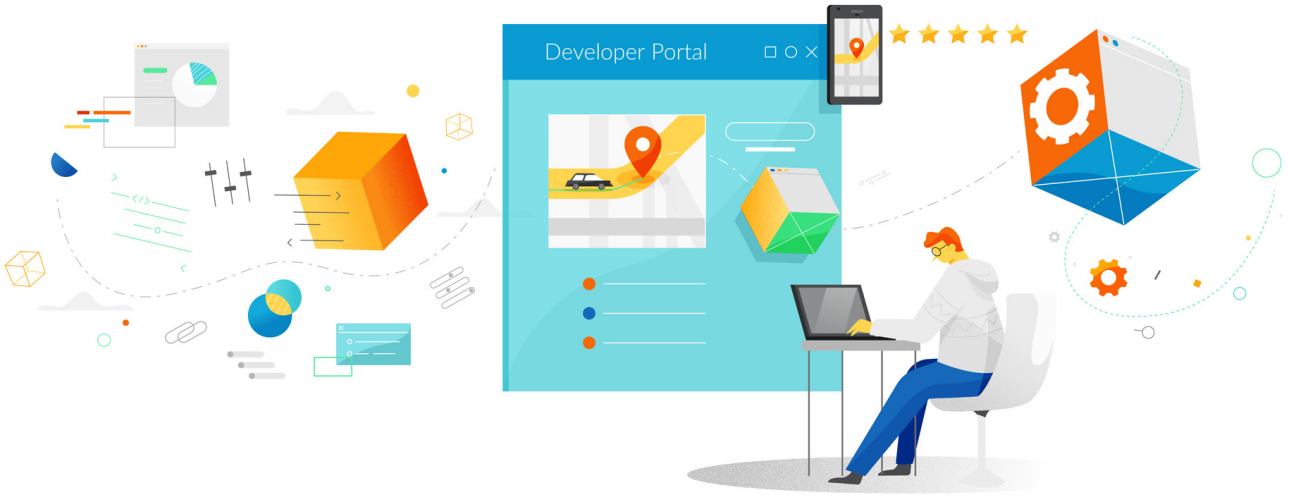
- Field-tested best practices
- Real-world use cases
- Developer experience checklist

Table of contents

Inside the API product mindset

Application programming interfaces, or APIs, are the de facto mechanism for connecting applications, data, and systems—but they're also much more.

APIs abstract backend complexity behind a consistent interface, which means they not only allow one kind of software to talk to another, even if neither was designed to do so, but also empower developers to leverage data, functions and other digital assets both more efficiently and in new and novel ways.

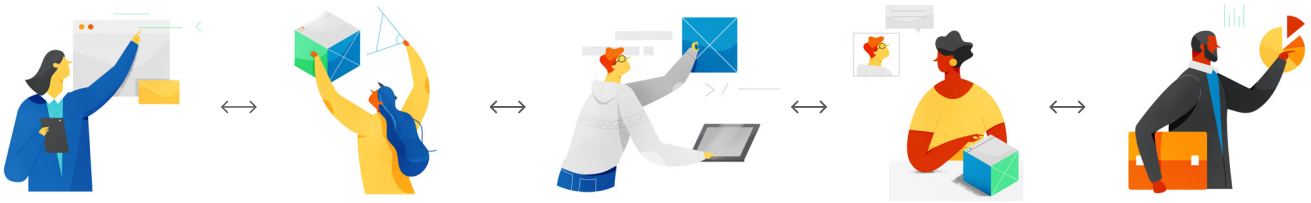


Developers have launched new ridesharing services by combining third-party mapping and navigational APIs with their own first-party data and functionality, for example. Many applications rely on third-party APIs, such as those from Twitter or Google, for authentication. When developers want to add voice commands to an application but lack the time, incentives, resources or expertise to build their own natural language technologies, they can turn to APIs to get the functionality they need.

Because they make digital assets easier to reuse and combine, APIs are in the middle of virtually every digital use case, as are the developers who leverage APIs to make those use cases possible. Many enterprises now view developers as one of the most essential actors in their value chains—the people who translate digital assets into digital experiences that move the business needle.

As businesses have realized that APIs can be pivotal to their evolution and growth, they have increasingly begun to manage APIs like products, supported by full lifecycles, long-term roadmaps, a customer-centric approach, and constant iteration to meet business needs. In our experience on Google Cloud's Apigee team, organizations that treat APIs as products—as opposed to one-off technology projects—are more likely to win with developers and realize the potential value of APIs as business accelerators.

As we explored in *The API Product Mindset* ebook, typical roles within an API product team include an **API Product Manager** who owns the pro- cesses and cross-functional coordination critical to the API's success; an **API Architect** responsible for designing and guiding the creation of APIs; an **API Developer** who builds APIs from the API Architect's designs and implements security policies and other protocols; an **API Evangelist** who serves as the voice of API consumers and owns partner and developer outreach; and an **API Champion** who works closely with internal ex- ecutive sponsors to communicate the value of the API program to the rest of the organization.



API PRODUCT MANAGER | API ARCHITECT | API DEVELOPER | API EVANGELIST | API CHAMPION

The API product team typically carries several critical responsibilities:

- Design secure and easy-to-use APIs and bring them to market
- Deliver a world-class API developer experience
- Drive ongoing API improvements with monitoring and analytics
- Maximize the business value of APIs through API monetization, ecosystem participation, developer evangelism, etc.

This eBook takes a deeper look at creating world-class developer experiences and shares best practices Apigee has observed for engaging developers and deploying a developer portal that fuels those experiences.

Understanding developers—internal and external—as API customers

APIs may be intended for different users or purposes. Broadly, the three types of API products are:

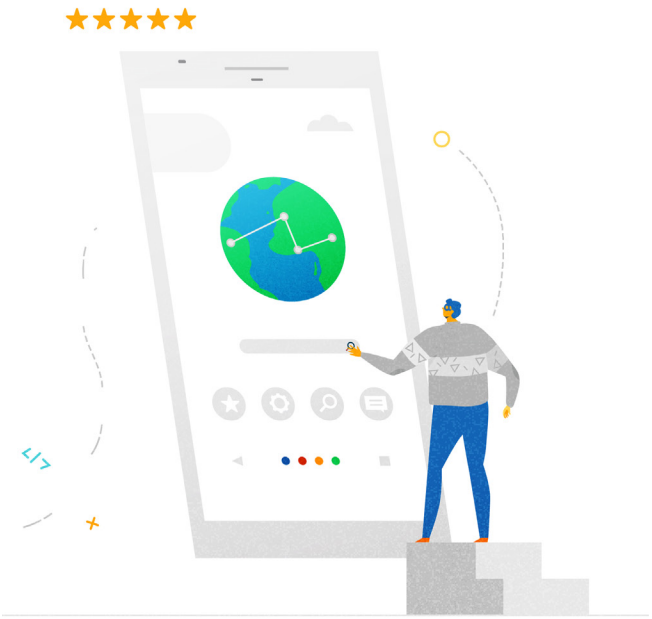
- **Public:** The API is available for anyone to consume, e.g., a branch locator API. Enterprises typically use public APIs to either monetize valuable proprietary services that may be of use to outsiders or to encourage adoption of their services within new developer communities that can help the enterprise reach new digital ecosystems.
- **Private or partner:** The API is available to select partners but not to external developers at large, e.g., an API to find out the availability of a company's financial advisers or APIs that are specially created for partner needs.
- **Internal:** The API is available only to certain employees within the enterprise, e.g., an HR database API that provides access to employee information.

For many enterprise leaders, it may be tempting to apply different management philosophies to different types of APIs and to view developers outside the company as customers while viewing developers inside the company as employees. These types of distinctions can be dangerous.

For instance, companies are increasingly mixing their internal-facing APIs with those of publicly-available APIs from third parties. By combining APIs in this way, companies can supplement their proprietary capabilities without having to proportionately invest in research, development, and staffing. These APIs also help businesses to more efficiently leverage valuable assets without bespoke considerations for each project. The end experience that these API combinations facilitate must be seamless to the user, regardless of whether the combinations involve public, partner or private APIs and regardless of whether the end user is inside or outside the organization. If some of the APIs are managed as products while others are managed as middleware, this cohesion may be at risk.

Retailers, for example, often combine their internal inventory APIs with external payment processing APIs and external shipping and logistics APIs in order to create seamless mobile e-commerce experiences. These retailer applications sometimes incorporate external third-party location and navigation APIs to offer functionality that lets customers pick up a purchase at a nearby store rather than have it shipped to them. Applications might include machine learning APIs to support voice interactions or create better recommendation engines.

The possible combinations are endless—but the point is, to a retailer's customers, the overall experience is the product of many interactions from many parties and many APIs, including those owned by the specific retailer and those owned by others. For the end user, the distinction between internal and external APIs is not particularly meaningful—they all need to be managed to provide great experiences for developers so those developers can provide great experiences for users.



Similarly, an API used for purely internal purposes may prove itself so useful that it becomes a candidate for externalization. But a business's awareness of the internal API's value may be itself largely predicated on the API being managed and monitored like a software product rather than a middleware detail. If internal-facing and external-facing APIs are approached differently, with the enterprise paying attention to usage patterns and customer needs only for the latter, the company may never understand which of its internal services could produce value if made accessible outside the company.

