7 Steps to building your API blueprint

MuleSoft®

Table of contents

Introduction

APIs are the key to creating new digital channels. They facilitate internal and external data sharing, enable mobile applications, and create new monetization opportunities.

Expedia

Expedia does $2 billion annually through their affiliate network APIs. $7 billion in eBay transactions is processed through APIs.

Many enterprises recognize the opportunities of APIs, determine that they need one, then work to quickly deliver one to market. However, without a well thought-out strategy and clearly defined roadmap, most of these APIs are brought to market with critical issues that prevent them from delivering expected business results. API programs like this are analogous to building a house by telling a contractor “I need to a roof over my head.” That’s not really what makes up a complete house. Proper construction begins with planning before building: What are the zoning laws? How will the house be used? How many people will be living in it? Only then can you develop the blueprint for construction. A house isn’t simply a roof; it’s electricity, plumbing, heating, walls, rooms, etc. and that needs to be mapped out. This design phase ensures the actual construction is planned out; you don’t want to build ¾ of a house to realize you want it to serve another function. Once that design is confirmed, you can build. You start with the foundation, and work your way out to the fine details. And, as any homeowner will tell you, the work doesn’t stop once it’s built. The only way to ensure a strong and healthy home is through maintenance - monitoring to make sure everything is in working order, and repairing where it’s not. Creating an API program is no different. Just as you wouldn’t build a house without a blueprint, you shouldn’t build an API without one.

Step 1

Define your business objectives

Before developing an API program every enterprise needs to answer the same question:

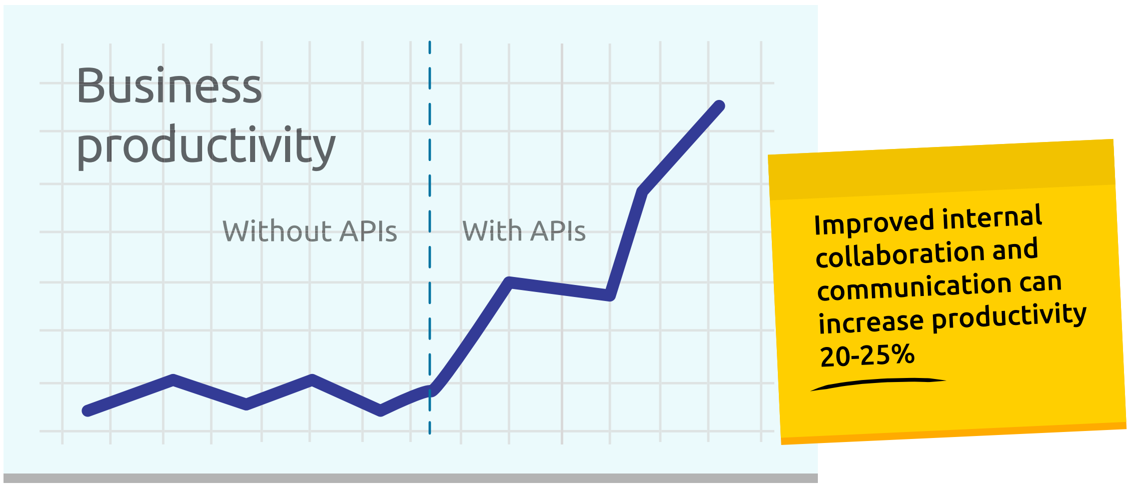
Why do I need an API?

You need to be very clear about why it makes sense to get into the API space. At this stage you are looking at your business holistically, not just through the lens of an API. How can you increase customer engagement with your company? How can you better leverage partners to enhance your value proposition to customers? What internal processes would benefit from easier data access across the enterprise? You want to develop a clear understanding of the business needs within your enterprise that could be addressed by APIs. Once you understand the landscape of opportunity, consider the potential business impact of addressing each of these opportunities, perform a cost benefit analysis, and choose the most logical starting point for your API program. You wouldn't open a brick-and-mortar store without performing an in-depth analysis of multiple opportunities and similarly, you shouldn't embark on an API program without this analysis.

Once you have identified the strategy for your API, it's time to define your business objectives for the API or the "why" behind your API program. Think through the tangible results you expect to drive by launching the API which might include new revenue sources or channels, new market penetration, internal efficiencies, content acquisition, extending capability through partners, and new product or capability introduction. Make sure the objectives you define are measurable as the ability to manage progress and improve your API over time are critical to its long-term success. The objectives you define will depend on the use cases you decide to pursue. Here are some of the most common business objectives that drive API programs.

