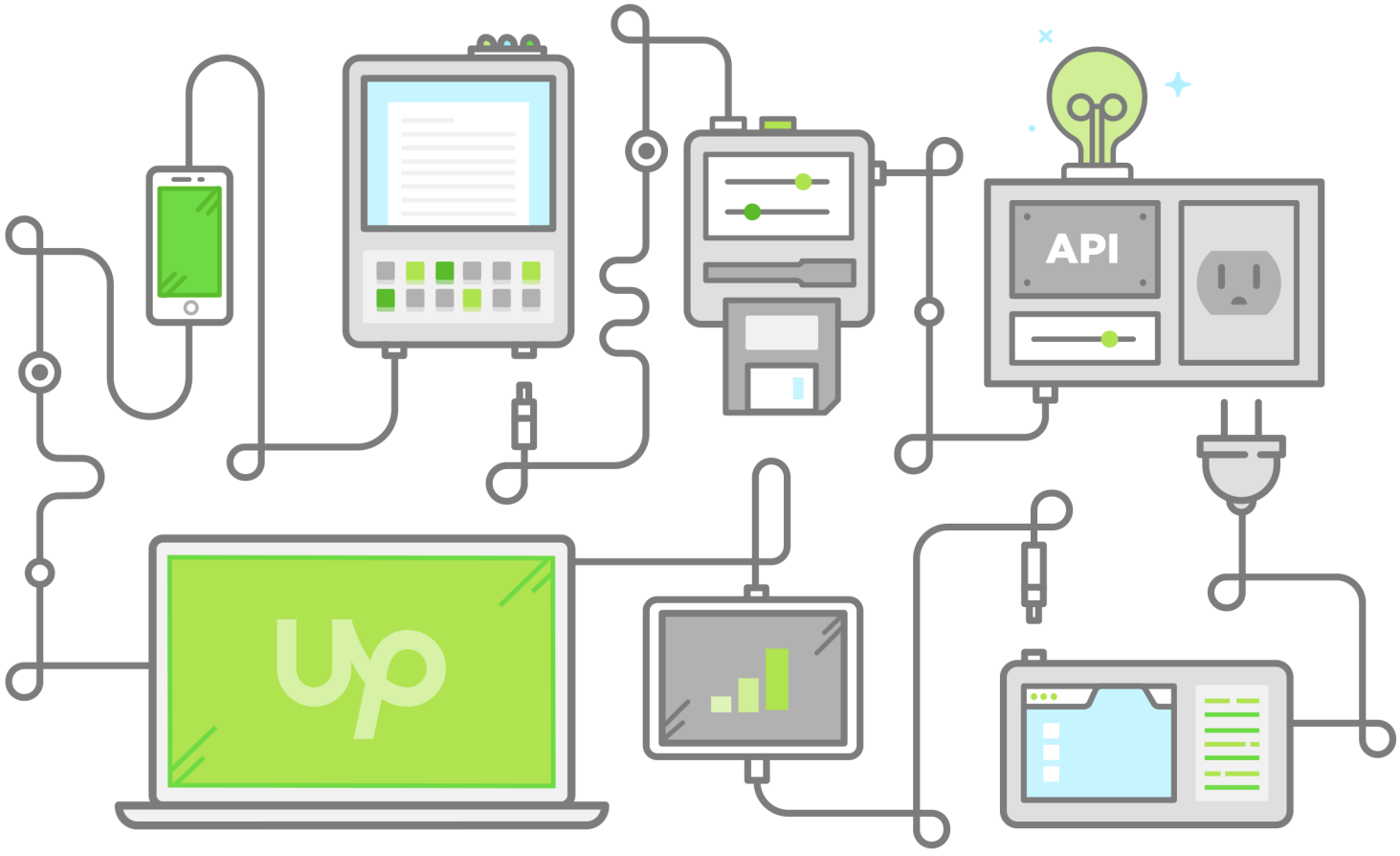
THE API ECONOMY

How to Plug In and Build Your Business



upwork

Foreword

Programming languages have been using application programming interfaces (API) since the 1960s—in the days of the mainframe. In fact, for all developers, understanding the concept of APIs is a cornerstone of their professional training. They have long known about the power and possibilities of APIs, and after reading this ebook, you will too.

If APIs are not a new concept, you might be wondering, "Why am I hearing more and more about APIs right now?" The reason is that software is not only becoming more ubiquitous, it's also becoming more interconnected. Without APIs, the digital experiences that we expect every day as consumers wouldn't be possible. For businesses, investing in a proper API strategy can pay significant dividends.

The API, in its simplest description, is a contract that allows software to communicate with each other and share information. In 2009, I led the mobile engineering teams at eBay—one of the world's largest online marketplaces. We built eBay's mobile experiences for the web, iOS, Android, Windows Phone, and BlackBerry. The technology that enabled a mobile user to interact with the eBay brand—for example, placing a bid in the mobile app and then having that action appear simultaneously on the ebay.com website—was the API.

The mobile app proved so successful that in just three years, the gross merchandise volume of eBay mobile grew from hundreds of millions to *billions* of dollars.

By 2012, eBay was responsible for almost one-third of the world's estimated mobile commerce due to its forward-thinking API strategy. If the company had not invested in a service-oriented platform and a thoughtful API strategy in the years before the rise of mobile, they would have lost that opportunity.

Mobile, however, is just the tip of the iceberg of a larger Cambrian explosion of devices. With the general ability of our electronics to connect to the internet, software and APIs are now everywhere. They power our phones, our televisions, our watches, and even our cars. The media calls this movement the Internet of Things. But the truth is, internet connectivity is *everywhere*, and with an API, your brand can be too.

This ebook provides the groundwork for what an API is, what its applications are, and how to build an API effort. From this foundation, you will understand how APIs are changing the world, and how you can use them to change yours.

Han Yuan

Senior Vice President of Engineering at Upwork

TABLE OF CONTENTS

PART 5

Talking Tech: What & Who You Need to Build a Great, Engaging & Secure API

PART 6

How Is Your API Performing?

Conclusion

Introduction

APIs (Application Programming Interfaces) are responsible for nearly everything we do on the web. They are how data connects from one place to another, then to your device. So with just a few taps or clicks, you can do things like order a pizza, book a hotel, rate a song, or download software.

APIs work quietly in the background, making the interactivity we expect—and rely upon—possible.

Here's how it may look in everyday life:

You're searching for a hotel room from an online travel booking site. Using the site's online form, you select the city you want to stay in, check-in and checkout dates, number of guests, and number of rooms. Then you click "search."

As you may know, the travel site aggregates information from many different hotels. When you click "search," the site then interacts with each hotel's API. Using the API, the travel site asks each hotel's database for all available rooms that meet your criteria. The results are then delivered back to the travel site. Within seconds of initiating your search, you see a list of available options with additional information like price, location, and other variables.

Think of an API as a messenger that runs back and forth, communicating between applications, databases, and devices. They keep us connected, so we can easily access data worldwide.

APIs Grow Stronger Digital Businesses

While developers bring APIs to life and are the primary consumers, APIs at their core are *business-driven* technologies—not tech-driven. They help companies work faster, more nimbly, and more cost-effectively. The people who should be thinking of ways to utilize APIs are marketers, business people, and product owners. That's why this ebook is geared towards you, the non-developer. When directed by your strategy, an API can grow to be an entirely new line of business for your company.

Fuel Greater Flexibility with APIs

APIs are like the connective tissue that links businesses, applications, data, and devices. They offer unprecedented flexibility to the way we develop software, build applications, share data, and even in how we engineer IT infrastructures. What's more, as mobile device and mobile app traffic continue to outpace desktop traffic, APIs will be the driving force behind it all.

> API-first, mobile-second, web-third prioritization gives product development the flexibility to go where business is heading.

Delyn Simons,
Vice President, Developer Platform at Ionic Security

Increasingly, APIs are replacing traditional websites and complex web systems in their ability to deliver safe, fast, secure ways to plug in. Many times, this creates a competitive advantage.