Finally, even though many businesses invest in great experiences for external developers who pay for monetized APIs or spread an enterprise's services to new use cases, some of these organizations give internal developers a second-rate experience, saddling them with manual approval processes, missing documentation, and other productivity-killing frustrations and potential security risks. It's counterproductive to recognize the value of well-supported external developers while neglecting to support internal talent.

Many large enterprises are accustomed to working in silos, for example, which means, among other things, that one team creating APIs in one part of the organization might not be aware of APIs being created by other teams throughout the organization. This can lead to work being needlessly duplicated and to APIs that might be simple for one team to use but confusing and prone to security vulnerabilities for another team's uses. In these organizations, internal developers are too often stymied by cross-team confusion and heavy governance models. Without an API team and a unifying product mindset presiding over all APIs, these internal developers may not achieve the status of "customer," the APIs they use may not achieve the status of "product"—and an organization's productivity and innovation could suffer as a result.

Field-tested best practices

Build an easy-to-use, self-service developer portal to drive adoption

The faster a developer can go from accessing an API to creating a new experience or service, the more likely they will be to adopt the API. Creating a branded, self-service developer portal—i.e., an online API storefront—is one of the most powerful ways to provide this access, regardless of whether the API customer is an internal or external developer.

A successful portal experience often starts with an API directory where developers can begin exploring. APIs should be not only accessible but also supported by documentation, sample code or SDKs, and even sandbox environments that make it easy for developers to start testing and experimenting. Beyond simply issuing API keys to developers, the portal may allow customers to purchase API products and packages tailored for different levels of functionality and traffic overhead.

Above all, a developer portal should clearly express the value proposition of a company's APIs and inspire excitement about how developers can use them. Effective portals eliminate barriers that impede developer productivity.

As Thomas Squeo, senior vice president at telecommunications firm West Corp, recently stated, developer portals are "the primary interaction point where [a developer] can go from awareness to activation to acquisition for an API and be able to bring it up to a 'Hello World' within maybe 30 minutes."

SECTION SUMMARY

Build a store for your APIs

- Enable self-service. Make it easy to get started. Reduce time to "Hello World."
- Go beyond access to APIs. Make the value proposition clear. Include documentation, sample code, and testing environments. Reduce complexity and barriers to entry, as with any product sold online.
- Craft developer experiences that provide the same support and care for both internal and external developers.

Create a community of developers

Peer-to-peer recommendations remain one of the most powerful forces in the software world. A thriving developer community can help an API program grow by creating buzz that encourages new developers to enter the community and by enabling feedback loops that help the API provider improve its program.

A developer portal may play a central role in fostering community. By including blogs and forums in its portal, an enterprise can not only help developers to share best practices around the use of its APIs but also create dialogs with its API customers. These community efforts are important to ensuring that a company's strategies actually serve user needs. Many businesses make the mistake of creating strategies from the inside-out, with internal assumptions leading the way, instead of from the outside-in, with user data and needs guiding development.

Additionally, enterprises should invest in establishing a presence where developers already gather—meetups, conferences, online channels, etc. API providers should have a strong presence on social media, both to interact with users and amplify success stories that users share, and should consider digital marketing tactics such as search advertising.

Whether they're sending representatives to a conference, sponsoring hackathons or posting to social media, enterprises should consider marketing tactics that may incentivize participation, from giving away swag to recognizing and promoting the work of top API users. Enterprises should also aspire to earn the respect of their customers via transparency whenever possible, whether sharing roadmap intentions and timelines or speaking candidly in the event of a breach about what caused the vulnerability and what can be done to fix it.

Many top businesses have created developer evangelist roles to take their community-building efforts to the next level. Evangelists can work both inside and outside the enterprise to champion adoption of APIs and provide a personal touch to supplement online feedback and marketing efforts. Top API evangelists are generally easy for developers to find on social media and often share their message during not only conference appearances but also webcasts, blog posts, and any other channels that offer an opportunity to interact with API users. Effective evangelists can even establish new business partnerships through their advocacy, giving them a prominent role in growing an enterprise's digital ecosystem.

SECTION SUMMARY

Spread the word

- Embrace a variety of channels. Be where the developers are.
- Considering incentivizing participation via swag or recognition.
- Task API evangelists with spreading the message. Be honest, transparent, and passionate about user success.

Never stop improving

The standard for a great developer experience is not static. What satisfies or empowers developers today may seem substandard tomorrow as new competing resources emerge and expectations among end users continue to evolve. API providers should regularly update their APIs, striving to continually close any gaps between what the API offers and what developers need in order to innovate.

An API provider must be able to offer support when developers are stuck. It must be able to maintain the quality of its services and remediate bottlenecks. To fuel improvements to its API products, the provider must collect not only hard data by actively monitoring API usage but also qualitative feedback from developers themselves. Developers are more likely to feel connected to an API program if they feel heard, and they are more likely to promote the APIs with which they feel most involved.

An API provider may also need to rethink the metrics to which it dedicates the most attention. The number of APIs produced, for example, is not a particularly meaningful metric because it provides little or no insight into how or by whom the APIs are being used. APIs that focus on how developers consume APIs—such as which developers are most active or which applications are logging the most API calls—open doors to better insights and better API iterations.

SECTION SUMMARY

Iterate. Then iterate again.

- Always look for ways to make APIs more useful to developers and to make the developer experience easier and rewarding.
- Create feedback loops. Provide resources to help developers when they get stuck.
- Invest in consumption-oriented metrics that provide insight into how and by whom APIs are being used.

Real-world use cases

How AccuWeather reached new audiences with its developer portal

AccuWeather, the world's largest weather data company, has for years been sharing its industry-leading weather APIs with global OEM partners such as Fitbit, Ford, LG, HTC, and Sony.

But it faced a challenge reaching individual developers, small businesses, and the weather-enthusiast community. This was an important audience for AccuWeather, said the company's senior technical account manager Mark Iannelli.

"A single developer always has the potential to be working on the next big thing and become our next big enterprise partner," Iannelli explained. "We needed a way to reach them."

> *"A single developer always has the potential to be working on the next big thing and become our next big enterprise partner, ..."*
>
> Mark Iannelli, Senior TAM at AccuWeather

By simplifying access to AccuWeather's unique functionality and APIs via a developer portal, the company has enabled a wide range of developers to build applications using the company's weather data. The self-service portal acts an an online store for AccuWeather's API packages and a hub for its external developer community.

This "democratization of APIs" and the improved developer experiences facilitated by the portal also open the door for new paying customers, Iannelli said. "As developers scale up successful API-based products, they have the possibility to purchase high-volume packages," he stated.

Within 10 months of launch, the portal had attracted more than 24,000 developers, issued over 11,000 API keys, and generated hundreds of paid package purchases.

But the benefits are even broader, Iannelli said. AccuWeather has also enjoyed not only indirect value from a growing developer ecosystem around its services but also exposure in new markets as developers leverage its APIs in new ways.

Developer experience checklist

Here are some key capabilities that businesses should consider to provide great developer experiences that will help grow their API programs:

- **Create a custom, branded developer portal to enable API users to access and understand how to use available APIs and write applications. Portal features should include:**
  - ☐ Self-service onboarding for new users
  - ☐ Role-based access control (RBAC) to maximize security
  - ☐ A registry of available APIs, including clear branding to help developers navigate resources
  - ☐ Documentation
  - ☐ Tutorials and case studies
  - ☐ Sample code and SDKs
  - ☐ Sandbox environments where developers can begin experimenting
- **Build a developer community through channels, including:**
  - ☐ Forums, blogs, customer success stories, and newsletters
  - ☐ Social media, SEO, and other online marketing
  - ☐ Hackathons, meetups, and conferences
  - ☐ A developer portal and APIs that are crawlable by search engines
- **Monitor and communicate significant updates or incidents to users, including:**
  - ☐ Versioning
  - ☐ Service status
  - ☐ Data breaches
- **Improve API products and support developers with tools and mechanisms, including:**
  - ☐ Analytics that provide insight into API usage
  - ☐ Feedback or bug reporting forums
  - ☐ A help center or FAQ