Internal data access and agility

Although it receives far less press than open APIs, internal use is the most prevalent use case for APIs. There are multiple drivers behind the growth of internal APIs but all address a common issue: unlocking the value of enterprise data. Most enterprises lock data inside systems, teams, and departments, creating a series of isolated silos. APIs help to unlock the value of this data by creating a series of defined interfaces for accessing it.



A 2012 McKinsey study showed that improved internal collaboration and communication can increase productivity 20-25%[1]. Internal APIs facilitate cross-team and cross-organization collaboration by giving the entire organization a single, consistent way to access enterprise data and services. This eliminates the need for business units or project teams to request custom integrations or data feeds for each new initiative they launch. With internal APIs, data and services are available on demand for them to simply tap into, reducing time to market for new projects and enhancing business agility.

[1] http://www.mckinsey.com/insights/high_tech_telecoms_internet/the_social_economy

Not only do internal APIs yield significant productivity gains but they also lay the foundation for partner or open API programs in the future. Amazon launched an internal API program that required every business service or data source to expose a well-defined interface and every team or department who wanted to use that data service had to do so by tapping into the API. This initiative delivered tremendous value to Amazon and is one of the reasons they have been able to innovate as quickly as they have. But the program delivered a second valuable benefit – it served as the groundwork for their open API program. Not only did they have a series of tried and tested APIs in place, they also had a depth of learning about API design, security, management and monitoring that accelerated the process of eventually going to market with an open API.

B2B / Partner connectivity

International Data Corporation (IDC) predicts that by 2020, 40% of the IT industry's revenue, and 98% of its growth, will be driven by 3rd parties. [2]

Apps, cloud storage, and service integration mean that APIs are part of the B2B landscape. To remove cost from the supply chain, to accelerate product and service delivery, and to enhance their value proposition to customers, companies are increasingly opening up APIs to a community of trusted partners. These APIs can be used to exchange information with partners in real-time, which expedites order processing and delivery for customers, for example. Or partner APIs can be used to allow partners to build extensions on top of your services, delivering enhanced value to customers.

One thing to keep in mind with partner APIs is that while they're not completely open, they still require the same security as open APIs. With the increase in 3rd party exposure, either with partner or public APIs, security is a major concern for the enterprise. Exposing certain data or functionality can open enterprises up to abuse of the API. Just because an API is private and available only to a limited set of trusted partners does not mean it is protected from hacking or abuse. Businesses planning to offer partner APIs should take steps to secure them as though they will be made public. API security will be addressed in detail later in this paper.

[2] http://www.idc.com/research/Predictions13/downloadable/238044.pdf

Facebook is a free social application that over 9 million apps are built on top of [3]. For Facebook, these apps are valuable because they drive increased user engagement; for the app providers, Facebook provides a sizeable base of potential customers to tap into. The Facebook API enables this symbiotic relationship that drives revenue for app developers and deeper user engagement for Facebook.

Mobile initiatives

According to a January 2014 study from Gartner, by 2017, mobile apps will be downloaded over 268 billion times, generating over $77 billion in revenue [4]. Mobile users provide personal data through apps, and enterprises can process this data to understand their users and offer other services of interest. And many, if not all, of those apps are powered by APIs. In addition, mobile apps engage customers and make it easy for them to do business with you.

Monetizing Digital Assets

It's hard to find a business publication these days that doesn't include an article on how companies are leveraging data to gain insights into their business, and better understand and serve customers. But in the digital age, the value of data goes beyond its internal value. Data has become a product that is delivered through APIs.

65% of enterprise executives believe digital assets will increase their business income over the next 3 years [5].

For example, banks have vast stores of data about the purchase history of their customers. If they can aggregate and anonymize this data to protect their customers, they could sell it to retailers looking for better insight into their buyers. That bank could simply expose the data through an API and create a new revenue stream by charging for its use. Monetizing digital assets is a way to deliver concrete bottom-line value to the enterprise with assets you already own.

[3] http://www.insidefacebook.com/2012/04/27/facebook-platform-supports-more-than-42-million-pages-and-9-million-apps/

[4] http://www.gartner.com/newsroom/id/2654115

[5] http://www.mckinsey.com/insights/business_technology/bullish_on_digital_mckinsey_global_survey_results

Step 2

Define your business model

With your API objectives now defined, it's time to define your API business model. The business model determines how developers will engage with your API, and if and how you will earn money from it.

