WHITEPAPER

Secrets of a Great API

Core principles for delivering successful APIs

Secrets of a Great API

APIs are not new. They've served as interfaces that enable applications to communicate with each other for decades. But the role of APIs has changed dramatically in the last few years. Innovative companies have discovered that APIs can be used as an interface to the business, allowing them to monetize digital assets, extend their value proposition with partner-delivered capabilities, and connect to customers across channels and devices.

When you create an API, you are allowing others within or outside of your organization to make use of your service or product to create new applications, attract customers, or expand their business. Internal APIs enhance the productivity of development teams by maximizing reusability and enforcing consistency in new applications. Public APIs can add value to your business by allowing 3rd party developers to enhance your services or bring their customers to you. As developers find new applications for your services and data, a network effect occurs, delivering significant bottom-line business impact. For example, Expedia opened up their travel booking services to partners through an API to launch the Expedia Affiliate Network, building a new revenue stream that now contributes $2B in annual revenue. Salesforce released APIs to enable partners to extend the capabilities of their platform and now generates half of their annual revenue through those APIs.

The pitfalls of a mediocre API

Organizations often decide to build an API without fully considering key success factors or without first engaging their stakeholders. In either case, the risk is that the API does not fit the needs of its users. And APIs that don't fit the needs of users have a high cost: limited adoption by developers and ultimately, a failure to meet business objectives. Once the API is designed and built, undoing these mistakes is difficult and time-consuming. In most cases, you must start over again, redesigning a new API, implementing it by connecting to backend services, then rolling it out again to the developer community. Worst of all, you will have to transition all existing users to the new API. This will require additional work which they may not have the time or willingness to do. At that point, you'll be faced with a tough choice - continue to support the original API and its users until they eventually (hopefully) migrate, or shut it off and potentially alienate and lose those users.

Another common pitfall of API programs is allowing the design of your API to be dictated by the constraints of internal systems or processes. This is never a good idea, but is particularly perilous when the backend functionality lives in legacy systems whose data schemas are overly complex or whose business logic has been extended over the years using hard-coded workarounds and convoluted logic. Exposing this kind of dirty laundry to your API consumers is a recipe for failure. APIs modeled after internal systems are difficult to understand and use and developers simply won't invest the time it takes to become productive with them. What you need is an API that is simple to understand and easy to use. Developers should be able to assess the functionality of your API and start using it in just a few minutes. The only way to deliver that kind of ease of use is to design for it upfront.

The Value of a Great API

A successful API is more than a feature; when you view your API as a product, it can be an enabler of your business strategy. Part of the magic of APIs is that creative developers find uses that the API designers never envisioned. If the API is well designed and easy to use, this can be an enormous benefit and opportunity, turning your service into a platform that can grow in many ways.

A great API encourages developers to use it and share it with others, creating a virtuous cycle where each additional successful implementation leads to more engagement and more contributions from developers who add value to your service. A great API can help you grow an ecosystem of employees, customers, partners who can use and help you continue to evolve your API in ways that are mutually beneficial.

But the promise of APIs can only be realized when target consumers begin to use them. For internal developers, APIs introduce a new way of working and one that will require some buy-in. In addition, internal developers won't use your API if they don't believe it's the best, most efficient way to achieve their goals. Well designed APIs that are easy to use will encourage adoption by internal developers, paving the way to a better-defined, more consistent and maintainable approach to development. For public APIs, the situation is even more competitive. An ever-increasing pool of APIs is competing for developer's attention, making the design and ease of use of your API critical to its adoption and ultimately, its success.

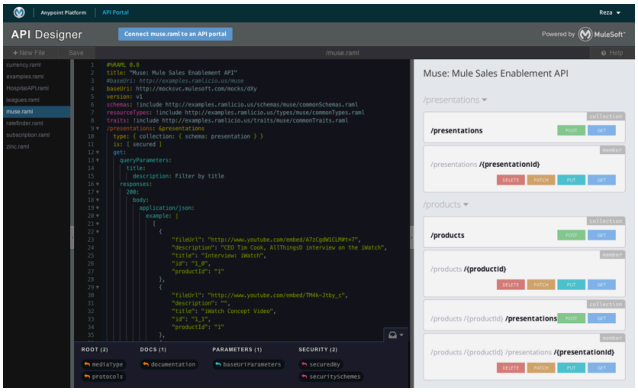
Unfortunately, too many API providers build their APIs before thinking through the critical success factors, resulting in APIs that fail to meet business objectives. Delivering a great API isn't hard if you follow a few proven principles. In this paper we'll demystify API strategy by reviewing the 4 secrets of a great API.

Secret #1: Design for great user experience

To deliver great APIs, design must be a first-order concern. Much like optimizing for UX (User Experience) has become a primary concern in UI development, optimizing for APX (API User Experience) should be a primary concern in API development. An optimal API design enables applications developers to easily understand the purpose and functionality of the API so that they can quickly become productive using it. It also allows organizations to focus on getting API design right before investing in back-end implementation, which is time consuming and expensive to undo if design issues aren't identified until after implementation.