apigee

Google Cloud

1

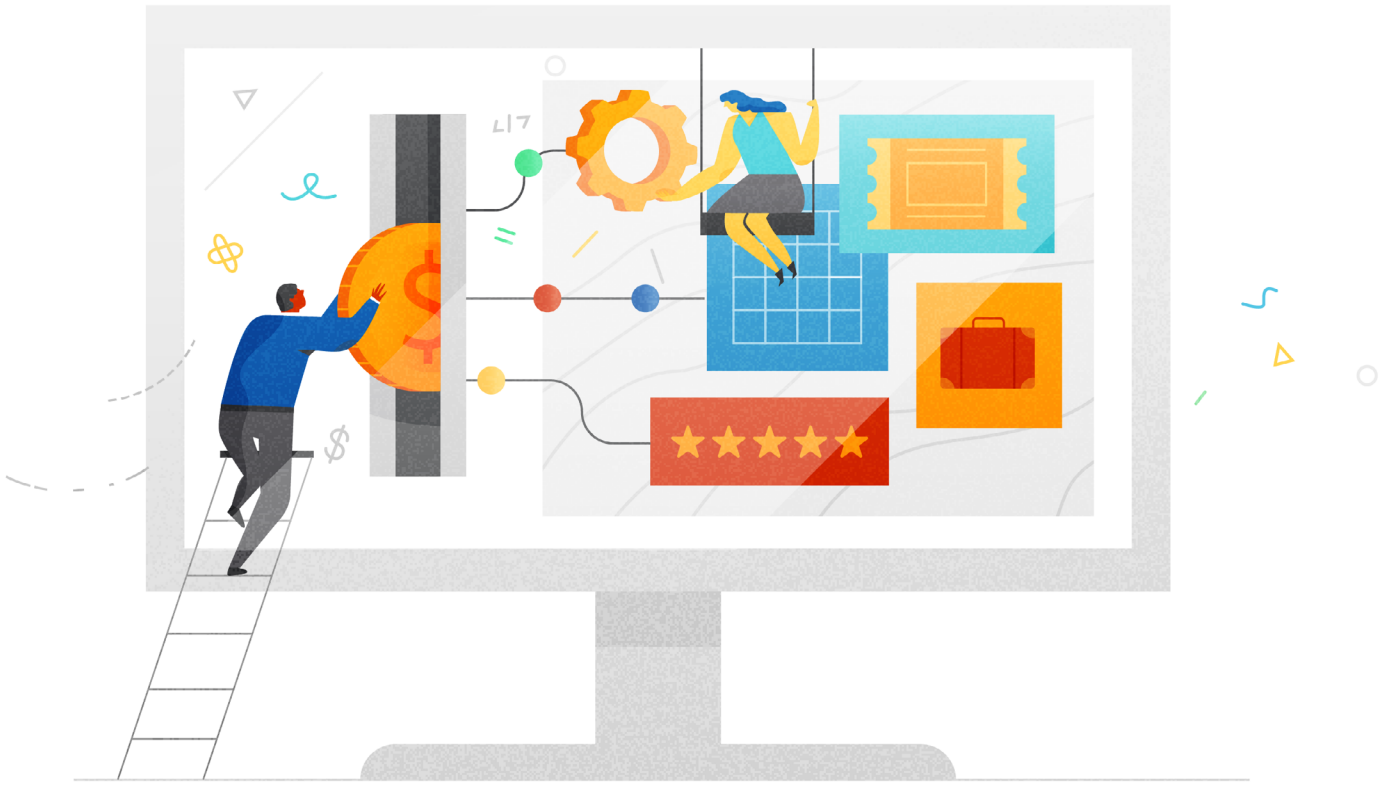
Inside the API Product Mindset

PART 2

Maximizing the Business Value of Digital Assets Through API Monetization

3

4



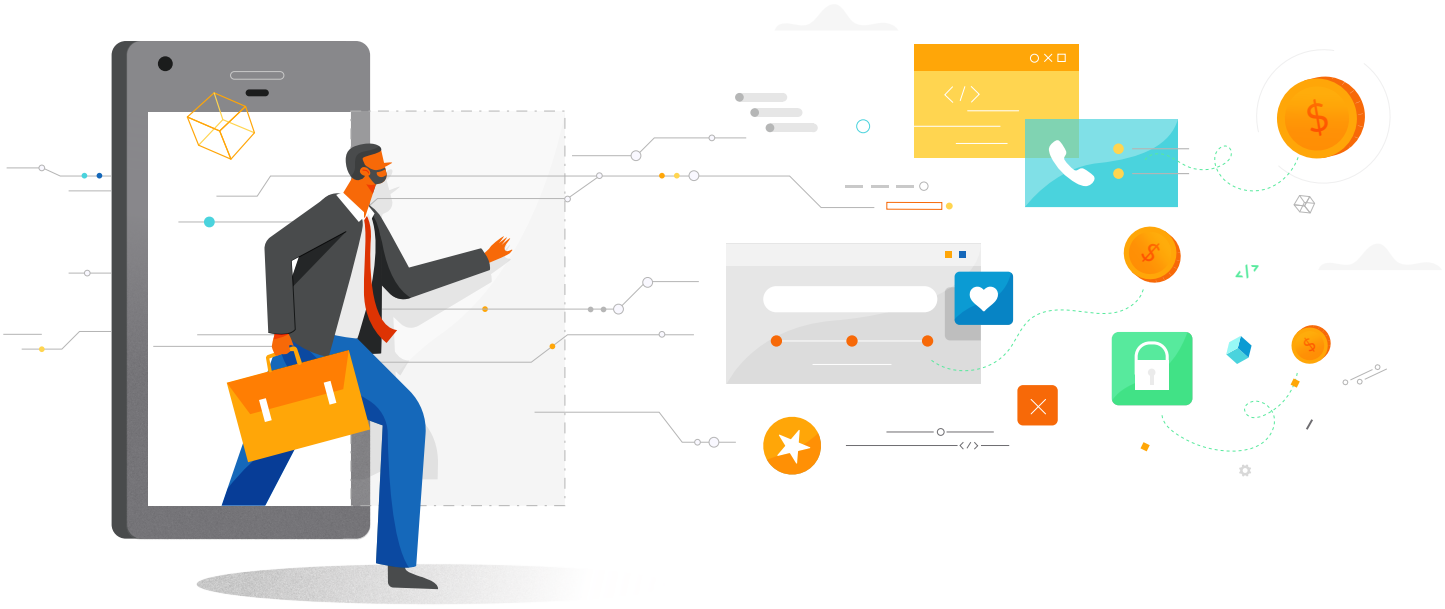
- Field-tested best practices
- Real-world use cases
- Monetization checklist

Table of contents

Inside the API product mindset

All businesses have digital assets: something valuable (functionality, data, access to certain product inventories or user bases, etc.) that can be expressed as software.

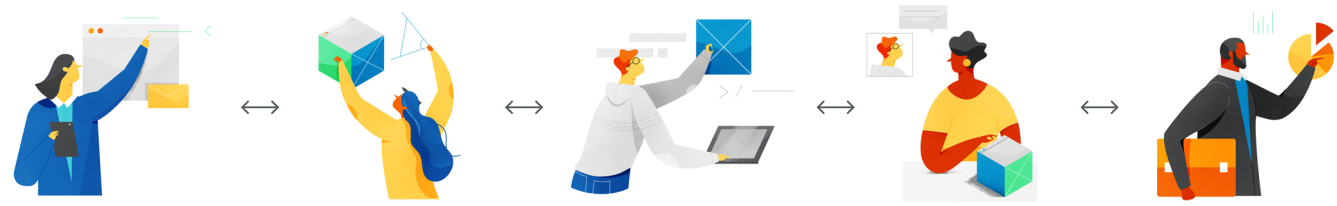
Application programming interfaces, or APIs, are the mechanisms that make these digital assets available to developers, both inside and outside an organization, who use them to build and add value to software applications.



An API abstracts a digital asset's complexity into a consistent interface, allowing developers to harness an asset even if they have no expertise in the underlying system that houses it. APIs allow developers to work more efficiently by reusing valuable functionality or data, and they allow assets to be combined more freely in new ways. The increase in developers leveraging these capabilities has not only led to an explosion of richer, more responsive digital experiences but has also positioned APIs in the middle of almost every organization's digital business initiatives.

As a result, many top digital enterprises view APIs not just as simple tools, but as full lifecycle products for developers.

As we explored in The API Product Mindset ebook, typical roles on an API product team include an **API Product Manager** who owns the processes and cross-functional coordination critical to the API's success; an **API Architect** responsible for designing and guiding the creation of APIs; an **API Developer** who builds APIs from the API Architect's designs and implements security policies and other protocols; an **API Evangelist** who serves as the voice of API consumers and owns partner and developer outreach; and an **API Champion** who works closely with internal executive sponsors to communicate the value of the API program to the rest of the organization.



API PRODUCT MANAGER | API ARCHITECT | API DEVELOPER | API EVANGELIST | API CHAMPION

The API product team typically carries several critical responsibilities:

- Design secure and easy-to-use APIs and bring them to market
- Deliver a world-class API developer experience
- Drive ongoing API improvements with monitoring and analytics
- Maximize the business value of APIs through API monetization, ecosystem participation, developer evangelism, etc.

This ebook dives deeper into one important aspect of this process: maximizing the business value of digital assets via API monetization.

Direct and Indirect Value: Not all APIs Should be Monetized

When a business makes its APIs available to external developers, those APIs can drive both direct and indirect value. Distinguishing these is a prerequisite to determining if, let alone how, an API should be monetized.

For example, many retailers make their store location APIs available for free to third-party developers. For both the developers who consume the API and the end users those developers serve, these APIs provide obvious value: better functionality, better applications, and better experiences. An API provider *could* charge for this value—but many decide they produce the most value and best customer experiences by bundling the capability or making it freely available to developers. After all, with more developers leveraging store location APIs, more applications will be delivering store locations to more users, which should theoretically lead to more sales. Is it wise to throw a wrench into this process by trying to monetize the API directly instead of enjoying the indirect benefits? A store locator has value—but perhaps not so much value or such unique value that many developers will pay for the chance to adopt the API.

In other cases, however, an enterprise may possess unique, hard-to-replicate and valuable data, such as the weather data AccuWeather's APIs provide. It may possess proprietary, valuable functionality such as the shipping and logistics capabilities offered by Pitney Bowes' APIs or the machine learning capabilities provided by APIs from a growing number of organizations. When an API provides access to valuable assets that developers are willing to pay for, directly monetizing the API may be more lucrative than giving it away to encourage adoption.

Regardless of whether or not an API is monetized, enterprises should think of developers as the API's customers. APIs should not merely expose digital assets—they should make the asset easy to consume and clearly present its value proposition. APIs that fail this baseline may struggle to gain adoption no matter if they carry a price tag, are given away for free to external audiences or introduced to in-house developers as internal resources.