For instance, Salesforce is the leading customer relationship management (CRM) platform. Thousands of businesses worldwide integrate Salesforce seamlessly within their own workflows, get automatic updates, and adapt the scalable software to meet their varying needs. Salesforce can offer all this through its API, which enables a business to plug directly into Salesforce's system.

Streamlining Back-End Systems with APIs

APIs aren't just for sharing data between companies. Private APIs are revolutionizing how things get done *within* companies. By providing a "self-service" architecture, APIs give developers an easy way to plug right into back-end systems, data, and software, letting engineering teams do their jobs in less time, with fewer resources. This convenience can streamline a company's existing IT infrastructure, the way internal teams collaborate, and how partners integrate. It's called "consuming your own API" and it's one of the biggest benefits of API technology.

In this ebook, we'll start by breaking down the technology of APIs so that you understand what they do and how they work. Then we'll provide you with a critical—yet often overlooked—list of tips and best practices to consider, so you can create a winning API strategy, assemble a team to build it, and start measuring its success.

PART 1

Understanding the Basics of APIs

What Is an API?

You don't have to be an engineer steeped in API development to get a good grasp of what this technology does. Let's start by breaking down the acronym itself with the help of an analogy: withdrawing and depositing cash from an automated teller machine (ATM).

Application

Think of an application like an ATM. When you walk up to an ATM, you expect it will allow you to access your account and complete a transaction like withdrawing cash. Like an ATM, an app provides a function, but it's not doing this all by itself—it needs to communicate both with the user, and with the "bank" it's accessing.

An app deals in inputs and outputs. Your web, mobile, or back-end application is like a machine that solves a specific problem. The software* may be a customer-facing app like a travel booking site, or a back-end app like server software that funnels requests to a database.

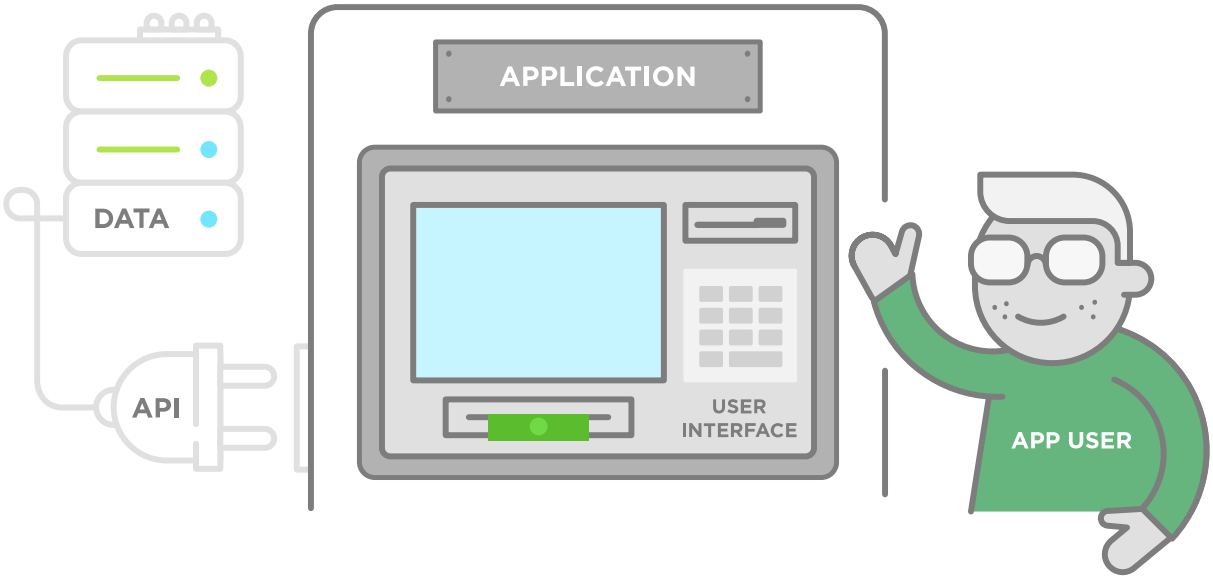
Programming

APIs allow the ATM to communicate with your bank. The programming is the engineering part of the app's software that translates input into output. In other words, it translates your request for cash to the bank's database, verifies there's enough cash in your account to withdraw the requested amount, the bank grants permission, then the ATM communicates back to the bank how much you withdrew so that the bank can update your balance.

**We'll be using the terms "application" and "software" interchangeably in this ebook.*

Interface

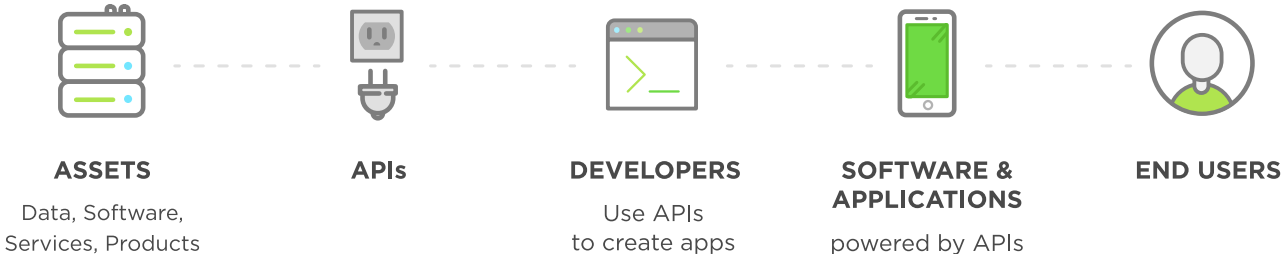
A user interface (UI) is how we interact with an application. In the case of the ATM, it's the screen, keypad, and cash slot—where the input and output occurs. We enter our pin number, punch in how much cash we'd like to withdraw, then take the cash that's spit out. Interfaces are how we communicate with a machine. With APIs, it's much the same, only we're replacing users with software.



In a nutshell, that's an API: an interface that *software* uses to access whatever currency it needs: data, server software, or other applications. In the case of the ATM, the machine is the end user of an API, not the customer pressing the buttons. It's the same in the digital world. However, it's important to remember that even though end users aren't an API's direct audience, without APIs, many of the apps we use every day would not exist.

The API Ecosystem

Before we dive into all that APIs can do, let's take a step back and define what an API's currency is and who (and what) is consuming it. This will put the role of APIs into perspective, and it will help you envision your own assets and audience in terms of an API.*



1. **Assets.** No matter what kind of API you're using or creating, it begins with shared assets. Assets can be data points, pieces of code, software, or services that a company owns and sees value in sharing. These could be links to Amazon's music library, Pinterest, or AccuWeather. There has to be inherent value in sharing these assets and a considerable demand for them (unless you're creating a private API, which we'll talk about later on). These assets reside on a company's server or database, so the next step is making them available to outside parties.

2. **The API.** Next is the API, which acts like a gateway to the server for any outside entities. It provides a point of entry for your audience—developers who will use those assets to build their own software—but it also acts like a filter for those assets. You never want to open up your entire server and all of its contents to the outside world. That's where APIs come in: they only reveal what you want them to reveal.

**Note that the API value chain will be different between public (open) and private APIs. We'll cover the difference between the two in the next chapter.*

3. **Developers.** The immediate audience of an API is rarely an end user of an app; it's typically the developer creating a software or an app around those assets. This is where assets take flight, yielding creative, new ways to implement data that previously may or may not have had any real business value to its owner. It also lets developers leverage reusable software components so they're not repeating work that's already been done. For public APIs, you're giving other developers creative, innovative ways to use your assets. For private APIs, you're giving internal developers a self-service portal that lets them do their jobs better and faster.

Assets: The currency of APIs and modern app development

Assets are the currency of any API. They're the purpose behind an API, they drive its value, and they also dictate its audience and its design. Assets are also the starting point of the API value chain. They can be anything a company wants to share—whether that's internally between teams, or externally with other developers.