Some business will make money from an API simply by charging for its use while others will make it free to use the API or even pay developers to use it because they plan to monetize it indirectly. This indirect monetization comes in a few different forms. Maybe the API provider sees developers as an extension of their business, bringing their product or service to new customers and markets through the apps they build. Or maybe they believe that developers who build on top of their API will enhance the value delivered to end customers of that product or service, making the product stickier and paving the groundwork for upsell and cross-sell opportunities. All of these business models are valid and yield significant benefit when applied to the right use case. This section will map out the 4 main API business models, their pros and cons, and factors to consider when choosing which model or models to use. The 4 main API business models are: developer pays, developer gets paid, indirect, and free.

Developer Pays

In this model the developers who use your API pay for the service you offer. There are a few different flavors of the developer pays model including pay as you go, tiered, and freemium. Pay as you go allows developers to consume an API, and pay only for the calls they use. This is advantageous as you don't put up any barriers to usage and you greatly reduce risk for the developer - you make money from the API when the developer makes money through his application. If developers are unsure of how many transactions they will need to support, they can still easily begin using the API.

With a tiered model, developers pay for API calls by tier - the more calls they use, the less they pay for each call. The tiered model is geared towards heavy users, and will be enticing to developers who know they will be making a lot of calls and want to get a price break.

With the freemium model, developers pay nothing for use of the API but their use is limited or restricted. If they want value added services, access to gated data, or increased calls, they must upgrade to a paid subscription. This is a powerful tool as it allows developers to work with your API at zero risk/cost, with the ability to upgrade later on. The Google Maps API is a good example of this model. Developers can use the API free of charge until their calls per day surpass a defined threshold. At that point they must pay a fee for all additional calls.

Developer Gets Paid

With the developer get paid model, API providers incentivise developers to build applications on top of their API by sharing with them a portion of revenue earned. This model makes sense for companies who can successfully grow their customer base and revenues through referrals.

amazon.com

Amazon.com has an Affiliate program where developers who use their advertising API receive a fixed percentage of the purchase price for goods sold upon a referral from the developer's app or site to Amazon.

Similarly, Rdio, a digital music service, has a Recurring Revenue Share program to reward developers who use their API. Every time a customer signs up for the Rdio service after being referred from the developer's app or site, that developer receives a percentage of the monthly revenue generated by that customer for as long as they remain a customer.

Indirect

With indirect monetization, APIs are the middleman to revenue generation. The API provider doesn't charge for use of the API but use of that API indirectly drives increased revenues.

ebay

eBay does not charge developers who use their API because that API extends the footprint of eBay services to new channels, new sites and apps, and new customers, and in so doing indirectly generates revenue for eBay. Today, over $7 billion is transacted annually through the eBay API.