Field-tested best practices

Many key API monetization best practices are encapsulated in "5 A's": Assets, Audience, Access, Adoption, and Administration and operations.

Assets: What business capabilities or data do you have that people want to use?

An API monetization strategy may start with identifying business capabilities or digital assets that have value outside the organization or even to other teams teams within the organization. Once valuable assets have been identified, they can be exposed as APIs, related assets can be grouped into API products, and related API products can be bundled into API packages, in order to target various developer needs.

SECTION SUMMARY

Define your valuable assets

- Identify valuable digital assets that could be useful to third parties or to other teams throughout the organization.
- Explore how those assets may be exposed as APIs, and bundled into products.

Audience: Who are the people who want to use your business capabilities?

To understand how to package digital assets into APIs that developers will use, an enterprise may need to identify constituencies of target customers and work to understand their needs. In some cases, the constituency is composed of communities of developers, and in some cases, the constituency is a set of business partners. Either way, an API provider should strategize from user needs rather than assumptions formed inside the organization. We call this approach working from the "outside-in."

Though an outside-in approach may suggest consideration of only an API provider's external customers or partners, that is not always the case, as developer communities exist within organizations as well. When thinking about API productization and monetization, APIs intended for internal developers should often be part of the same conversation as those intended for external developers.

Some internal APIs prove themselves so valuable that they become candidates for externalization and even monetization—but if those internal APIs are not managed like products, this opportunity may never be recognized. Moreover, monetization features applied to purely internal APIs can help businesses track finances within internal departments.

Enterprise leaders must also understand, as noted above, whether the API's target audience is more valuable under a direct-monetization model or under a model that seeks indirect benefits. Many people will use a resource, but only some within that group will be willing to pay to use that resource. Among those willing to pay, some will first need to first try an API for free, and some will need to be offered flexible billing that scales to their needs and uses. To find the right monetization mix, enterprises may need tools to tune monetization strategies on the fly and to pursue many models. Popular API monetization models include:

- **Freemium:** The API provider offers a basic level of service, defined in terms of quotas and/or capabilities, for free. Higher tiers of service that offer more capabilities or higher usage quota are offered for a fee.
- **Usage-based:** The API customer is charged for each transaction. Possible variations include
  - *flat rate models* in which the developer is charged a fixed rate for each transaction;
  - *volume-banded models* in which the developer is charged a variable rate depending on transaction volume;
  - usage-based models in which *custom attributes* dictate fees. For example, the developer may be charged for each transaction but the amount charged varies based on the number of bytes transmitted.
- **Revenue sharing:** The API provider shares with the developer a percentage of the revenue generated from each transaction. Possible variations include
  - *fixed share models* in which the developer receives a set percentage of revenue generated from each transaction;
  - *flexible share models* in which the developer receives a variable percentage of revenue generated, based on total revenue generated across API transactions over a specific period of time.

SECTION SUMMARY

Know your audience

- Identify your asset's target market, whether a community of developers or B2B customers, and how best to address that market's needs.
- View both internal and external developers as customers and manage both internal-facing and external-facing APIs as products.
- Remember: some APIs can be directly monetized, and many monetization models are possible—but many APIs are more powerful if they're given away to drive adoption and generate indirect value.

Access: How do people find and access your product?

Most products need a store—and in the case of API products aimed at the open market, that store often takes the form of either API marketplaces or a custom developer portal: a one-stop destination where customers can discover, explore, access, and test a provider's APIs.

Providers that build API portals should include a catalog that makes it easy for developers to discover APIs, and they should include documentation, sample code, blogs, and forums that clearly express the APIs' value and make it easy to start working with the APIs within minutes. Developers visiting the portal should be able to access API keys and otherwise onboard themselves via self-service processes, and the portal should include sandbox environments for developers to safely experiment.

SECTION SUMMARY

Make your APIs accessible

- Identify API marketplaces where target developers are already active, and consider investing in a developer portal to serve as a first-party storefront for APIs.
- Go beyond providing access to APIs. Provide sample code, documentation, testing tools, and other support materials to help developers make the most of your APIs and understand the value of your monetization packages.

Adoption: How do you market your product?

Various forms of marketing can also play an important role in driving adoption of APIs. In forming an outside-in viewpoint, enterprises should task evangelists with forming relationships with developer communities and create a presence at conferences and developer gatherings. A developer portal should include forums and other community features that allow users to exchange best practices and provide feedback to the API provider. SEO, SEM, social media, and other mainstream marketing efforts may be important as well.

SECTION SUMMARY

Spread the Word

- To make developers aware of your APIs, consider investing in marketing and appointing an API evangelist.

Administration and operations: How do you service and improve your product?

As with any product, a monetized API that is popular today may not be as lucrative in the future if it isn't updated and supported. API consumers are *especially* likely to have higher expectations for APIs they pay to access, so they will expect a high level of service, support, continuous improvement, and so on.

To meet these evolving expectations, it's important that API providers maintain uptime of their services and be able to change pricing models as developer behavior changes. It should almost go without saying that APIs providers must protect their products from threats, as developers will abandon an unsafe service.

API providers also need a strategy that encompasses both technical operations tasks, such as release management and API monitoring, as well as business operations tasks, including generating insights from analytics and using those insights to form strategies for growth. This means API providers need to continuously monitor their products so they can quickly detect and address service disruptions. Providers should use API analytics to observe how APIs are being used by different developer groups, learn how different monetization models and product features are affecting adoption, and shape iterations so that their APIs become more useful in the future. API providers should invest in tools that allow them to flexibly and responsibly modify pricing and A/B test different offerings.

SECTION SUMMARY

Keep delighting API customers

- Invest in testing and monitoring tools that help detect and resolve API-related issues.
- Adopt analytics tools that provide insight into how APIs are being used and how APIs can be improved.
- Prioritize the ability to flexibly tune API pricing models based on changes in user behavior.
- Protect customers by investing in proactive monitoring and security against bots and other threats.
- Make APIs more useful over time by iterating based on API usage trends, user feedback, and other data.

Real-world use cases

Telstra: Reaping broad benefits from API revenue

For Telstra, the largest telecommunications service provider in Australia, API monetization has played a key role in not only attracting new business, but also in raising the profile of the company's API program to key decision makers within the company.

When Telstra launched its first public API—a basic SMS messaging API—in 2015, its goal was both to experiment with ways to modernize its legacy systems and products, and to make those systems and products more modular and user-friendly for internal developers, said Steven Cooper, Telstra's API and platform evangelist.

"It was geared toward testing the waters and gauging what was needed internally for publicly available APIs and services," Cooper said.

> *"As soon as you start monetizing things, it really changes the way the business thinks about APIs"*
>
> David Freeman, Telstra

The response was impressive: within a few months, over 3,000 developers registered to use the API. With this initial success under its belt, Telstra began to focus on attracting new audiences. It launched its developer portal in 2017 as an online storefront for its APIs and built a roadmap for future API products.

As Telstra built up its ability to monetize its APIs and distribute them through its portal, new doors opened to new audiences. The opportunity to access and incorporate Telstra's APIs into new applications could be extended not only to enterprise partners but to individual developers and small teams—a dynamic that significantly expanded the range of developers innovating with Telstra's services.

Through its developer portal, developers sign up for limited free trials to experiment with Telstra APIs, including 1,000 free calls to Telstra's SMS and MMS APIs. There's also a rate package based on API usage for developers who need greater call volume.

"Telstra had some great traditional products out there such as SMS and MMS," said David Freeman, Telstra's general manager of API enablement. "What we've seen is that we can actually turn those into real, monetizable API products."

Monetization also provides a new and important metric for the Telstra API team: revenue.

"What are the quality metrics we really need to measure? We've taken a bit of a change on that now—we're looking at things like revenue because our program has matured," Freeman said.

Another very important achievement that Telstra's API team realized by implementing monetization: producing tangible revenue has gotten the attention of Telstra's business leaders and impressed upon them the value that comes from exposing telco services as APIs.

"As soon as you start monetizing things, it really changes the way the business thinks about APIs," Freeman said. "That helps us punch above our weight within the organization and helps us to have a voice."

Monetization checklist

Here are the key capabilities that businesses should consider when forming API monetization strategies and evaluating API monetization solutions:

- **Assets: Package valuable digital assets into monetizable products that can be easily consumed by developers**
  - ☐ Package a collection of related RESTful APIs as an API product
  - ☐ Create rules to define which API calls qualify as monetized transactions and to enforce monetization limits on API proxies
  - ☐ Issue PCI DSS compliance certificate, specify terms and conditions and support multiple currencies
  - ☐ Dig into API traffic analytics to determine which APIs are most popular and which APIs might be ripe for monetization
- **Audience: Manage your customers' experiences and options**
  - ☐ Support for various monetization models: revenue sharing, usage-based billing, subscriptions, freemium, etc.
  - ☐ Create and manage rate plans
  - ☐ Manage renewals and notify customers of updates with notification templates, support webhooks, etc.
  - ☐ Measure developer engagement and funnel analytics to top developers and apps that use your APIs
- **Access: Help your customers find and use your API products**
  - ☐ Create an appealing and intuitive self-service developer portal
  - ☐ Publish API packages and rate plans, and integrate with payment providers
- **Adoption: Help your customers hear and learn about your API products**
  - ☐ Facilitate community among the developers using your APIs and create feedback loops with forums, blogs, and other outreach channels
- **Administration and operations: Track metrics and derive insights that help you improve your API products**
  - ☐ View monetization-specific reports for billing, prepaid balance, variance, and revenue use cases
  - ☐ Set up alerts to notify administrators of API issues
  - ☐ Investigate API issues to identify the root cause of problems, regardless of whether the use case involves a developer app, proxy or backend target

About Apigee API Management

The Apigee API management platform delivers full lifecycle API management to help businesses unlock the value of data and securely deliver modern applications. Apigee offers a rich set of capabilities to enable enterprises to monetize their APIs, including tools to test and flexibly tune pricing models, robust monitoring tools, and API monetization.