Assets can be data points, pieces of code, software, or services that a company owns and sees value in sharing. Examples include Google Maps, Facebook, and Twitter.

Assets often need to be protected—especially secure data and financial information. In this respect, an API becomes more like a guarded gate than a doorway. It's designed to both share and protect assets, giving only certain users access to specific assets.

4. **Apps.** The best apps out there are powerful because they're connected to data and services. In turn, this allows them to provide richer, more intelligent experiences for users. And these powerful apps are only made possible thanks to APIs. API-powered apps are also compatible with more devices and operating systems, providing more seamless experiences.

5. **End users.** The beneficiaries of these apps are the end users themselves. The apps enable end users tremendous flexibility to access multiple apps seamlessly between devices, use social profiles to interact with third-party apps, and more.

Apps vs. Websites

Mobile and device traffic continues to overtake desktop and browser-based traffic. As a result, the way we use the web has trended away from websites and more toward apps. What drives the technology of web and mobile apps? APIs.

For some businesses, traffic through their APIs is already eclipsing traffic to their websites. That's because APIs replace the need to interact with a back-end system. Compared to an app, traditional websites with calls to the server and file transfers are slower and more tedious to build. An app can just plug into an API and access a multitude of data and software.

An API can also help improve, secure, and streamline IT infrastructures by adding an extra layer between the back- and front-end stacks. It's no surprise that apps quickly become way more desirable (for both parties) than a website. More and more, if you want to give customers what they want and keep up with the competition, you *definitely* need an API.

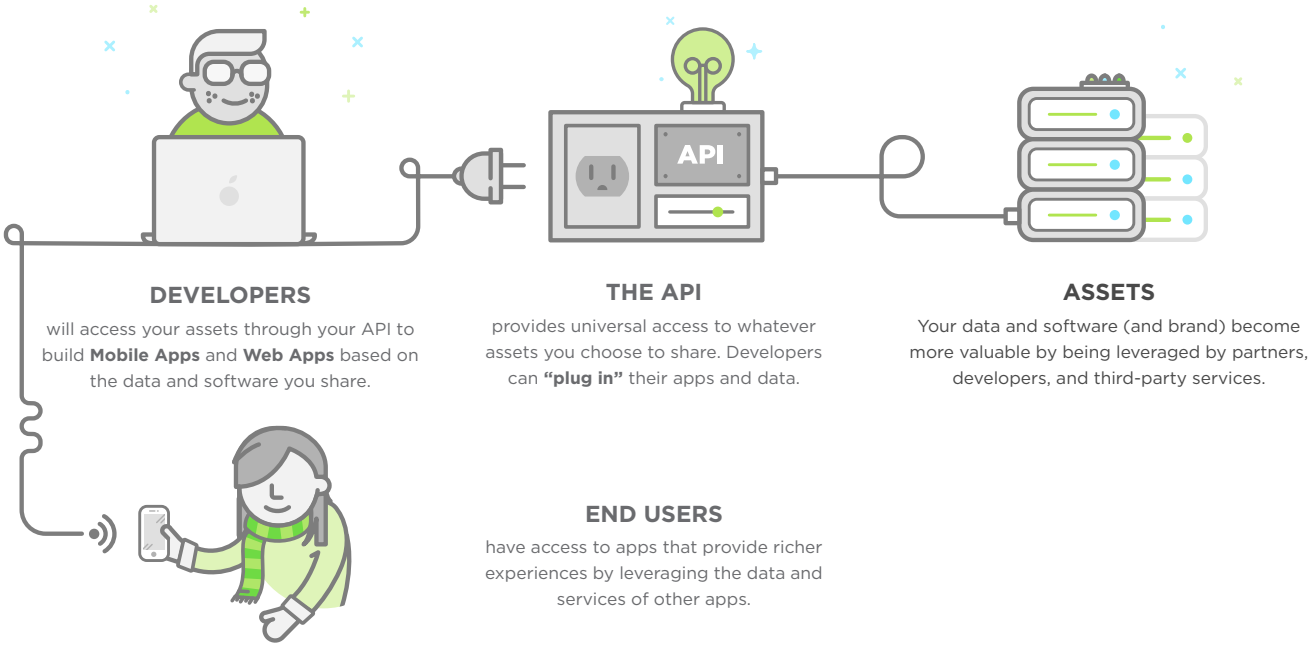
What Do APIs Do?

Now that you have a big-picture idea of how an API plays into the user-software-developer landscape, let's take things a step further. There's a lot more to APIs than meets the eye—including a multitude of uses and benefits that make them indispensable to modern app development and IT infrastructures. They're also able to drive customer retention, increased engagement, and entirely new revenue streams.

A website uses a URL address to make a call to a server and pull up a webpage in a browser. APIs also facilitate calls to a server, but they execute it more simply. They connect the web, allowing developers, applications, and sites to tap into databases and services—much like open-source software. APIs do this by acting like a universal converter plug offering a standard set of instructions.

How Do APIs Work?

APIs *are* programming interfaces, but they're also tools with endless business opportunities. Here are a few other ways to think about APIs and how they can function.



1. **APIs act as a doorway that people with the right key can get through.** Want to give specific people—but not everyone—access to your assets? An API acts like a doorway to your server and database that those with an API key (or a paid subscription) can use to access whatever assets you choose to reveal. You can also set specifications, rules, and levels of administrative rights to determine who gets through that door, how often, and what they're leaving with. A key could give a user read access, write access, or both—it's up to you.

2. **APIs are like a contract.** When you first build an API, it's like creating a contract with the developers you're giving access to. That agreement defines your API and has to remain intact throughout the lifecycle of your API. You can add features to change specifications with new versions (e.g., making your API accessible through a "freemium" plan), but you can't remove existing features from it.

APIs *must* stay backwards compatible, or you risk breaking the apps developers have built with your assets. By giving access to your assets, you're creating an agreement users can rely on. We'll talk more about backwards compatibility and versioning later on.

3. **APIs let applications (and devices) seamlessly connect and communicate.** An API can create a seamless flow of data between apps and devices in real time. This not only lets developers create apps for any format—a mobile app, a wearable, or a website—it allows apps to "talk to" one another. This is also how an app like Spotify works seamlessly on whichever device you're using (e.g., your laptop and your smartphone), and remembers what you listened to last. Uber has integrated its ride requesting service with partner apps via its API, giving users the ability to schedule rides within other apps—for example, when booking flights on United Airlines, planning travel with TripAdvisor, or making dinner reservations through OpenTable. This is the heart of how APIs create rich user experiences in apps.

4. **APIs let you build one app off another app.** Entire businesses and popular web applications like Hootsuite, Zapier, and IFTT (If This Then That) have been built solely on creative ways to leverage APIs. APIs allow you to write applications that use other applications as part of their core functionality. Not only can developers get access to reusable code and technology, they can leverage other technology for their own apps. TweetDeck is an excellent example of an app that takes the core functionality of Twitter, then expands on it with more features and its own unique interface.

5. **APIs act like a "universal plug."** What if all of those people with keys to your door speak different languages? With an API, it doesn't matter. Everyone—no matter what machine, operating system, or mobile device they're using—gets the same access. Think about those universal outlet plugs that let you use an appliance in any country's socket. An API is a lot like that; it standardizes access. Once a user enters their API key, the API determines what level of access they have, then the assets they request are packaged up by the API in a language their software can understand.

6. **APIs act as a filter.** Security is a big concern with APIs—after all, you're giving outsiders access to your servers and all they contain—which is why they have to be carefully constructed. APIs should give controlled access to assets, with encryption, permissions, user management, and other measures that keep too much traffic—or malicious traffic—from compromising your data or server. Also, certain assets need to be shielded while others are revealed, like allowing people to check out books from only certain parts of the library. This is a very important API design consideration for industries that are heavily regulated like health care and finance.