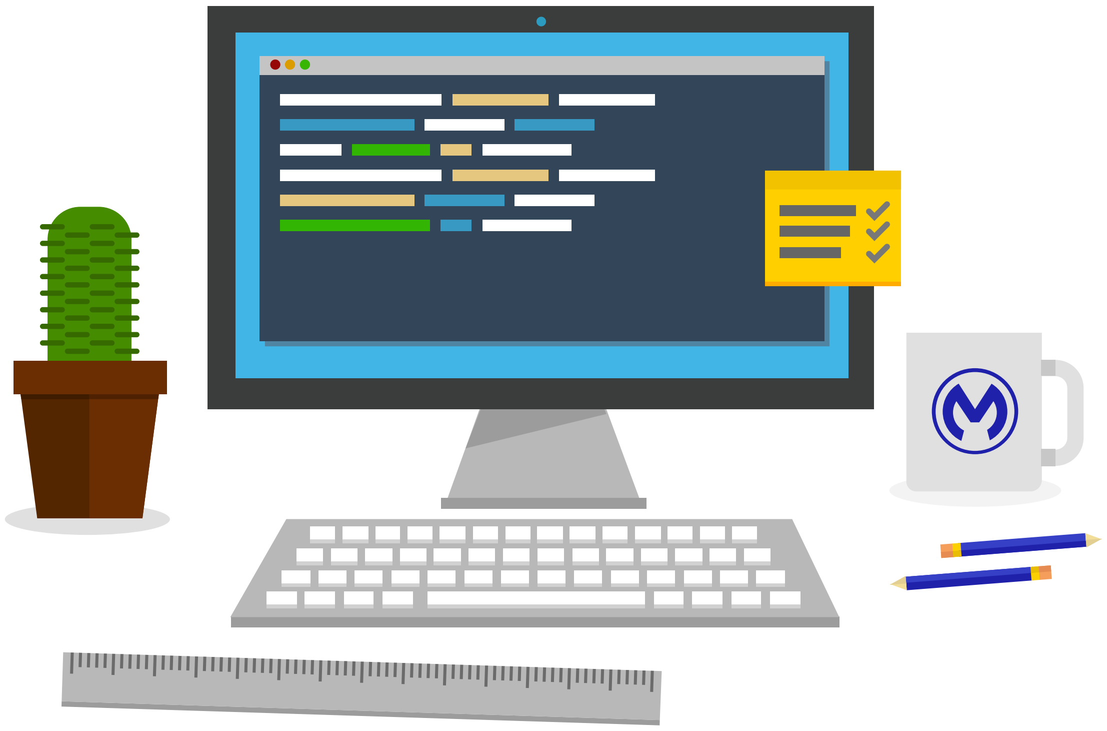
Free

The free API business model allows 3rd party developers to work with an API at no cost. There are still indirect monetization opportunities with this model as it often increases the “stickiness” of a company’s product or service through value-added capabilities delivered through 3rd party apps that leverage the API.

Twitter do not charge for use of their API, applications like TweetDeck, developed by 3rd party developers, make it easy for enterprises to work with and get useful data using the Twitter API, which subsequently drives increased usage of Twitter. The applications built on Twitter that leverage the Twitter API have been a key driver behind Twitter growing to see over 15 million calls per day.

Another common flavor of the free business model is to monetize the API through advertising.

Facebook’s API is free for developers to use. However, as those developers deliver apps targeted at the Facebook user base, they advertise the new app on Facebook to drive sales and adoption of their own product. Although the Facebook API is free to use, it indirectly generates advertising revenue from the developers who want to sell their apps into Facebook’s base of 1.3 billion active monthly users.



Step 3

Design your API

After deciding what kind of API to deliver and mapping out the objectives and business model for it, it’s time to begin design. This is where publishers begin to write their API so that it’s easy for developers (consumers of the API) to read, test and use.

In design, the most important question to ask is “who am I designing this for?” With APIs, it’s developers so delivering on their needs and expectations is critical to the success of your API. The best way to understand what developers want from your API and how they might use it is to get their input and feedback during the design phase. It’s important to not be thinking about the implementation of the API at this stage but rather to focus solely on designing a great API user experience. Getting design right before building the API will save the time and cost of identifying issues late in the development process that require reworking both the design and the implementation. To enable developer feedback and input during the design phase, providers will need a mocking service that allows developers to interact with the API even before its backend has been implemented.

APIs are designed using an API specification. There are many options out there including WADL, WSDL, Swagger, and RAML. Good API specs are human-readable and easy to use. While design is critical, that doesn't mean publishers need to invest hundreds of man hours in delivering it. Specs like RAML make it easy to write and reuse patterns and structure, so publishers are not repeating tasks over and over. This reuse also facilitates standardization across APIs, which becomes important as companies build out their API programs over time.