Now that you've finished reading, why stop learning?
Visit the Apigee website for more.

apigee

Google Cloud

1

2

Inside the API Product Mindset

PART 3

Building and Managing Secure APIs

4

- Field-tested best practices
- Real-world use cases
- API security checklist

Table of contents

Inside the API product mindset

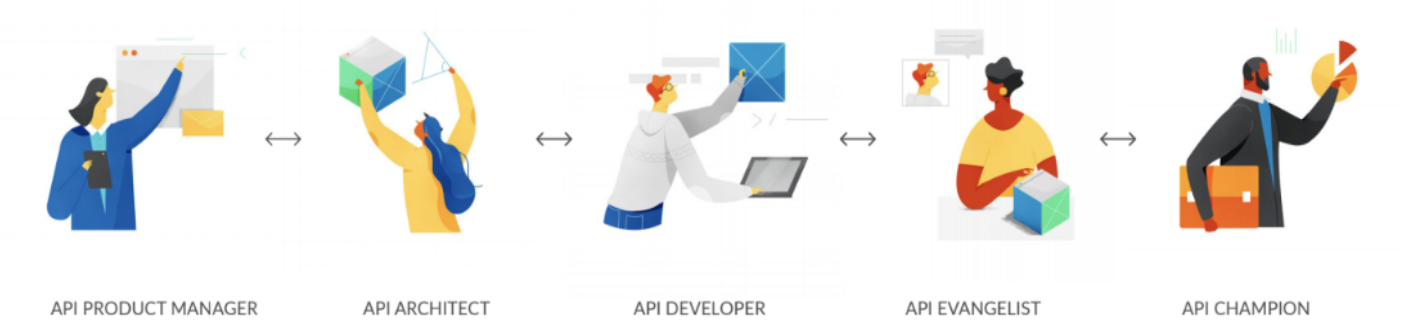
APIs (or application programming interfaces) are the de facto standard for building and connecting modern applications. With APIs, a business can securely share its data and services with developers, both inside and outside the enterprise, to foster new operational efficiencies, unlock new business models, and enable business transformation.

APIs are often characterized as products for developers who build the connected experiences that power the digital economy. All businesses have valuable digital assets—functionality, data, etc.—but many of the systems that contain this value were never meant to easily connect. APIs abstract this complexity into an interface that enables developers to leverage systems in which they have no expertise, and to combine digital assets in new ways.

In this way, APIs are not only expressions of a business's capabilities and points of differentiation but also the mechanisms that make those capabilities and differentiation leverageable for strategic purposes.

With this in mind, many successful organizations manage APIs like products, with full lifecycles, long-term roadmaps, a customer-centric approach, and constant iteration to meet business needs. In our experience in Google Cloud's Apigee team, organizations that treat APIs as products—as opposed to one-off *technology projects*—are more likely to realize the potential value of APIs as business accelerators.

As we explored in The API Product Mindset ebook, typical roles on an API product team include an **API Product Manager** who owns the processes and cross-functional coordination critical to the API's success; an **API Architect** responsible for designing and guiding the creation of APIs; an **API Developer** who builds APIs from the API Architect's designs and implements security policies and other protocols; an **API Evangelist** who serves as the voice of API consumers and owns partner and developer outreach; and an **API Champion** who works closely with internal executive sponsors to communicate the value of the API program to the rest of the organization.



The API product team typically carries several critical responsibilities:

- Design secure and easy-to-use APIs and bring them to market
- Deliver a world-class API developer experience
- Drive ongoing API improvements with monitoring and analytics
- Maximize the business value of APIs through API monetization, ecosystem participation, developer evangelism, etc.

This ebook dives deeper into one important aspect of this process: designing secure, easy-to-use APIs.

Balancing Protection and Ease of Use

APIs expose data or functionality for use by applications and developers, which means they are the doors and windows that allow access to a business's valuable digital assets—and thus to the heart of the business itself. Like all doors and windows that provide access to something valuable, APIs should be designed with security top of mind.

API developers generally understand the importance of adhering to API design principles because no one wants to design or implement a bad API. Even so, it may be tempting to look for shortcuts to meet aggressive sprint timelines, get to the finish line, and deploy an API. Though understandable, these shortcuts may pose a serious risk: unprotected APIs.

Vulnerable APIs can expose a business's core data and services to a variety of malicious attacks. Many large enterprises and organizations have suffered breaches, the consequences of which can range from embarrassing to catastrophic, as a result of holes in API defenses.

These holes are not always due to negligence. Developing secure APIs isn't as simple as it might seem, as the API provider must often strike a balance between ease of use and security.

One of the main goals of an API is ease of use; an API is of little value if developers aren't consuming it, and no one wants to work with an API that is so locked-down with security mechanisms that they get in the way of productivity. The challenge in API security isn't locking down the API; rather, it's managing APIs that are secure yet still flexible and easily-accessible enough to foster innovation.

Striking this balance means API providers should generally avoid the complex systems dependencies and heavy-handed governance models that typified previous generations of IT strategy, when connected digital markets hadn't yet risen to prominence and the corporate firewall was seen as a fortress to repel outsiders.

Indeed, legacy security practices in general may no longer be viable because they rely on limiting the rate at which enterprises update their software. Enterprises that persist in a slow, cumbersome update process run the risk of being unable to react to new vulnerabilities. For most organizations, the best path is to adopt technology strategies, such as API-first development, that support constant updates and provide a plane for implementing security controls.

As part of the API product mindset, enterprises should always consider the following API security best practices: data encryption, end user and application authorization, rate limiting, and bot detection.

Field-tested best practices

Don't mess with TLS

"Transport layer security," or TLS, is the foundation of API security. Protecting network traffic using an encrypted channel is the easiest way to ensure that sensitive information is not susceptible to attacks while in transit. Encrypting traffic over the network should be seen as an ironclad requirement; no API should go without it.

TLS also helps enterprises ensure that the client, such as a mobile app, is communicating with the correct server. For example, a free WiFi network in a coffee shop might be attached to a fake DNS (domain name system) server set up to route banking transactions to another site. Without TLS, a mobile app using this WiFi network could be open to attack. It's worth noting that though this feature of TLS is activated by default, a number of client libraries and tools make it easy to turn off or bypass this feature—don't do it.

API developers must also be cognizant of using TLS properly, as there are many different options. What cipher suite to use? What encryption algorithm to apply? These things change all the time—and it's important that companies stay on top of TLS changes, update their configurations by following the advice of internal security teams and external experts, and expect more changes in the future. If a team hard-codes particular TLS versions and cipher suites into servers, for example, they may make future updates more difficult. Many API teams test TLS configurations with services such as the SSL Server Test from Qualisys SSL Labs.

Beyond TLS, API developers must vigilantly keep up with the security world and should constantly assess and consider measures that go beyond basic encryption. For example, API providers might consider employing trace tools for debugging issues, data masking for trace/logging, and tokenization for PCI (payment card industry) and PII (personally identifiable information) data.

SECTION SUMMARY

TLS is the foundation

- Never turn off TLS.
- Keep up with TLS changes.
- Consider going beyond encryption with trace tools, data masking, and tokenization.

Ensure strong authentication for both end-users and applications

API teams should build APIs that provide authentication for both end users and applications.

OAuth is the de facto open standard for API security, enabling token-based authentication and authorization on the Internet. It provides a way for end users and applications to gain limited access to a protected resource without the need for the user to divulge their login credentials to the app. For APIs, it allows a client that makes an API call to exchange some credentials for a token, and that token gives the client access to the API.

Unlike a password, a token uniquely identifies a single application on a single device. A key best practice for API teams is to build authentication of all end users into critical applications, as it's the only way to keep security credentials outside the app. API teams building API products should familiarize themselves with the full capabilities of OAuth and current authentication best practices.

Security-minded API teams will additionally recognize that OAuth is not just about authenticating end users—authenticating applications is also a fundamental part of API security. For example, by authenticating applications, a provider can stop runaway applications that continue eating up resources when they should have stopped running. Some API teams have neglected application authentication, but fortunately, OAuth now natively includes this concept; in order to build an application that gets an OAuth token, a developer must supply not only user credentials but also application credentials.

Application authentication does not provide complete security against attacks, as anyone with application credentials can access and potentially abuse APIs. But it provides an important extra layer of defense. Developers should consider using different credentials for each version of an application to make it easier to pull a bad version.

Though OAuth is an incredibly useful security standard, it's made up of a complex family of specs, and there are numerous ways to use it. To make leveraging OAuth simpler, many API teams rely on API management platforms to generate OAuth tokens and apply granular control over what a token is allowed to do.

API platforms can also help enterprises improve security by managing access within the API team to sensitive resources, such as the developer portal. Features such as role-based access control (RBAC) help API teams to manage access according to roles that define user privileges. Such roles might include "organization administratrator" with full access to resources; "read-only organization administrator" with read-only access; and "business user" with access to tools to create and manage API products but read-only access to other resources.