Now that we've seen a range of what APIs can do, let's take our understanding of APIs a step further with a look at their two biggest differentiators: public and private APIs.

APIs and the Cloud

A majority of organizations have some or all of their operations in the cloud, whether it's a public cloud, private cloud, or a hybrid of the two. APIs make the transfer of data between these components seamless and secure.

Because the cloud introduces a number of complexities from a development perspective, APIs are even more important in connecting your back-end systems to the cloud. Also, the cloud can be notoriously vulnerable, and the transfer and storage of massive amounts of real-time data (say, from Internet of Things devices) presents even more entry points for attacks. Having strong encryption measures and a secure API are paramount to your cloud strategy.

PART 2

Private APIs vs. Public APIs

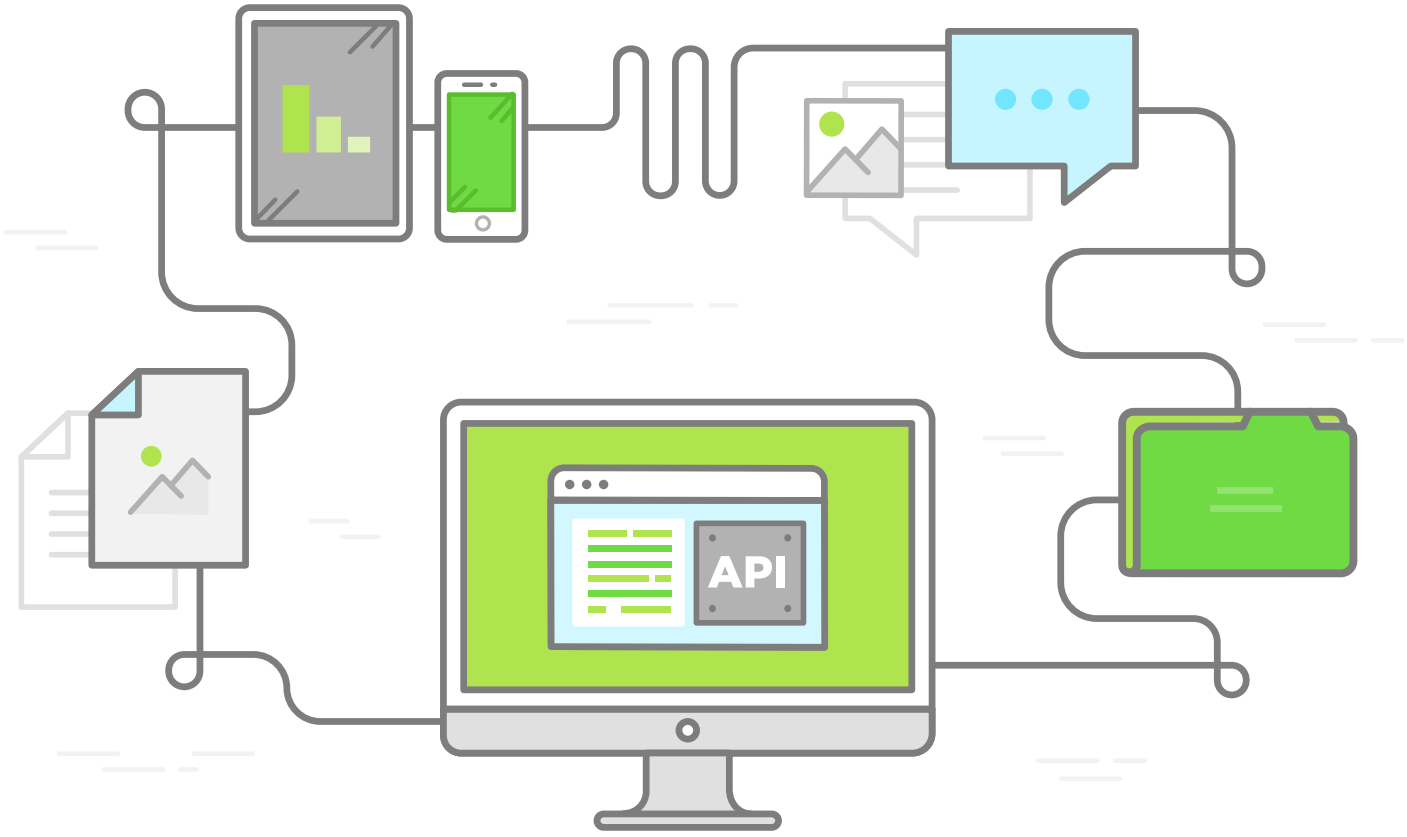
Public vs. Private: What's the Difference?

Before we get into the bigger picture of how APIs are having such a big impact on the way we do business, we have to address a very important fork in the road: **public APIs vs. private APIs**. Even though they're essentially built the same way, they have different value chains, different audiences, and vastly different business values. In this section, you'll see some of the values, risks, and rewards of each, and be able to determine whether one or both are right for you.

PRIVATE API	PUBLIC API
Better IT infrastructures	Apps compatible with multiple devices and operating systems
Efficiency with reusable building blocks of software	Opens up your data to third-party developers to innovate
Faster iteration, testing, and innovation with a "self-service" approach to shared data	Gives users faster, seamless access to files without web services or file transfer
Less stress on network and web services when replaced with an API	Opens up data and software, so security, privacy, and rights management can be a concern

A *public API* is probably what first comes to mind when you think about APIs: the Twitter API, Facebook API, Google Maps API, and more. But these are only the tip of the API iceberg. *Private APIs* are far more common and prevalent and are designed only for use within an organization and its authorized partners. At the time of this ebook, there are over nine million developers working on private APIs, and around 1.2 million working on public APIs.

Let's see how they can apply to your company and the advantages of each...



Private APIs: The Self-Service Developer & Partner Portal

A private API makes internal, shareable assets more of a **self-service user experience** for your company, while a public API focuses on tasks like expanding a brand and cross-device compatibility.

A private API is all about rethinking your company's IT architecture, saving time and resources, streamlining collaboration, and giving your developers unfettered access to everything they need without hassle. More companies see value in "consuming their own APIs." In fact, of the billions of API calls made, around two-thirds of those calls are companies making calls to their own APIs. Companies mainly use private APIs to improve their agility, flexibility, and velocity. This often translates to remaining competitive, higher productivity, and improved efficiencies.

Private APIs can also reduce operating costs and create new revenue streams. They can eliminate the need for layers of web services, back-end applications, and server resources, which can speed up production for your development teams or your partners who have access. On the partnership front, you can create versions of your private API that are customized for your partners, enabling faster technical integrations.

Things you can do with a private API:

- Build internal apps for company use
- Build customer-facing apps with internal assets
- Create a shared pool of data and assets that allows teams to collaborate faster and easier
- Strengthen partnerships, allow potential partners to test out integrations, and streamline technical integrations
- Build internal apps around a microservices model

Reasons private APIs are great:

- Self-service style access to data and software gives teams greater agility, and creates a less siloed environment
- Rapid and scalable development fuels innovation (because there's less need to build things from scratch)
- Faster mobile development
- Simplification of IT infrastructures
- They facilitate better business development because partners can easily integrate.

Public APIs: Granting Outside Access to Your Assets

Open APIs are all about letting the outside world in. They provide a set of instructions and standards for accessing the information and services you're sharing, making it possible for developers to build an application around those assets. This entire concept (and APIs in general) is somewhat borrowed from the idea of open-source software: create it, open it up to users, then let them run with it.

Some public APIs are even growing faster than original lines of business. Companies like Salesforce are seeing business through their APIs outpacing their original focus, reportedly generating 50 percent of their revenues through APIs.

Things you can do with a public API:

- Publicly share your assets with anyone you want
- Make software solutions you've created "open source"
- Create a seamless flow of data to and between devices
- Forge new partnerships and integrations
- Facilitate data distribution
- Get information and analytics about audience, users, and traffic

Reasons why public APIs are great:

- They encourage experimentation among developers. Devs can focus on their own unique app functionality and surround it with fully functional, distributed processes developed by others, accessed through APIs.
- They make it easy to promote and extend your brand.
- They offer a streamlined alternative to interacting with your back-end system.
- You're contributing to the API value chain, not just extracting value from it.