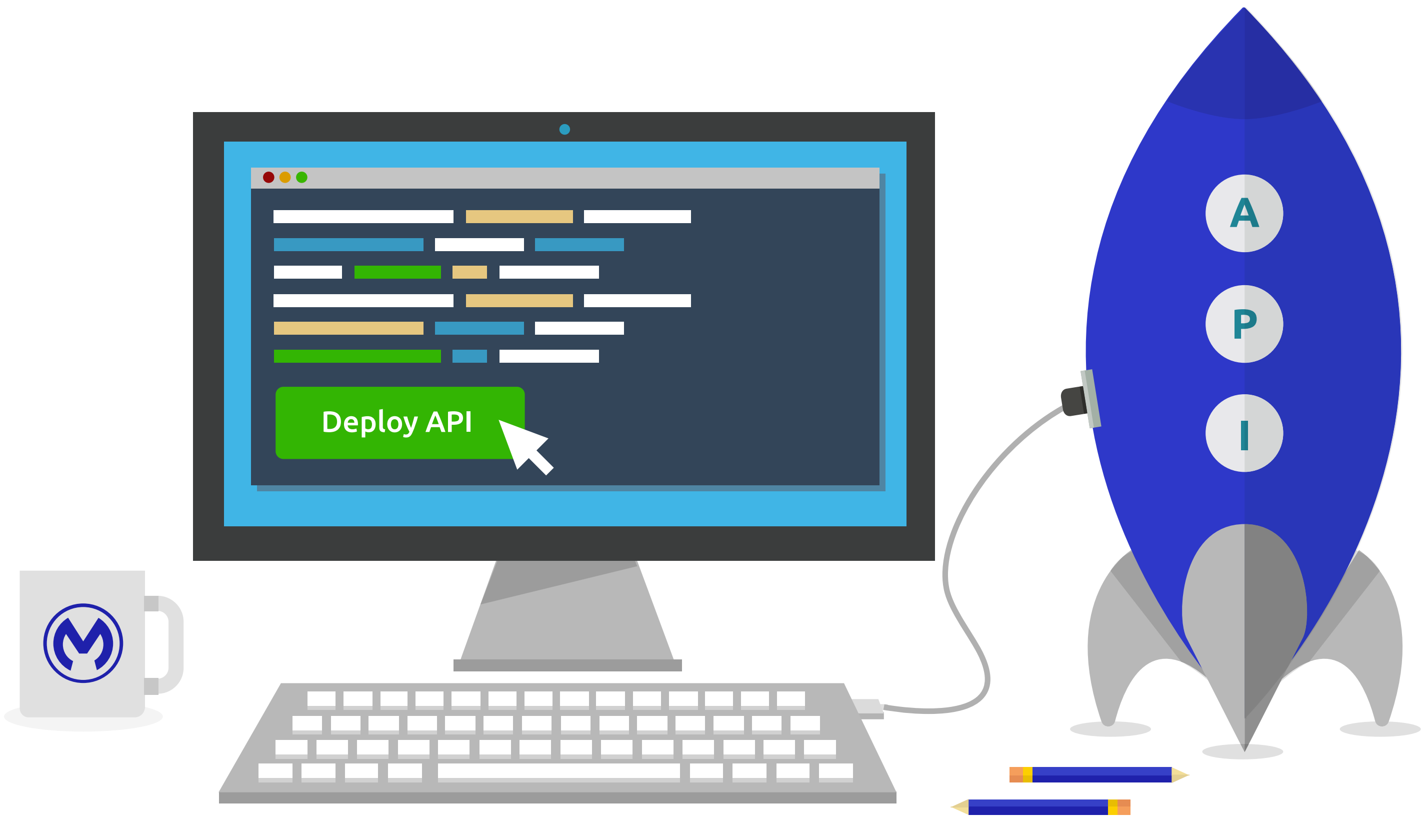
Getting the design of your API right is important. Once you release your API and developers begin to use it, any changes you make to the design of your API will impact those consumers and potentially, bring down the apps they've built on top of your API. Developing an API is similar to developing a website, except that websites can be designed quickly and changed often. APIs can't. If a person encounters an error on a website, he or she can see that it isn't working and look for a way around it. Application code isn't smart enough to work around an unexpected API response. Even minor changes to an API interface will likely break an application that uses it. Therefore, iterate on your design, shop it around with a number of developers, and only move to the next lifecycle stage when you're comfortable that you've nailed down the right design.

Step 4

Build and deploy your API

Once designed, the API needs to be built by connecting to the backend services or applications that will power it. If the API will be connecting to existing web services, this connection will be a simple proxy and should be fast and easy to configure. However, if data needs to be orchestrated across multiple systems on the backend, transformed to a new format, or if the backend system is legacy or otherwise difficult to connect to, building the API is more complex. In this case, integration and orchestration capabilities are required. Be careful not to write custom code in the API layer to perform these tasks as that is a short-sighted solution that will make your API brittle and difficult to manage and maintain over time. Leverage a fit-for-purpose integration solution here or choose a gateway that delivers these capabilities natively.

The second major component of building your API is to build out the governance to manage security and access.