SECTION SUMMARY

Don't neglect authentication

- Use OAuth to authenticate users.
- Authenticate both end users and applications.
- Consider RBAC to manage access.

Rely on rate limiting

API teams should always consider using rate limits for additional API security, as any API could be subject to a brute force attack. In a brute force attack, automated software is used to generate a large number of consecutive guesses as to the value of required data, such as a login password. If there is no rate limit, these attacks can continue indefinitely, with bad actors deploying a distributed password-cracking API that keeps running until it manages to infiltrate a system.

In the face of such threats, it is critical to apply basic rate limits to APIs. For example, an API team might establish a limit that forbids an application from calling an API more than 500 times per second or a certain number per day. To avoid against performance lags, downtime, and other backend degradation, it is also a good practice to enforce a spike arrest, which throttles the number of requests that can be made to an API during a traffic surge, or per-app quotas. As with OAuth usage, many organizations use API management platforms to help them apply rate limiting and spike arrest capabilities across their APIs.

SECTION SUMMARY

Keep brute force attacks at bay and manage traffic

- Use rate limits to protect against brute force attacks.
- Apply spike arrests to avoid performance lags or downtime during a traffic surge.

Beware of bad bots

As business-critical functions have shifted to connected devices, the automated connecting of software and systems has become an indispensable part of how business gets done. Unfortunately, automation has also enabled new forms of cybercrime—namely, bot attacks, in which bad actors deploy automated software programs over the Internet for malicious purposes, such as identity theft.

Many bots are useful. In fact, millions of bots play critical roles in enabling the API-powered connected experiences driving the digital economy. The key for enterprises is to enable beneficial automation without also enabling harmful bots.

Bad bots might arise when a hacker acquires a compromised API key, perhaps from a partner or mobile app, and then reverse-engineer how the app works in order to emulate necessary API call flows. From there, the hacker can run hundreds or thousands of bots at scale, producing scores of ostensible "users" who appear to be doing normal things, such as purchasing products or accessing loyalty accounts. Because the bots are actually working in concert, however, they can end up manipulating the prices of goods on auction sites, impacting how best-selling products appear on search or recommendation engines, or awarding a hacker millions of loyalty points they did not earn—just to name a few examples.

These risks point to the obvious importance of properly managing API keys—but that's only a start. API teams must also monitor not only API access, but how traffic behaves. They should look at the behavior of all the users coming in to identify who they are, where they come from, and most importantly, what they do, for example.

There are numerous approaches to stopping malicious behaviors, but technologies and approaches that work for the network or the web do not necessarily work for APIs. To thwart bot attacks on APIs, API teams' tactics should include sophisticated machine learning-based solutions that can analyze API request traffic, identify patterns that might represent unwanted requests, and learn as hackers rotate their attempts across a large set of bots.

SECTION SUMMARY

Use machine learning to put bad bots in their place

- Monitor not only API access but also traffic patterns in order to spot suspicious behaviors.
- Apply sophisticated algorithms and machine learning to spot bad bots, and note that approaches that discern network or web attacks may not be effective for APIs.

Real-world use cases

Urban Science: Protecting a Rapidly Expanding API Program

Urban Science serves customers in the automotive industry by making business recommendations and proposing solutions based on scientifically validated results. The company provides decision support services to most global automotive OEMs and retailers and also serves the healthcare and retail markets.

Building on its history of providing market intelligence to the automotive industry, Urban Science has been offering its Marketing Intelligence Cloud solution for several years to U.S. automotive marketers and their agencies to enable them to make more data-based decisions to improve the effectiveness and efficiency of their marketing efforts. The platform relies heavily on APIs created and managed via the Apigee API management platform to keep data flowing quickly, securely, and efficiently.

> *"Having real-time data at our fingertips, combined with our ability to take action quickly, is instrumental in keeping our product always-on."*
>
> Luke Mercier, Urban Science

Urban Science uses Apigee security features to help ensure that the 11.5 million monthly transactions that run through the Marketing Intelligence Cloud remain protected—and will remain so even with the platform's exponential growth.

Apigee offers an API management proxy frontend to the platform and is used to help secure all endpoints. There's no access point to the Urban Science Marketing Intelligence Cloud that doesn't go through the Apigee gateway and its security features.

With 650 published APIs available to third parties as well as to its internal development teams, this protection is particularly important. But API management security features do more than protect sensitive information.

Apigee dashboards run continuously on several screens, where the Urban Science API team monitors traffic down to the minute. This data is used to observe the Marketing Intelligence Cloud platform's growth, adoption rates by product, and adoption rates by customers.

Additionally, the data is used to help scale the platform and plan for more resources, CPU memory, and storage. At the same time, team members use alerting and notification features to enable them to take action on potential problems in real time, often resolving possible outages before they can affect customers.

"We've prevented several outages with Apigee alerts," says Luke Mercier, Urban Science's global system manager. "We've taken action to stop an issue from becoming a systemic failure. Having real-time data at our fingertips, combined with our ability to take action quickly, is instrumental in keeping our product always-on."

API security checklist

API security should be of paramount importance to any enterprise that is exposing digital assets. Here are some key aspects of security that business leaders and API teams should look for when evaluating API management platform providers:

- **Authentication and Authorization:**
  - ☐ Support authentication of users and applications with TLS, SAML, OAuth 2, two-factor authentication, API keys, and mechanisms to block or limit known bad actors or people who abuse terms of service
  - ☐ Manage user identity and RBAC by integrating with LDAP and active directory
- **Threat Protection:**
  - ☐ Protect against malicious activities such as XML poisoning, JSON and SQL injection, and DoS attacks and DDoS attacks
  - ☐ Detect and prevent bot attacks in real-time using machine learning techniques
  - ☐ Provide quota and spike arrest and IP blocking capabilities to act against API abuse
- **Privacy and Compliance:**
  - ☐ Provide the ability to log all actions, and the ability to audit logs
  - ☐ Compliance with SOC 2 (Service Organization Control), PCI DSS (Payment Card Industry Data Security Standard), and HIPAA (Health Insurance Portability and Accountability Act)
- **Scale and Compute:**
  - ☐ Handle a massive number of API keys and tokens and compile policies in runtime

About Apigee API Management

The Apigee API management platform delivers full lifecycle API management to help businesses unlock the value of data and securely deliver modern applications. Apigee offers a rich set of security capabilities to help enterprises protect their digital assets. To learn more about how Apigee's authentication, monitoring, rate limiting, and bot protection capabilities help API product teams throughout the world, visit secure APIs.

Now that you've finished reading, why stop learning?
Visit the Apigee website for more.

apigee

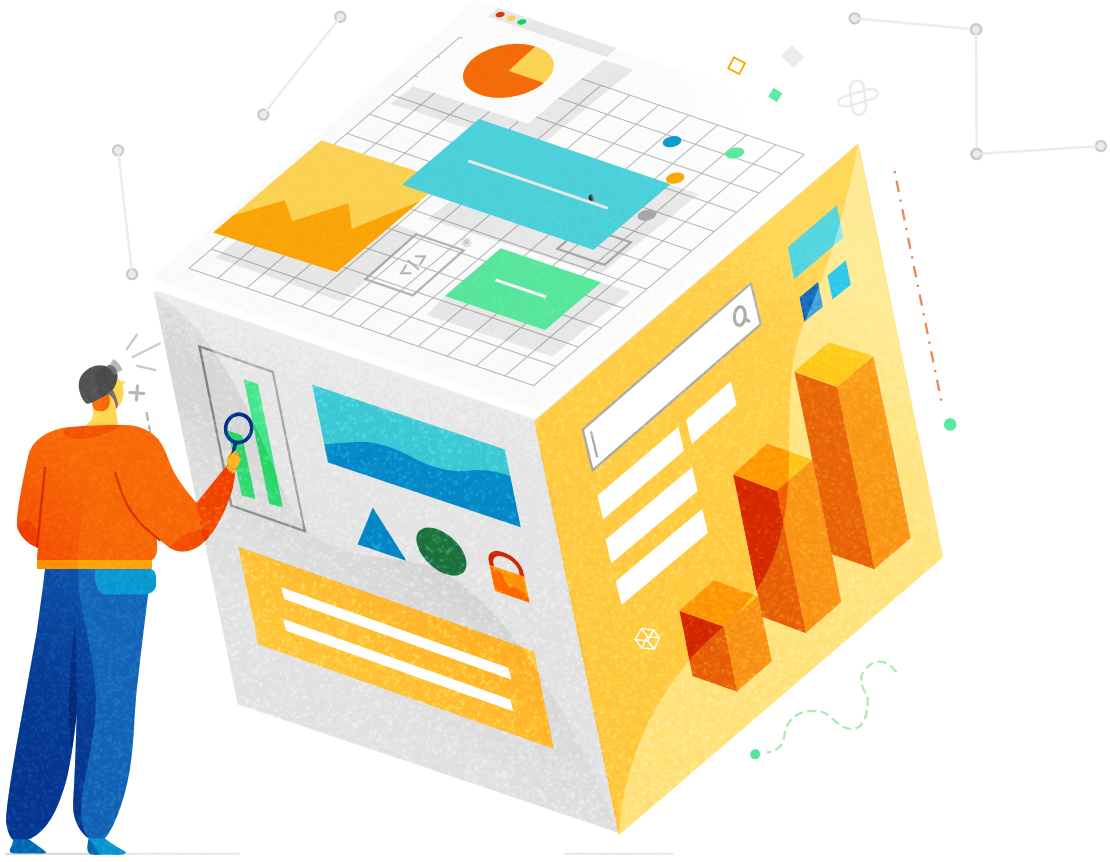
Google Cloud

1
2
3

Inside the API Product Mindset

PART 4

Optimizing API Programs with Monitoring and Analytics



- Field-tested best practices
- Real-world use cases
- API monitoring and analytics checklist