Examples of Public APIs

- **Twilio** is an excellent example of how apps can improve their functionality by utilizing a software solution through an API. Twilio offers calling and text messaging functionality, enabling apps like Uber, OpenTable, Airbnb, and eHarmony to give users real-time, SMS text updates to their phones.
- **eBay** was one of the early public API pioneers. They've since built a series of APIs on top of their core API, for both the buying and selling sides of the business. These APIs give high-volume sellers an easier way to list items in bulk, schedule listings, handle accounting, shipping, and more. On the consumer side, eBay's APIs make their mobile app a rich, interactive tool, allowing users to check listings on the go, get updates to their mobile phones, and more.
- **YouTube's** API is all about enabling other apps to have embedded videos and video players. The API also grants in-app access (and playback controls) to the millions of videos on the site.

If you're leaning toward creating a new line of business with an API, it's important to know the risks and rewards. With a private API, you'll have fewer objections to overcome around security and privacy concerns.

PART 3

The New API Economy

7 Ways APIs Are Revolutionizing Digital Business

Now that we've covered what APIs are and can do, it's time to start thinking about all that they're capable of. Once you start to grasp the myriad of applications for API technology, you begin to see that APIs are rapidly changing more than just how apps function—they're revolutionizing digital business. Let's explore their potential even more by looking at some API use cases.

1. Create a new business, new experiences, and new helpful resources for users—or "mashups."

Often, app makers have the time, resources, and creativity to use an API provider's data sets or services in ways the provider never imagined. App makers can integrate different sets of data together to create a new functionality. For example, an app could pull weather and city data so that drone users have the location-based information they need to fly safely. It might provide flight conditions, wind, precipitation, and proximity to airports.

Another way to use assets is by merging one API with another to create a totally new experience. These are called "mashups," and when done right, they can have a lot of potential. Entire apps like Zapier have built successful businesses from a mashup. Zapier is a site that matches up social apps to let you automatically share things between your accounts when you set up an API-powered "zap." For example, users can create a zap that stores Foursquare check-ins in a Google Spreadsheet, or a zap that triggers a new message in Slack when a new commit is created in GitHub.

Using an API to share data between apps can help create a more complete picture from scattered bits of data. An app like Google Fit can import data from workouts you've done in fitness apps like Sworkit, Strava, or Nike+, giving users a more comprehensive look at their fitness.

2. Sharing data could create new revenue with minimal effort.

With data APIs, businesses can benefit by *sharing* what was once considered private, proprietary assets. In the API economy, companies that may benefit the most are the ones that can aggregate this data efficiently, then offer it to other organizations. Common types of data APIs are data set APIs, analytics APIs, and government data APIs.

For example, a city's open data API could offer a range of valuable information from its database, including traffic and transit information, hours for local facilities, environmental data, available parking lots, and more. You could create an app that makes apartment hunting easier by mashing up map data and available apartment listings.

Existing data can drive new lines of business through an API. By making your data available to outside developers, their mashups could create tailored experiences that drive engagement or more sales by predicting a user's preferences based on behavioral data.

3. Streamline application and software development with modular chunks of reusable code.

In the API economy, code is rarely written from scratch anymore. Instead, code is shared through APIs so that developers don't have to reinvent the wheel. You can leverage entire software solutions and other apps' information and services. This multiplies efficiency so much that you can build things in a click that once took weeks to code.

APIs also grant access to reusable chunks of code that can be assembled into bigger software projects. This became especially helpful when web components—the newest set of standards by the W3C (the World Wide Web Consortium)—simplified web development using modular, encapsulated bits of code. Through low-level APIs, web components allow developers to build complex web applications in smaller chunks.

This is also driving the use of **microservices** and service-oriented architectures (SOAs) in software development.

By chunking down complex builds, it streamlines development and breaks larger projects into manageable components. The approach also benefits complex databases. Instead of setting up massive database architectures, developers simply access existing data through an API.

> Service orientation continues to evolve beyond use of private, internal APIs. Microservices now support the breakdown of monolithic legacy IT architecture into smaller, independently deployable containers for DevOps teams to manage.

Delyn Simons,
Vice President, Developer Platform at Ionic Security

4. APIs are ushering out complex, monolithic IT infrastructures.

Aside from opening up software and services to outside developers, APIs play another crucial role in the way we run our businesses—specifically, our entire approach to IT infrastructures. Businesses, and specifically enterprise businesses, have traditionally relied on enormous IT infrastructures.

APIs offer massive benefits to IT architectures by adding a new layer of separation between users, applications, and server-side systems. Being able to plug right into a system via an API rather than making calls through complex web services is incredibly efficient.

Almost every business uses the cloud—whether for scalable storage, computing, or analytics—and APIs play an important role in connecting all of these components and safely sending data between them.

Microservices & the Rise of DevOps Teams

Breaking app functionality into more manageable chunks can reduce the time, cost, and complexity involved in large software projects, and APIs make it all possible. This approach is considered an easier way to develop large applications. One of the reasons is from a project engineering perspective. When the different components of an application are separated, they can be developed concurrently.

That's why the DevOps culture is replacing the legacy approach of hulking software projects. The new approach breaks down components and uses third-party services to build them. To achieve this, companies are changing more than just their software building blocks—it's a whole change in mindset. DevOps teams (a blend of development and operations teams) have to understand microservices as a concept, know how to strategize it, and have the structure in place to work with this model. The result is a more continuous software release cycle, rather than the long, drawn-out dev cycle of bigger projects.

5. APIs are driving new and better partnerships.

One of the most powerful reasons to build an API is to create or strengthen partnerships. This could increase your audience reach, boost sales, multiply leads, and more. APIs make it easier to strengthen existing partnerships and test potential ones because they eliminate a multitude of technical constraints previously associated with setting up integrations.

API-driven partnerships may provide you with more reliable partnerships too. Once a partner has written code and integrated your assets with their back end, applications, and analytics, they're less likely to end a business relationship in which an application hinges on your assets. In these cases, API-powered connectivity also acts like a contract, which gives you a strong hold on partnerships.

6. APIs are giving marketing teams the power to create data-rich campaigns and initiatives on the fly.

Need to spin out some data-driven landing pages, interactive advertising, or customized email marketing? Your marketing team can use APIs to access the data they need. This lets them iterate quickly and create new, data-rich campaigns without tapping into the engineering team.

Marketers who use location-based advertising are gleaning data from Google's Awareness API, which provides deeper insights into data collected from a user's smartphone. Marketers know details like activity, how the weather is where they are, if they have their headphones in, or if they're close to a beacon device. From this, marketers can personalize a user's experience by suggesting things like a nearby activity that's highly relevant to them at that time.

7. Offer customization of software and services so companies can create their own interfaces and versions that suit their needs.

APIs can grant access to the back end of a product so developers can create their own custom solutions. Different from a software-as-a-service (SaaS) product, an API can grant users higher administrative access that allows them to customize an app or software on a more granular level. SinglePlatform, a menu template software provider for restaurant websites, allows restaurants to seamlessly share their menus across different sites. Through its API, restaurant menus can be accessed via Facebook, Yellow Pages, Yelp, Foursquare, and TripAdvisor. For clients who need advanced integration and customization of their menus, they can access SinglePlatform's REST API and control how menu information appears.