API Policy Management

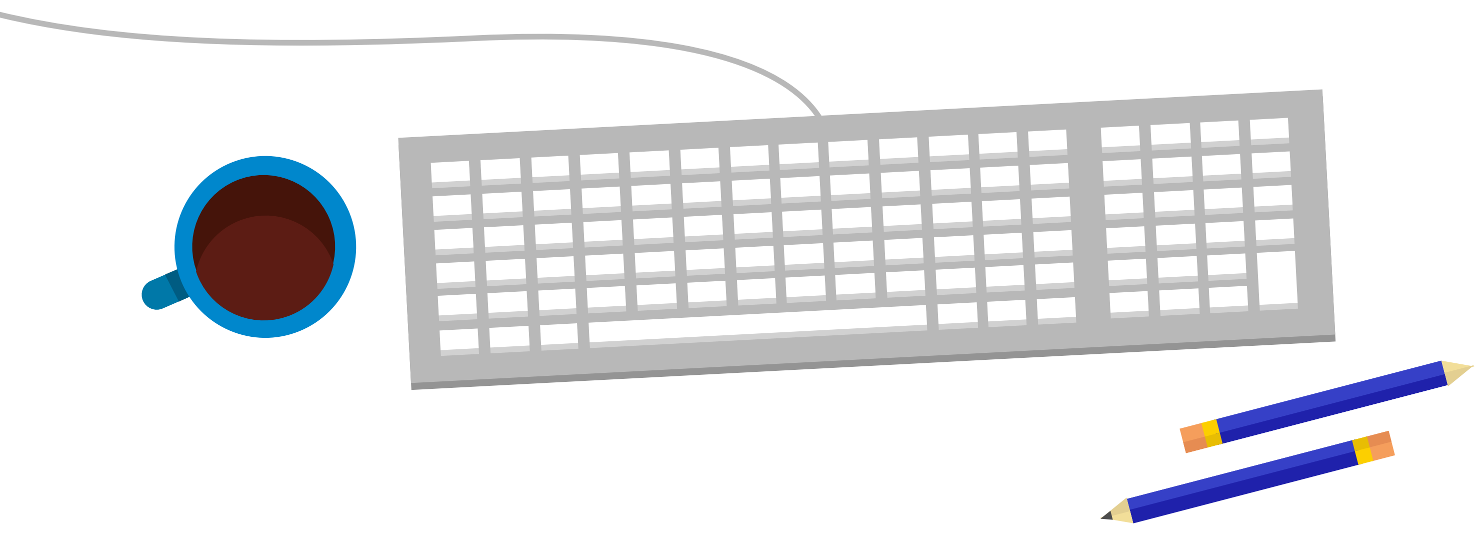
Before an API is deployed, policies that govern how it can be used must be established. These policies will dictate who can access the API, how users will be authenticated and authorized, and how much traffic they can consume. Implementing policies to control access is critical to keeping your APIs and the underlying services they leverage protected from unauthorized access. Policies that limit consumption are important to keeping the API performing at peak levels - without these policies, unanticipated spikes in API traffic could overwhelm the API and cause it to go down, bringing down all of applications consuming the API as well. Most solutions provide a number of pre-built policies out of the box in a policy library to manage common tasks like rate limiting, throttling, and security enforcement. In addition, they will allow for custom policies to be created.