Table of contents

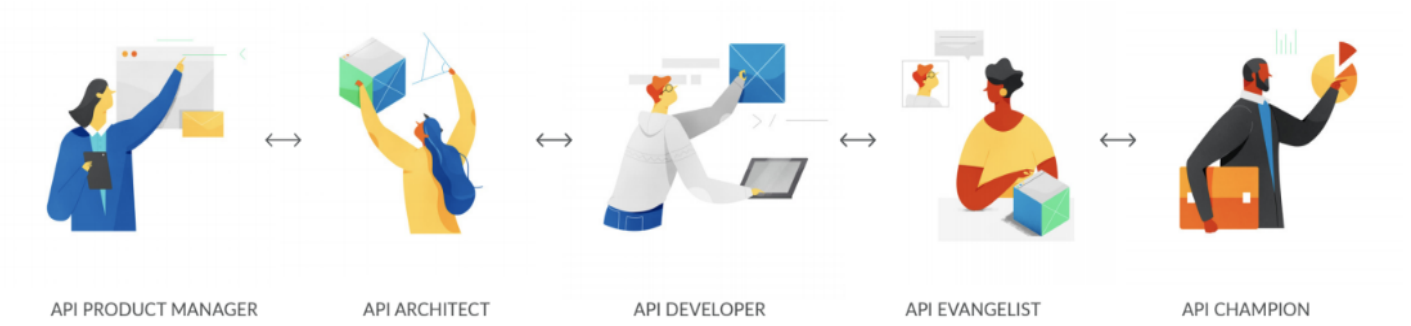
Inside the API product mindset

APIs (or application programming interfaces) are the de facto standard for building and connecting modern applications. With APIs, a business can securely share its data and services with developers—both inside and outside the enterprise—to foster new operational efficiencies, unlock new business models, and enable business transformation.

APIs are often characterized as products for developers who build the connected experiences that power the digital economy. All businesses have valuable digital assets—functionality, data, etc.—but many of the systems that contain this value were not designed to easily connect. APIs abstract the complexity of making these connections into an interface that allows developers to easily leverage digital assets and combine them in new ways for new services.

In this way, APIs are not only expressions of a business's capabilities and points of differentiation but also the mechanisms that make those capabilities and differentiation leverageable for strategic purposes.

With this in mind, many successful organizations manage APIs like products, with full lifecycles, long-term roadmaps, a customer-centric approach, and constant iteration to meet business needs. In our experience in Google Cloud's Apigee team, organizations that treat APIs as products—as opposed to one-off *technology projects*—are more likely to realize the potential value of APIs as business accelerators.



As we explored in our recent ebook *The API Product Mindset*, an API product team carries several critical responsibilities:

- Design easy-to-use and secure APIs
- Deliver a world-class API developer experience
- Drive ongoing API improvements with monitoring and analytics
- Maximize the business value of APIs

This ebook dives deeper into a particular aspect of an API program: how to use API monitoring and analytics to optimize your API programs and digital strategies.

Understanding API Monitoring and Analytics Use Cases: *Right Now* vs. *Trending*

Given that APIs are products, it should go without saying that monitoring and analytics are an integral part of the API product lifecycle. Would an API be a useful product if its provider were not proactively watching how it performs in order to help ensure SLAs are met and increase the odds that developers are happy? Of course not. Are users likely to remain happy if the provider does not apply analytics to improve the product and intelligently plan investments and roadmaps? Same answer: no.

Because both monitoring and analytics involve decisions based on data derived from API traffic, the two topics may blend together from certain angles—but in many ways, they are distinct. Some users of an API management platform may access only a finite number of monitoring tools without touching analytics dashboards and vice versa.

Broadly, one can think of monitoring as watching continually updated streams of data that informs real-time action to maintain system health. It is focused on *right now*—the equivalent of a heart rate monitor that lets the user know "your heart is fine" or "seek help—you're having a heart attack."

One can think of analytics, in contrast, as the analysis applied to data to create insights, sometimes in near real-time, but more typically in retrospect. Analytics are less focused on *right now* than on what is *trending*; instead of a heart monitor alerting the user to a heart attack, it's the physician making lifestyle and exercise recommendations based on a patient's activity data, dietary habits, and lab results. One extreme is focused on keeping API performance up to standard in the present tense, the other is focused on learning over time how to improve APIs, how they are being used, and how they might be leveraged for new business opportunities.

Enterprises should consider this full range of API monitoring and analytics use cases—from in-the-moment action to retrospective insight—when assessing API management solutions. It's important to see this as a spectrum, with the data collected during monitoring fueling the insights derived via analytics. It's also important that the monitoring and analytics be integrated into the core management system, rather than added via an outside service, so that monitoring and analytics can more directly lead to management action.

Operations teams, for example, need a robust system for monitoring large volumes of traffic and detecting and fixing API issues in minutes—ideally before they impact customers. They need to be not only alerted when something goes wrong, but also enabled to diagnose quickly and correct any problems. Maintaining the health of APIs is important at all times, as service degradation can lead to customer churn, revenue loss, and reputational damage—and it is particularly important during seasonal traffic spikes. Failure to handle peak traffic during Black Friday or Cyber Monday could literally cost a business millions of dollars.

But operators are only one type of user, and monitoring system health in near-real time is only one use case that an API monitoring and analytics solution should support. Indeed, some use cases do not divide monitoring and analytics discretely. For example, suppose an API features an SLA of 2 seconds, and that the business has set up alerts to trigger whenever latency slips to at least 1.5 seconds. When diagnosing the problem, the operations team might find something immediate, such as a spike in requests, but it might also find something more systemic, such as latency that's slowly been inching up over the last month. In the latter case, the situation rapidly shifts from a monitoring use case to an analytics one that may involve more team members, and the ability for the management solution as a whole to support this shift can be critical.

Possible use cases along this monitoring-analytics spectrum are numerous. API product managers will want to understand how APIs are being adopted and consumed, which specific APIs are popular with which developers, how APIs can be improved, and so on. Developers who leverage APIs will want to know how their apps are performing. Business stakeholders will want analyses of whether API investments are paying off and how future investments can best be allocated. Security professionals may require monitoring and analytics tools to effectively combat threats.

Enterprises that neglect this multiplicity of users and use cases for API monitoring and analytics may find themselves unable to easily recover in the event of a major outage, blindsided by changes in user behavior, susceptible to bad actors, and ill-equipped to measure an API program's progress and plan future investments.

Field-tested best practices

Know the needs of different users

API monitoring and analytics are often mentioned in the same breath and may be aspects of the same API management solution—but the term refers to a range of tools and use cases that need to fulfill the requirements of a variety of users. Key user groups include:

- The API team: The API team includes API developers who need visibility into the step-by-step behavior of APIs in order to diagnose latency problems and otherwise improve performance. The team should also include a product manager responsible for the success of the API program who needs analytics related to API adoption and usage, often sliced across dimensions such as channels, developers, or locations. It may also include an API evangelist responsible for communicating with and relaying important information to developer communities, including insights into how developers are leveraging APIs.
- Developers: Analytics can help app developers know how their apps are doing and may provide insight into how apps can be improved—which means that by sharing insights and metrics with developers (e.g., success rate, response times, and response codes), enterprises can increase the likelihood that developers produce quality apps.
- Operations administrators: Operations teams need to understand API patterns and anticipate when to add backend resources or make other critical adjustments. They are responsible for maintaining peak performance and API availability, which means they need to monitor throughput, latency, and errors in near real-time, be alerted when problems arise, and harness tools to combat bots and other security threats.
- Business stakeholders: Business stakeholders need to understand how API investments are impacting and driving digital business strategies, and they need to use insights gleaned from analytics to inform continuous strategy iteration and where to invest API dollars in the future.

SECTION SUMMARY

Different users, different needs

- Keep in mind the spectrum of analytics and monitoring use cases when assessing solutions.
- Remember: Monitoring use cases and solutions involve near-real time visibility into API traffic, and analytics use cases and solutions involve insights into API usage and improvements over time. Though some tasks involve aspects of both monitoring and analytics, these terms are not interchangeable and may mean different things to different users.

Invest in proactive monitoring and analytics tools integrated into the core API management solution

It is important that API monitoring and analytics be part of the core API management system, not a black-box, bolt-on solution. Enterprises should consider white-box, contextual API monitoring and analytics tools that are part of the main management solution. When monitoring and analytics tools are integrated directly, rather than bolted on, the platform managing APIs is the same platform capturing data—which means the data can be acted on more easily. Not only do alerts and the tools that mediate them exist within the same interface, but many tasks can be more easily automated (such as automatically throttling traffic to a struggling backend) with this kind of integration.

Traditional synthetic or black-box monitoring tools that involve periodically probing the system are generally limited to reporting API availability data. These tools, which are separate from and run atop the management platform, run live checks on a predefined schedule, calling the API to make sure it is available, but they do not offer visibility into performance metrics. This means that synthetic monitoring may alert an operator to a problem, but it is unlikely to help them understand the nature of the problem. With such tools, operations teams need to manually investigate multiple systems and correlate debug sessions in order to diagnose API issues.