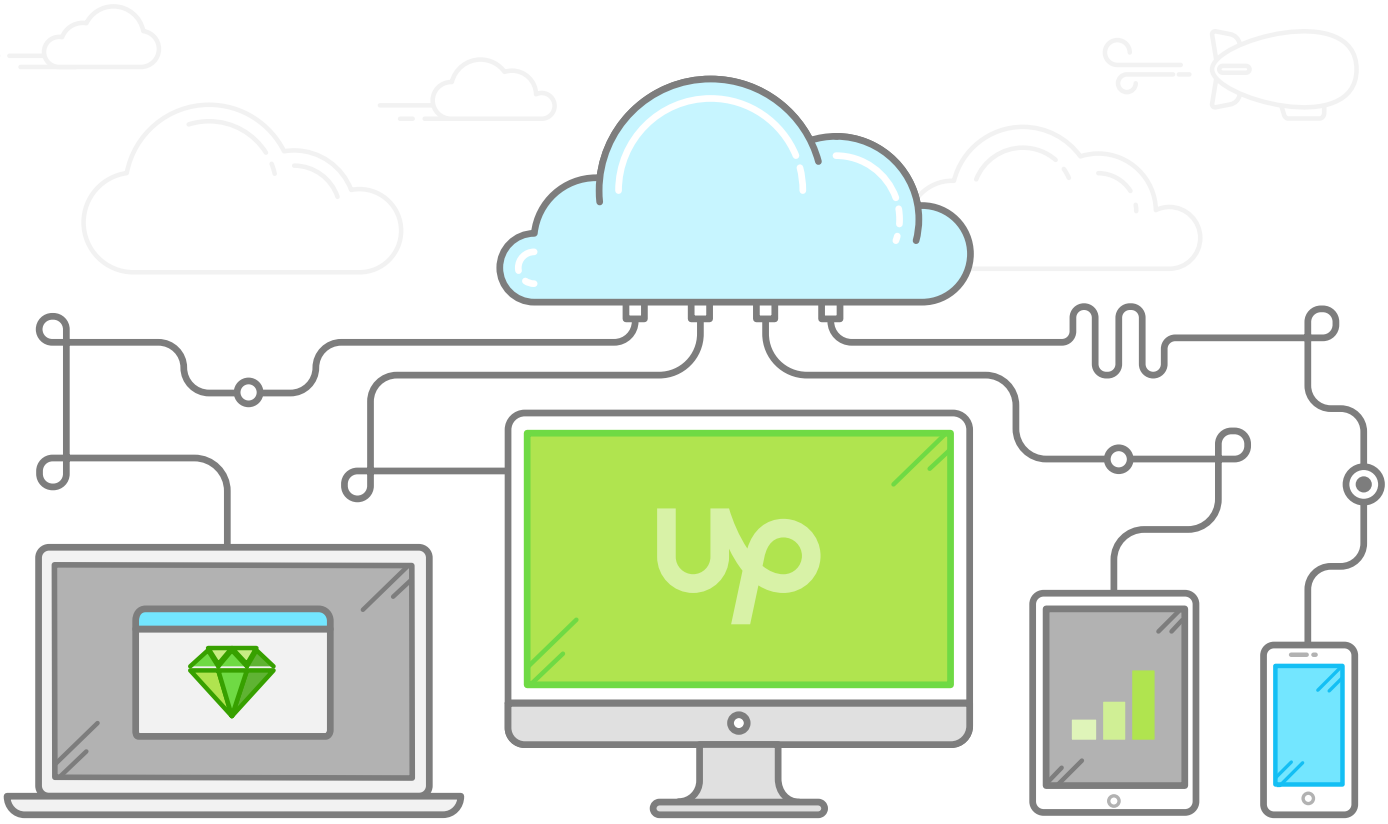
Customization is all about giving developers access to already excellent software solutions to adapt how they want to use them. Understanding this, Google's AdWords API allows users of its advertising program to build custom interfaces for managing their campaigns.

So, is an API the right decision for your business?

Now that you're armed with insight and you know more about using APIs to meet specific, strategic business needs, is building your own API right for you? In order to determine this, and have a successful API, you should begin with a clear strategy. Let's look at how you can get started.

PART 4

Diving In: Things to Know Before Building Your Own API



It All Starts with a Clear, Concise Use Case

If you have assets that others could use and benefit from, like a software solution or valuable customer data, a public API may be the road for you.

If you're looking to help your teams do their jobs more easily, and you want to be more agile in developing new and better apps and partnerships, then a private API could be excellent for you.

To know for sure, establish your use case, then design your API around that goal. Know which metrics you'll track and how you'll measure success. This will be important when pitching the API to other departments and executives.

Some reasons why you might build an API include...

- **The competition has one, but yours is going to be better.**
  Your API should be clearly differentiated from what else is out there, whether it has better data, more unique data, or is more focused on support and being developer-friendly.

- **You want more mobile market penetration**.
  An API makes your data available in a format that any developer can use, whether it's for an app or a wearable device. It allows your data to be where your customers are: mobile and on the go.

- **You need a second mobile app.**
  With an API, you won't have to recreate a lot of the work you've already done.

- **You want to drive use to your app or business, ramp up engagement, and build loyalty toward your product or service.**
  Get insights into your customers, give them the ability to interact with your app, or make user experiences more intuitive with better data.

- **You want to monetize your API.**
  Some companies are entirely API companies, but your use case could simply be monetizing an existing demand for your assets. For example, cloud-based app deployment provider Heroku offers free access to its API to start, but as calls to the API ramp up, users have to pay. This is a good model for rapid prototyping and startups because starting out as a free service will eliminate barriers to entry.

- **You want to streamline your IT architecture and internal collaboration.**
  Use a private API to communicate internally and streamline collaboration. Big teams can work on things separately, but there's value in interaction between different teams.

APIs: Where the CMO's and the CIO's Roles Meet

We've already talked about how APIs are business-driven technology. APIs can also be a highly valuable piece of your marketing pie—especially if you're planning to add more mobile components to your strategy, want to spin out targeted marketing and data-rich advertising campaigns, or grow through networking partnerships like Airbnb, Snapchat, and Instagram have.

Think of the API as the intersection between the interests and goals of the chief marketing officer (CMO) and the chief information officer (CIO). Sometimes, what the CMO may want to do with a company's data can hit security or technical barriers. Or it may require more of the engineering team's time than they can spare.

If the CMO and CIO create the strategy together, it will make development smoother from the start.

Who else to bring to the table:

- Legal experts
- Marketing team
- DevOps team
- Project manager

Addressing Legal Concerns & Internal Objections

Legal issues, rights, terms of use, strategy, partnerships—these are all things to consider when thinking about your API. Starting with your assets, it's important to establish what rights you have to share them through your API, and how other people can use them. For an API that's based largely around sharing content, both of these will have major implications on the design of your API.

If you don't own your assets or have explicit rights to all of the content you share, **your legal team** may suggest you take these or other steps:

- Include a rights management system, establish contracts, or build in "rights tagging," which tags ownership of content on sites or in feeds through scripts.
- Create a terms of use.
- Create a privacy policy. If your assets are either personal data of customers or data of a sensitive nature, a privacy policy for your API is a good idea.

What Makes a Great API?

APIs can be badly designed. To design a successful API, you'll want one that snaps in perfectly to your code, is built beautifully, and is easy to use. If you want people to love your API, consider these parameters.

A great API is...

- **Intuitive and easy to use.** APIs should always give developers ways to do their jobs faster, more easily, and with less resources. You don't want to make developers scratch their heads when they're trying to use your API or they won't stick around for long. Developers' user experience (UX) is incredibly important with an API, so make it intuitive. Assume all developers using your API are too busy to learn it—it should appeal to them in how easy and simple it is to use. Take Twilio, for example. They've engineered their service so users can make a call to their API in around five minutes. The better the dev UX, the more likely they are to stay engaged. Continued engagement with an API makes for a successful API.
- **Well-documented.** A big part of making an API easy to use is providing plenty of documentation that explains how your API works. Make sure it's searchable, always up-to-date, and so complete that developers are able to solve problems on their own. Show tips, how to do certain things, and so on. Remember, simple questions should be simple to answer.
- **Able to deliver great service, even when traffic spikes.** Most APIs monitor traffic and restrict access to a server when volume spikes. It's an important way to control costs and protect your server from failing due to an overload of traffic, and to ensure speed and quality to other API users.
- **Safe and secure.** What identification, authorization, and authentication measures will you design to protect your assets and control who can do what with your API? We'll cover this in detail in the next section.