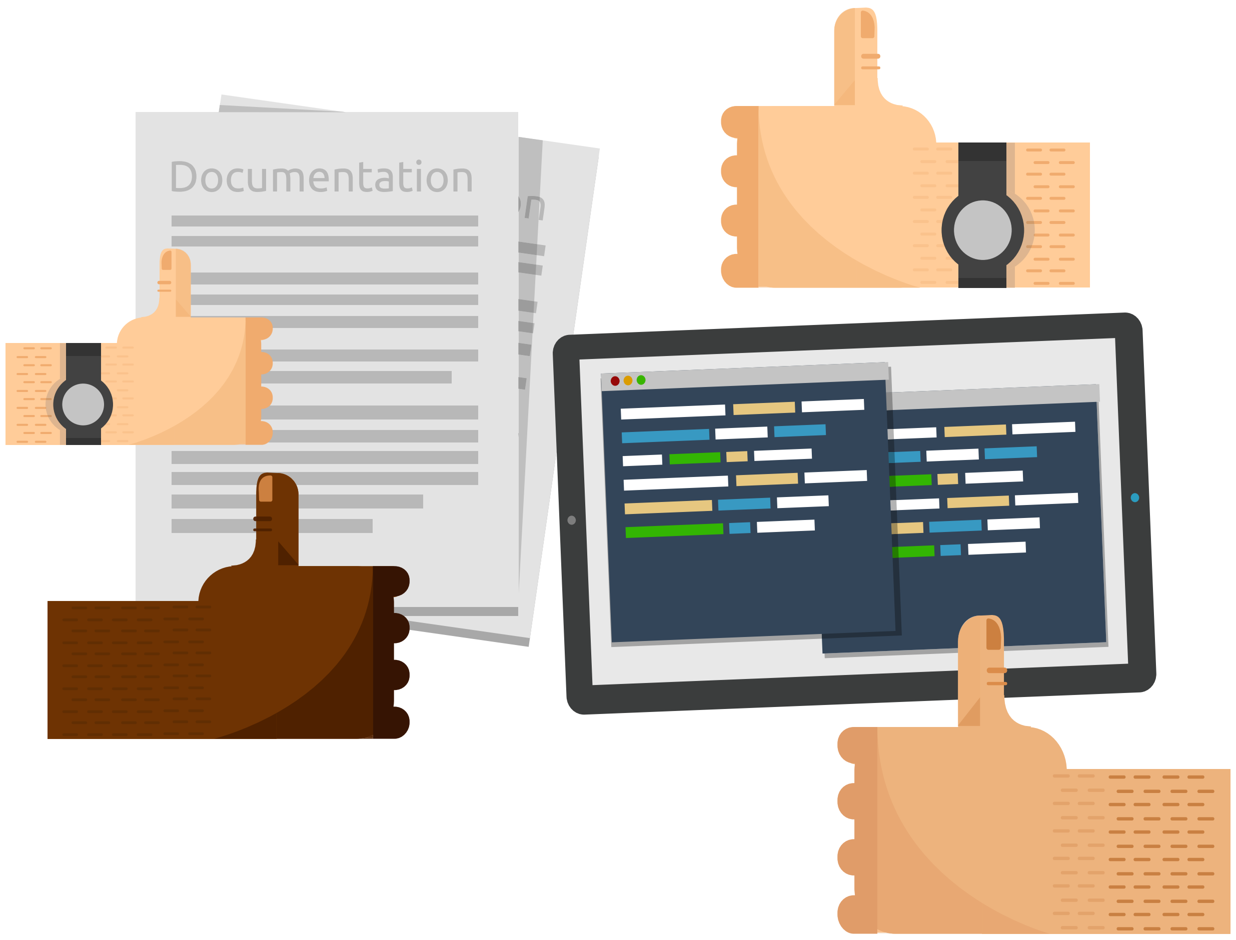
API Security

Without the right security controls, your API and the data it exposes are vulnerable to unauthorized access. API security is essentially composed of 2 core tasks - authenticating the user to verify that they do have access to the API, controlling what that user has access to through authorization. It's important to decide what mechanisms you will use to secure your API before exposing it to users.

Authentication is the process of determining whether someone or something is actually who or what it declares itself to be. This is handled by asking that user for an ID and password and validating those against an identity store. This validation can be performed in a few different ways. For internal users, an enterprise LDAP server is typically integrated with the API management solution to verify credentials before granting access. For external APIs, many API providers require a token to access the API. The benefit of tokens is that they protect the user from having to share their credentials with the API. Instead, a token is generated based on those credentials and passed to the API to authenticate the user. OAuth is rapidly becoming the standard method for token-based authentication of APIs.

The second step in securing your API is establishing what data and functionality a user will have access to based on who they are and the kind of relationship they have with you. For example, a customer should only be able to access details about their own account, and a freemium user might only have access to basic functionality. This authorization is performed by policies applied to an API. Authorization policies allow API publishers to establish multiple access tiers and grant specific levels of permissions to different user groups.





Step 5

Engage developers with your API

Driving adoption of your API by engaging developers with it is the perhaps the most critical success factor of your API program. Engaging developers with your API requires that you build and publish a portal where developers can find your API, learn how it works and begin using it. Surprisingly, research from over 11,000 APIs listed on ProgrammableWeb shows that 99% of those APIs are missing one or more of the components required to successfully engage developers to use your API.

As discussed in step 3, a key element of API design is getting feedback from developers - your API can only succeed if developers like it and want to use it. During the design phase, publishing your draft API to the developer portal is an effective way of engaging developers for early feedback. Once your API goes to production, it's published to the developer portal for discovery and use.

The best developer portals include all of these elements:

- **Getting started:** This page provides the essentials that allow a developer to rapidly begin working with an API. This includes how to register for API keys or tokens and authorization is managed. The Getting Started or Quick Start page is likely to be the first thing developers will look for and click on, so it should be easy to find on the portal.

- **Documentation:** The Documentation page is where the developer can get into the meat of your API and learn about its functionality, supported verbs data formats, etc.

- **Code samples:** As part of your documentation, code samples provide users with an fast and easy way to see how the API works.

- **Interactive documentation:** Making real calls against the API gives developers a chance to test drive the API. Seeing exactly how requests work goes a long way to understanding the documentation, and simplifying what is typically a huge pain point for developers.

- **Error codes and responses:** When developers experience errors, the first place they are likely to look for answers is your dev portal. Having a section dedicated to error codes and responses greatly increases the usability of your documentation.

- **Forums:** Developers like talking to fellow developers, getting input on how to use an API, learning tips and tricks other developers may have discovered, etc. However this page will only increase engagement if it is monitored properly. A forum that is never checked, where questions go unanswered, is a red flag for developers.

- **FAQs:** Ideally the layout of your documentation leaves no question unanswered. However, an FAQs page gives developers an overview of key information in one place, rather than requiring them to click around making it a worthwhile investment.