By contrast, an API-specific management solution that provides near real-time monitoring and enables ops staffers to quickly diagnose and resolve problems can enable teams to more efficiently and effectively keep abreast of the essential aspects of their API-powered digital business.

API monitoring should provide more than just availability metrics and alerts; it should detect anomalies and empower users to dive deeper and find the root cause of issues. Monitoring dashboards should provide at-a-glance visibility into hotspots, latency, and error rates while enabling users to drill down to find policies where faults occur, target problems, and address other specific elements that require remediation. A default view might provide quick looks at the APIs with the most traffic, the highest error rates, or the most latency, for instance. Other views should enable users to dive deeper, such as a timeline view that displays historical trends for a given API.

SECTION SUMMARY

Real-time problems demand real-time monitoring and insights

- Invest in real-time, API-specific white-box monitoring. Black box tools that are bolted on to the main management platform typically offer only limited options for automation and periodic system probes that can leave a business in the dark.
- Identify and resolve issues by drilling down from high-level metrics into the specific policies, targets, and code causing faults.

Define the right alerts and use historical data to set up and tweak alert thresholds

Alerts are obviously an important monitoring feature—but they have to be configured correctly to trigger the intended results. If they fire too often in situations that don't actually require someone's attention or action, for example, members of the operations team may begin to ignore them.

When developing APIs, enterprises should perform tests to set baselines for latency and establish initial values for alerts. These values should be refined over time based on analytics and the enterprise's "signal-to-noise" ratio (the proportion of alerts that require investigation and remediation). This is an interesting case in which analytics insights flow back into monitoring, rather than the more typical case where monitoring data becomes the foundation for analytics. Alerts should always be configured to surface before a customer SLA is violated.

Enterprises should also consider including a short description of recommended actions in alerts, which can help a first responder get up to speed more quickly. Many businesses also connect alerts to their service management system to ensure that when an alert is triggered, it is tracked for follow up.

Full-featured API management solutions may also include "collections" features to reduce the number of alerts an enterprise needs to implement and manage. Rather than allowing alerts to fire for each API and potentially causing chaos, an operator can create alerts of a given collection of proxies, targets, and developer apps. To be useful, collections should include items with common characteristics, such as APIs with similar latency requirements or error rates, or developer apps assigned the same level of business criticality.

SECTION SUMMARY

Use alerts—but don't overuse them

- Configure alerts to notify operations administrators in the event of a problem, and to help ensure that SLAs are met.
- Leverage analytics to derive insights, tweak thresholds, and avoid sending too many alerts; if operators are alerted when there is no emergency, they are likely to begin ignoring notifications.
- Use "collections" to group alerts with common characteristics, and to limit the number of alerts and keep them relevant.

Focus on consumption metrics

Production-oriented measures, such as the number of APIs produced, provide little if any insight into the performance of an API program. Metrics that describe how APIs are being used are more likely to be valuable. Is API traffic trending up over time? Who are the top developers? When is API response time fastest or slowest and which geographies generate the most traffic? On which devices do end users run the apps that leverage the APIs? Which APIs are generating the most revenue or driving some other business KPI? Businesses that can answer these questions are well-positioned to understand their API users, communicate progress to internal stakeholders, align around problems and goals, and ultimately grow their API programs.

Not all API management products are equipped to produce this level of detail. It's one thing for a solution to generate basic traffic metrics about an API—but it's another thing for it to generate metrics about developers and apps that call this API. It is important that any API monitoring and analytics solution be able to produce this more descriptive data, typically by allowing the enterprise to register app and developers and apply policies to proxies.

SECTION SUMMARY

Learn from how APIs are used

- Focus on metrics that describe how APIs are being used, such as which APIs are driving the most traffic, trending up or down with developers over time, etc.
- Connect API analytics to KPIs to align teams around internal goals and communicate the API program's progress to business stakeholders.

Test assumptions and iterate based on insights

API developers apply policies to ensure robust app functionality while protecting backend systems—and API teams must ensure that once these policies are implemented, they function as expected.

If an API producer implements the wrong policy, for example, other developers might not adopt the API. Suppose an API developer applies an OAuth policy to a product catalog API. This policy would force end users to authenticate before getting generic information about a company's products on a mobile app.

Such friction can be a blocker to adoption—which is why it is important to analyze API patterns across a wide population of customers and iterate APIs based on insights. Moreover, just as analytics might reveal blockers, they might show that some APIs are unexpectedly popular or being used in unexpected ways, perhaps even to the extent that they could be monetized or shaped into new lines of business. They might show that an API is being used in an unexpected geography, leading the enterprise to reassess how it promotes APIs to that market and how its business operates in the market as a whole.

Analytics are where assumptions and intentions meet user behavior—and where new, smarter iterations are born.

SECTION SUMMARY

Always look for ways to make APIs more useful

- Avoid assuming that users will react to API design and policy decisions as intended; always be prepared to make adjustments based on how APIs are consumed.
- Leverage API analytics to understand and eliminate sources of user friction.

Real-world use cases

Driving the Business While Maintaining Visibility and Control

Traditional monitoring tools are limited to reporting availability information by making synthetic calls to live APIs, limiting visibility into API traffic and, because the tools are add-ons to the main management platform, limiting control when problems arise and need to be addressed. APIs are literally expressions of an enterprise's ability to do business—if they go down, so does the opportunity to do business. As Apigee customers attest, these limitations are simply not acceptable for the mission-critical use cases that many organizations demand.

In a case study, AccuWeather senior technical account manager Mark Iannelli explained how Google Cloud's full lifecycle API management platform is helping the company to support and delight developers. "With Apigee Edge, we're able to keep close tabs on who's signing up, what sort of traffic they are producing, from where—and even observe patterns in traffic activity," he said. These proactive monitoring and analytics capabilities built into the core management platform enable AccuWeather not only to ensure developers are enjoying a good experience working with the APIs, but also to identify developers who might be candidates for other API packages or options.

amadeus

> *"Knowing the number of transactions, response times on APIs, or the page travelers are spending the most time on could be invaluable for us to make informed decisions."*
>
> Xavier Gardien, Amadeus

"Understanding how our APIs are consumed is also key for us and our customers. With Apigee we are able to see this and provide them with a detailed view of API analytics," said Olivier Richaud, senior manager, API management & web services, technology platforms & engineering at Amadeus, and Xavier Gardien, head of portfolio and product management, technology platforms & engineering at Amadeus, in a blog post. "In this big data era, knowing the number of transactions, response times on APIs, or the page travellers are spending the most time on with a mobile app could be invaluable to make the informed decisions that help us maintain an edge over competitors. This also serves as a great feedback tool to closely monitor where the industry is heading."

Whether for insights, to meet SLAs, to help ensure APIs are delivering excellent developer experiences, or to help ensure business-critical services are available, enterprises rely on API monitoring and analytics for API transaction loads both small and large—and they can be *very* large.

During 2018's Black Friday Cyber Monday (BFCM) period, the number of API calls for Apigee's retail customers (excluding those who host the platform on-premises) reached a peak of 108,000 transactions per second. This sort of exceptional digital activity—and the millions or even billions of dollars it can generate—is only going to increase: API calls to Apigee's platform, which maintained 99.999% uptime through BFCM, grew 95% compared to the same five-day span in 2017.

Apigee Edge's integrated monitoring capabilities enable users to precisely find the source of an API error—whether it's a developer application, the proxy layer, or a backend target. Navigating from an alert notification, users can diagnose an error in just a few clicks; with traditional tools, this requires toggling between and cross referencing multiple systems.

API monitoring and analytics checklist

Here are some key monitoring and analytics capabilities that businesses should consider when evaluating API management solutions:

- **Monitoring Across the API Value Chain:**
  - ☐ Provide in-depth insights into API availability and performance metrics
  - ☐ Enable users to drill down into granular details such as latencies and errors caused by proxies and backend targets
- **Precisely Diagnose Issues:**
  - ☐ Investigate API issues quickly without toggling among multiple tools and correlating debug sessions and log sources
  - ☐ Precisely diagnose the source of errors, whether in the developer application, proxy layer, or backend target
  - ☐ Tree-map views for network operations center (NOC) teams to visualize issues
- **Generate Contextual Alerts:**
  - ☐ Provide insights for users to take appropriate actions in the context of the issue being investigated
  - ☐ Facilitate grouping of proxies and targets to monitor business critical APIs
  - ☐ Support alerts by webhooks and other channels such as Slack, PagerDuty, or email
- **Gain API Insights:**
  - ☐ View overall traffic for all of the APIs in an organization, watch how developers engage with your APIs, and see which apps receive the most traffic
  - ☐ Monitor and compare traffic for specific API patterns across multiple API proxies; understand changes in API traffic relative to business, marketing, or partner events
  - ☐ Identify spikes or dips in API traffic and gain insight into what is happening around the time of the anomaly; see API proxy traffic patterns and processing times

About Apigee API Management

The Apigee API management platform delivers full lifecycle API management to help businesses unlock the value of data and securely deliver modern applications. Apigee offers a rich set of capabilities to enable enterprises to gain control over and visibility into API traffic, including the ability to automate troubleshooting and problem resolution and to derive insights from API usage. Learn more about Apigee's API monitoring and analytics capabilities.

Now that you've finished reading, why stop learning?
Visit the Apigee website for more.

apigee

Share this eBook

on social

with a colleague

Google Cloud