In the next section, we'll dive into some of the more technical aspects of building APIs so you'll be able to navigate conversations and decisions with developers, product managers, and your legal team to get your API built to your specifications.

PART 5

Talking Tech: What & Who You Need to Build a Great, Engaging & Secure API

Laying the Groundwork for Your API Strategy

In this section, we'll cover **API design** considerations, who you'll need on your **API development team** to get started, and how to plan for the **safety and security** of your API.

But, before the first line of code is written, a few key things need to be established up front. What will the operating model be? How will you address things down the line like user management, security, and those desirable but challenging spikes in traffic? What will your authorization and authentication measures be?

Things That Influence the Design of an API: Assets, Audience, Strategy, and Versioning

Let's start with a checklist of things to nail down before designing your API. It's important to treat an API project just like you'd treat a software project. You can choose just about any architecture and any schema, but none of that will matter if you haven't planned the following up front.

1. **Use case**
   This is the main reason you're building your API. Is it for mobile market penetration? Monetization? Strengthening partnerships?

2. **Audience**
   Who are the developers you're creating it for? What tech do they use? And how will they be using your API, and what actions should it perform? It's crucial to know your audience up front, and design your API specifically for them. Their UX will be a major factor in the success of your API, and they'll also drive a few of your more technical decisions.

3. **Affordances**
   What can people do with your API (now, and later)?

4. **Schema**
   This isn't the "be-all, end-all" of your API's design, but it's important. Your use case will drive the most effective schema, which lets you organize your API. How will your data be formatted?

5. **What do your users want?**
   You'll model your schema based on these parameters. You'll also need to think about the design of your user interface.

6. **Do you need to set use limits on your API?**
   Most open APIs (both public and private) need some line of defense from overuse and abuse to protect the server and control costs. These can be anything from rate limits and throttling to data transmission limits and call volume by application limits. Discuss this with your customer service and DevOps teams to help you anticipate volume and establish the limit that works best for your business needs and users.

7. **Will you use caching for performance issues?**

8. **Will you use a data center, or the cloud?**

9. **REST or SOAP?**
   These two "styles" of writing APIs really speak to the architecture of an API, each with its own benefits and implications for how your API will communicate with the server. REST APIs (Pure REST, Pragmatic REST, and more) are based on the HTTP protocol and can be simple to build and scale; SOAP (Simple Object Access Protocol) is a bit more complex. Reasons you may want to build an API to be RESTful include connectivity, discoverability, data health, and scalability.

10. **JSON or XML?**
    Which data model is right for you? JSON (a subset of JavaScript) is very popular for APIs because it's more compact and can interface well with JavaScript-based web apps. XML, while more powerful, requires more work from programmers.

11. **Are you trying to monetize your public API?**
    The traffic measures listed above can also be used to let you monetize your API. If someone wants more calls than their existing agreement allows, they can pay for more—and so on and so forth.

Who to Engage: Assembling Your API Team

Next, you'll need to assemble your API development team. Think of your API like a software project, which means you'll need both **product managers** and **developers** to make it a success.

The best API developers have a mix of technical and interpersonal skills. Why? Because not everyone will agree that sharing your company's data and software is a great idea. You'll need to have devs on board who can not only design well, but also communicate the value of your API well. They'll be juggling a lot: coding a beautiful API, getting in the heads of the devs using the API, solving user problems, and adhering closely to your company's security and legal concerns.

Consider API developers versed in:

- **Key front-end development technologies like JavaScript** They're building this API for other developers, after all.
- **Back-end development technologies**
- **HTTP, HTTPS:** Transfer protocols and encrypted protocols are a cornerstone of API technology.
- **RESTful API design, or SOAP:** REST is an architecture that's more data-driven, while SOAP is a standardized protocol for transferring structured information that's more function-driven. SOAP relies more on XML, while REST tends toward HTTP and JSON.
- **JSON or XML:** These are both ways to format data, and common skills in web development.
- **API traffic** and reliability like setting rate limits, quotas, and throttling
- **API security** like OAuth delegation protocol, or other authentication measures

API Traffic Control

What's the difference between rate limits, quotas, throttling, and spike arresting?

Most APIs restrict access to a server by volume for a period of time with **rate limits**, **quotas**, **throttling**, and **spike arresting**. They're all common means of monitoring traffic through an API, and an important way to control costs and protect your server from failing due to an overload of traffic and to ensure speed and quality to other API users.

Rate limits control calls to an API during a period of time and stall an app's ability to make calls for a period of time if it's going over its limit. **Quotas** give users or apps an upfront number of calls they can make to an open API per second, minute, hour, etc. Exceed that quota in the given time period and an error message is sent back, cutting any further calls off until the time period rolls back over.

Throttling is a bit different, but also deals in restricting calls to an API. Instead of receiving an error message, users' calls to the API are slowed down if a certain number of calls are exceeded in a set time period.

Spike arresting is a built-in braking system that shuts access down if calls get out of control—either from malicious intent or just poorly written code. Spike arresting is a good idea no matter how busy your API is going to get, if only to prevent major disaster.

API Security: How to Protect Yourself When You're Opening Assets to the Public

Security is a top priority when creating an API. Whenever you're opening up your server and network, you've got to be certain you're only sharing what you intend to share, and you're not exposing anything sensitive or private, or creating vulnerabilities in your infrastructure. Like most security plans, your API security should be designed to prevent, detect, and react to any problems. That's why it's so important to consider early on in the decision-making process, and why it will continue to be an ongoing priority as your API evolves.

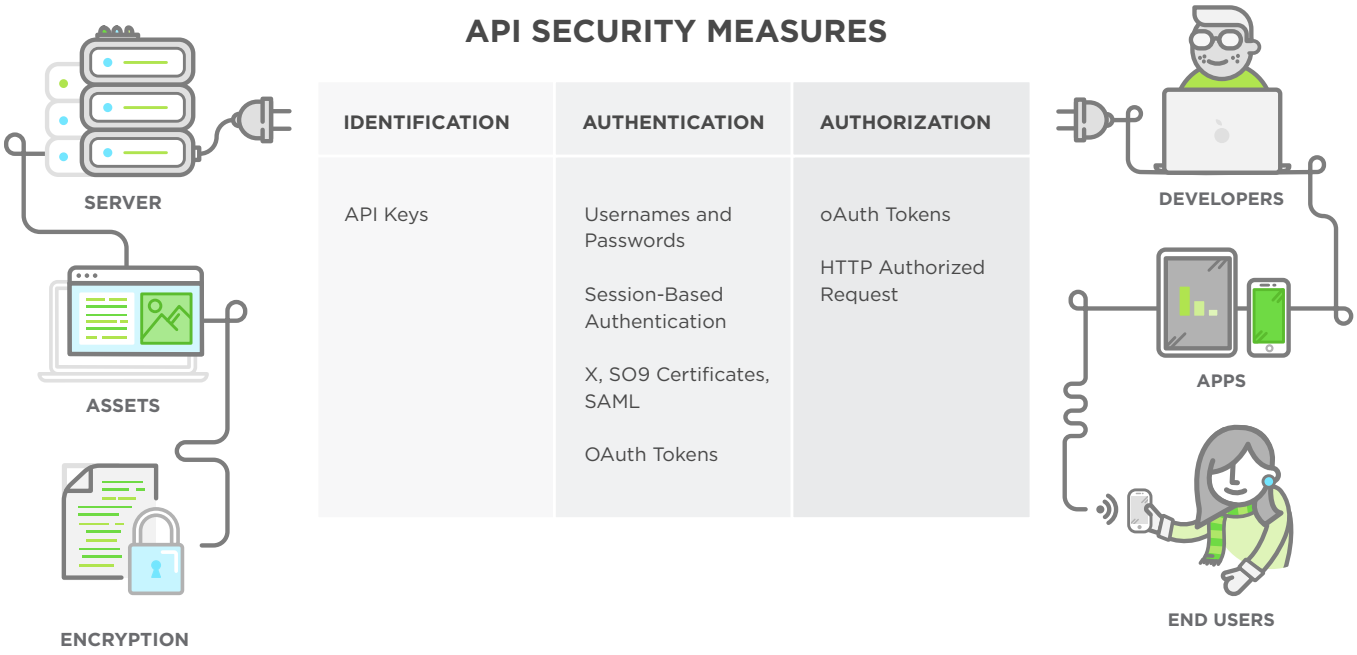
Security questions to consider:

- What API assets need to be protected, and how are you going to protect them?
- Security measures can impact API performance and programming—how much will they affect yours?
- Who's accessing your API, and do you need to know who they are?
- Will you need identification and authentication measures?

