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 03

Understanding developers—internal and external—as API customers 05

Field-tested best practices 07

- Build an easy-to-use, self-service developer portal to drive adoption 07
- Create a community of developers 08
- Never stop improving 09

Real-world use cases 10

- How AccuWeather reached new audiences with its developer portal 10

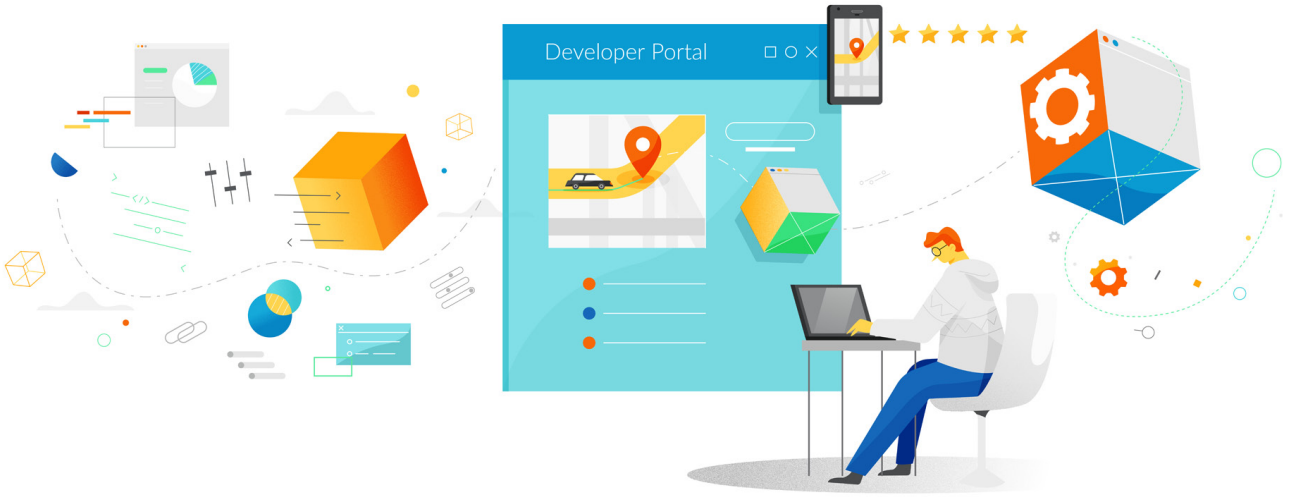
Developer experience checklist 11

About Apigee API management 12

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.

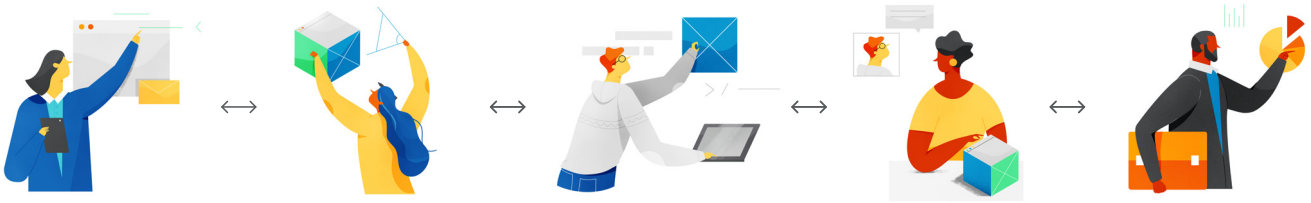


Developers have launched new ridesharing services by combining third-party mapping and navigational APIs with their own first-party data and functionality, for example. Many applications rely on third-party APIs, such as those from Twitter or Google, for authentication. When developers want to add voice 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:

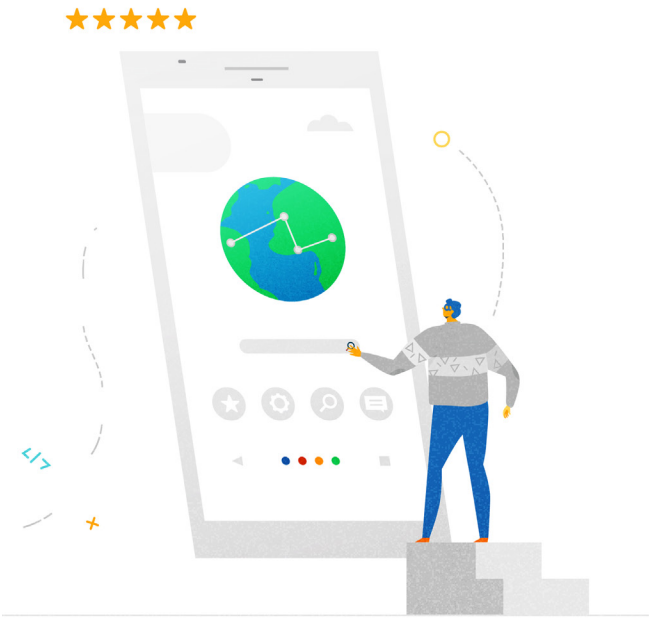
- **Public:** The API is available for anyone to consume, e.g., a branch locator API. Enterprises typically use public APIs to either monetize valuable proprietary services that may be of use to outsiders or to encourage adoption of their services within new developer communities that can help the enterprise 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

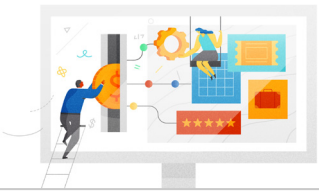
Here are some key capabilities that businesses should consider to provide great developer experiences that will help grow their API programs:

- **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

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 build customized developer portals, drive ecosystem growth through API adoption, help deliver world-class developer experiences, and manage APIs as products to empower developers. Learn more about Apigee's Integrated Developer Portal.

Now that you've finished reading,
why stop learning?



Inside the API Product Mindset - Part 2

Maximizing the Business Value of Digital Assets Through API Monetization

Inside the API Product Mindset - Part 3

Building and Managing Secure APIs

Inside the API Product Mindset - Part 4

Optimizing API Programs with Monitoring and Analytics

apigee

Share this eBook

on social

with a colleague

Google Cloud