- **API status:** An API status page provides uptime and response time, and shows the consistency and reliability of your API.

Once you have designed and built a great API and published it to a complete, easy-to-use developer portal, the next step is to drive adoption of your API. Developers can't start using your API until they know about it so how will you get the word out? Getting the word out about your API and making it discoverable by the developer community is important for all APIs, whether they're external and open to use by 3rd party developers or internal and only available to developers inside your organization. However, the methods you should use to attract developers does vary by audience type. To attract 3rd party developers to your open API, there are 2 proven methods. First, developer evangelists can speak to developers in their own language, earn their trust, and engage in productive conversations with them. Another way to get in front of 3rd party developers is by organizing hackathons. Hackathons are a great way to build a community for your API by uniting a group of developers to meet one another, collaborate on projects and learn new technology. The popularity of hackathons has skyrocketed over the past few years, with attendance and prizes growing exponentially.

Last year Salesforce held a hackathon where the winner received $1 million cash. That draws a lot of attention, and a lot of developers to your product.

Driving adoption for internal APIs is also critical as the projected benefits of internal API programs can only be achieved when a critical mass of internal developers begin to use published APIs. The first step in engaging these developers with your API is simply advertising its availability and promoting the developer portal you created. But there are additional methods to drive an audience for your API. For example, consider making your API the only way that internal developers can access enterprise data or services. This is the method that Jeff Bezos employed with great benefit at Amazon. Bezos mandated that every enterprise service be developed with a well-constructed, consistent API, and that every application leveraging that data source or service must go through the API – no back doors were available. While this may be too strict for many companies, the underlying principle can be leveraged. APIs should be the fastest, easiest way for developers to deliver capability to the business and they should be easy to access and use.

Step 6

Manage and monitor your API

When an API moves to runtime deployment, managing and monitoring are critical so that issues can be rapidly identified and usage and performance can be tracked. The API gateway allows the API runtime to connect to backend services and applications and also serves as an abstraction layer so that the API can be managed. The gateway is where developers connect to the API. Using caching and routing features of the gateway, publishers can ensure high availability and fast response times.

Once the gateway is configured, the API will need to be monitored at both operational and business levels. This functionality is typically delivered via reports and dashboards in an API Analytics solution. Metrics like API status, response time, and uptime are essentially the vital signs of the API and serve as the foundation of operational monitoring. In addition to these core metrics, API providers will want to monitor how well individual APIs and the entire API program are performing at a business level. This data serves two important functions. First, it helps API owners and business sponsors track the progress of their API program and calculate the ROI of APIs. Secondly, it gives them deep visibility into the consumers of their APIs so that they can optimize for the right users and use cases to drive increased usage the the future. Some important business metrics to monitor include API consumption by country/region, API consumption by platform, and traffic by API consumer.

Step 7

Measure impact

Your API is a living, breathing part of your business; and should be continously improved uopn. Once it is designed, deployed, and being used by developers, publishers should plan a time to review the success of the API program. This is not a technical review of the API, but rather, this is a structured process of reviewing the objectives and goals for the API and determining how well the API is measuring up. Are you seeing the adoption you were expecting/ hoping for? Is that adoption driving the amount of revenue you expected? If not, it might be time to revisit what you're delivering in the API, how it's designed, how you're promoting it to developers, or your business model.

Your API was designed and created to serve a purpose within the enterprise. However, goals and needs change. You need to continually review your API to make sure it is still serving the business.

As with building a home, you build it with one function in mind, but you may need to remodel it as your family grows or you may need to move entirely. This is the same with your API. Your business objectives may change, and you may need to adjust your API slightly. Or you may need to move to a new version. Regularly re-evaluating your objectives, and how your API is serving those objectives, is critical to the effectiveness of your API.

Conclusion

An API program is essential for the modern enterprise. APIs can help companies capture new customers, create new revenue channels, more effectively leverage partners and better serve customers. However, without careful planning from the outset, the potential of APIs will go unrealized. Delivering a successful API program is like building a great home – it requires the right blueprint. And that blueprint must consider all stages of the API development lifecycle from planning through design, implementation, management, and developer engagement. In addition, because your API is a living extension of your business, it must be easy to adjust in response to market or business changes. The best way to ensure the success of your API program is by starting with the right tools. The Anypoint Platform for APIs is the only API management solution that supports APIs throughout their lifecycle, with design tools that make it easy to deliver APIs developers love, and the architecture to ensure agility over the long term.

To learn more about the Anypoint Platform for APIs visit:
www.mulesoft.com/platform/api.

MuleSoft®