The Basics of API Security

There are **three main security measures** that most APIs use: identification, authentication, and authorization. You might not need all three, or you may use a combination of the three, depending on the sensitivity of your assets.

Before writing the first line of code, you must have a plan in place for what you're sharing, what you're securing, and who you're sharing this information with. How you program the API structure for security will directly impact other design and programming decisions. Be sure to discuss this with your legal team and other key people.



API SECURITY MEASURES

IDENTIFICATION	AUTHENTICATION	AUTHORIZATION
API Keys	Usernames and Passwords Session-Based Authentication X, SO9 Certificates, SAML OAuth Tokens	oAuth Tokens HTTP Authorized Request

Level 1: Identification: “Who Are You?”

Identification answers the question: “Who are you?”

API keys are unique codes that developers are given and can use to access your API. API keys are randomized, unique identifiers so there’s no need to mess with passwords. In addition to giving the developer access, an API key can give you the ability to monitor use and metrics. They’re incredibly easy to use and give you built-in analytics—a big bonus.

API keys aren’t encrypted, so don’t think of them as the last and final word in security. They’re a good place to start, and offer a general way to monitor and manage who’s accessing your API. They make it a cinch to shut off access to your API if someone is breaching terms of use or a glitch is causing too many calls to your API. For example, the spam filter plugin Akismet that’s commonly used on WordPress blogs can shut down access by an API key if it detects that the blog using that key is breaking its terms of use policy.

Level 2: Authentication & Authorization

Authentication: "Prove you are who you say you are."

This is how an API answers the question, "How can I determine that you are who you say you are?" By implementing things like usernames and passwords, you get the next level of security. Authentication can also affect throttling, allowing an application to make more or fewer calls to the API based on authentication method.

- **HTTP Basic Authentication** involves authentication with an API key.
- **OpenID** is an example of an authentication technology, which redirects users to a site where they can enter a username and password, gives proof to the site that they are who they say they are, then directs them back.

Authorization: "What can you do with the API?"

Just because you're authenticated to use an API doesn't mean you can do just anything with it. Authorization defines user roles and permissions. For example, an "Admin" login would give you permission to do more within an API. Think of authorization like a driver's license in that it can offer different levels of permissions. If you're 18, a license lets you drive. If you're 21, it lets you drive and buy alcohol.

> **OAuth (and OAuth2)** is an example of authorization technology. Rather than just verifying identity, it grants access to protected resources or lets an app do things on your behalf without you having to sign in. OAuth is designed to bypass the need for a user to enter their username and password. Instead, it grants permission with access "tokens." Using tokens instead of passwords makes APIs less susceptible to security issues.

Encrypting API Traffic with SSL

Encryption should go hand in hand with any API design that's granting access to sensitive information. Using an SSL (Secure Sockets Layer) or TLS connection is another level of security that keeps API keys secure. It can be more complicated to set up, but is very important given most of this data is being transferred over wireless networks, which can be more vulnerable.

Can Your API Evolve? A Note About Versioning

The answer is yes, but it absolutely has to be backward compatible. As a marketer, you're always thinking about how to improve, update, and evolve strategies. An API is not unchanging, but there are restrictions. If you plan well enough in advance, your API may be "versionless"—for example, you could plan not to add in certain features that you know for sure will need updating down the line.

A few rules about versioning:

1. **You can add to, but not take away from, the functionality of your API.** You don't always have to reversion an API if you add a feature. However, the underlying functionality of your API cannot change or you'll risk breaking everyone's software that's built on top of it.

2. **When you do add new features, let your analytics guide you.** It's best to only add a few new features to an API at a time and only add ones that you know are in demand.

3. **Use a new version as an opportunity to fix bugs.** Without changing any underlying functionality, you can fix mistakes in previous versions with updates.

4. **Add a "mediation layer" to manage newer versions.** If you have to make changes to your API, an easy technique that doesn't affect your users (or your server) is to add a layer of software that transforms the code of requests and responses between the app making a call to your API and an older version of your API. This basically cuts the server out of the equation.

5. **Go versionless altogether.** This may very well be the approach you want to take, but know that it takes a great deal of planning and a "less is more" mindset, like leaving off functionality you may want to offer in order to stay versionless moving forward.

PART 6

How Is Your API Perfoming?

Setting Key Metrics

No matter what your role in your company, if you've worked to get an API up and running, you'll eventually want to provide quantifiable success of your API project.

If you're planning to track performance and usage metrics to demonstrate the success of your API, know what metrics you need and how you're going to measure them. This will be important when pitching your API to your executive team—and down the line after your API has launched.

Here are a few ways to approach the metrics.

Sample metrics for a private API...

- How much has it reduced the amount of resources you previously required?
- How much is it making things easier for you and your partners?
- How much did it streamline your IT infrastructure?
- How is life easier for your developers? What productivity gains have been made?

Sample metrics for a public API...

- How many new users are you getting?
- How many new interactions are you seeing through your API? Through which channels?
- What are the request and response metrics?

The Future of APIs

Over the last five years, there's been a broadening of interest in enterprise-oriented technologies like SaaS, big data, microservices, and AI. APIs are the nexus of all four of those areas. As more and more tasks, interfaces, and customer service portals are automated, APIs will provide the speed and efficiency required to support these technologies.

> APIs have been the change agent for the most successful pivots in tech. From Amazon to Netflix to Facebook, businesses that build in an API-centric way find it easier to reach their next customers ahead of their competition.

Delyn Simons,
Vice President, Developer Platform at Ionic Security

Drive Business Forward

It's a common misconception that APIs are an IT concern. But now you can see that's a myth. APIs are a business-driven technology that should be strategized and implemented by marketing *and* IT.

What's more, you don't need to be a tech company, government agency, or internet giant like Google to develop APIs. With both public and private applications, nearly any company can create an API and benefit from it by:

- Streamlining operational efficiencies
- Increasing customer loyalty through better user experiences
- Creating new revenue streams
- And much, much more

Getting Help to Bring It to Life

Once your organization decides to build an API, the next step is to pull together the right team to get it done. From developers who specialize in APIs to project managers, you can find the experts you need to make your project a reality on Upwork, the world's largest freelancing website.

Working with freelance professionals not only provides the exact, specialized talent you need at the right phase of your project, their experience can also help you get it done more smoothly. By including freelancers on your project team, you also have the opportunity to work with the best talent located anywhere in the world, which may reduce costs and shorten delivery times.

However you get started, what's most important is that you do. As mobile traffic continues eclipsing desktop traffic, providing well-strategized APIs may become a standard way of remaining competitive in today's business environment.

About Author

Carey Wodehouse

Carey Wodehouse is a freelance content marketer and writer based in Richmond, VA who's written for clients ranging from online retailers and global market research firms to financial corporations. As an IT/web development content writer, she's dedicated to making the complex world of web development a little easier to navigate.

Editor: **Krista Bruun**
Design: **Stanislav Krutylin**
Visual concepts: **Mina Reimer**