The best way to design an API that developers want to use is to iteratively define the structure of the API in an expressive manner

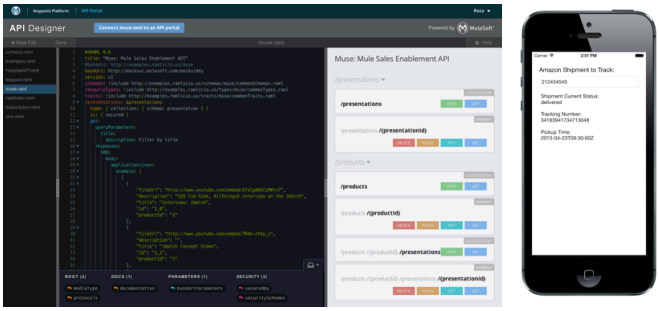
and get feedback from developers on its usability and functionality along the way. The API Designer is an example of this concept in action. The API Designer is an open source design environment that leverages RAML, the RESTful API Modeling Language. The API Designer provides an editor for drafting the APIs structure while rendering in real time an interactive console to enable interaction with the API. As the API is designed, application developers can interact with it and test its behavior, thanks to an integrated mocking service that returns the values a call to the live API would produce. Because APIs designed in RAML are concise and easy to understand, application developers can rapidly assess the APIs functionality and usability and offer concrete feedback on ways to improve it.



Secret #2: Optimize for use case

There is no such thing as a one-size-fits-all API. Even for the same underlying service or set of services, multiple APIs might be required to support different types of users and use cases. An API should be optimized to fulfill a specific business request in a specific context. Too often APIs are modeled after the design of the backend services or applications they expose instead of the use case they fulfill. This results in poor performance of the client application, poor user experience, and ultimately, poor adoption.

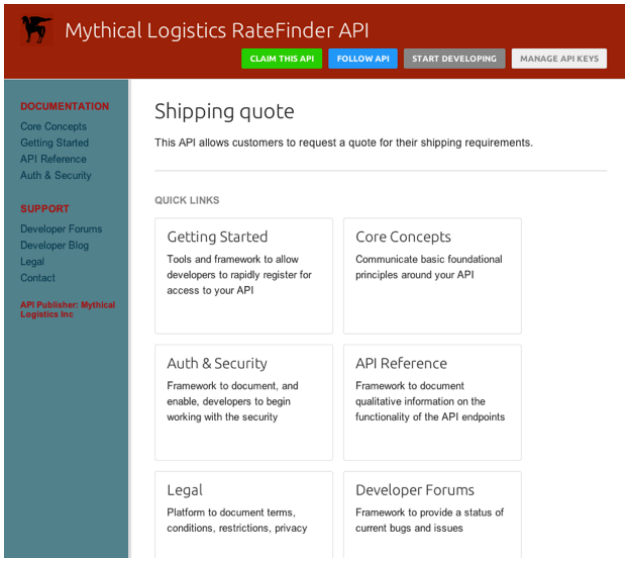
To optimize your API for a specific use case, think about how coarse or fine-grained it should be. For example, if you're designing an API to enable access to sales order status from a mobile device, you need to consider the constraints of that use case. A mobile application has a higher sensitivity to number of network trips, latency and data size than a web application so this API should be designed to limit backend calls and minimize the size of data returned. In addition, this use case is fairly granular – the API will lookup an order based on order number and return a status. Therefore, the API should expose this specific fine-grained functionality so it can be invoked independently. If the underlying service it accesses is coarse-grained and you anticipate building additional APIs on that service to address additional use cases, consider a tiered approach. Expose fine-grained services that users can access directly, and add coarse-grained services on top of them to support broader use cases. Users can choose to call the fine-grained APIs directly or if they need the combined functionality of multiple fine-grained calls they can use the coarse-grained APIs. This API designed in API Designer is an example of an API optimized for this case.

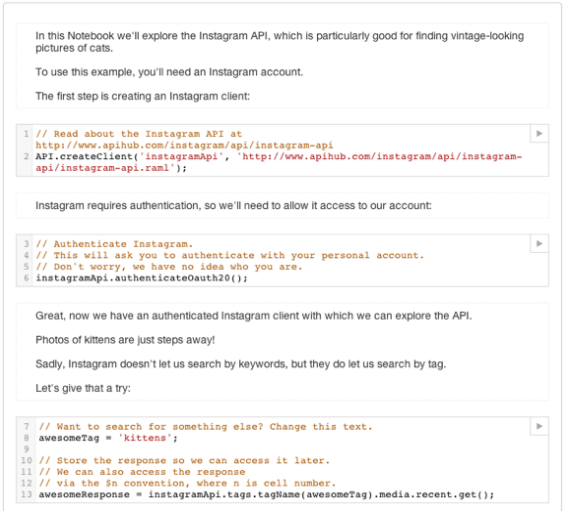


Secret #3: Provide easy access

Finding an audience for your API begins with publishing it to a portal that allows developers to discover your API so they can begin evaluating it for their use case. The developer portal should include all of the tools developers need to learn about and begin using your API. Developers reviewing your API will only invest a few minutes before deciding whether or not to continue; having information available in a clear and easy-to-consume format will encourage them to stick around rather than go elsewhere. Developers will quickly scan your documentation to get an overview of its functionality then zero in on what it will take for them to get up and running on the API. From there, they'll quickly want to see some examples and ideally, start interacting with the API. Developers are far more likely to use an API that includes interactive documentation that allows them to interact with the API over static pages that only allow them to read about it.

The API Portal delivered in MuleSoft's Anypoint Platform for APIs is a good example of the value-added features that make it easy for application developers to engage with and start using an API. The API Portal includes interactive documentation that not only describes the endpoint but also the fields required to call that API and the data that is returned. In addition, you can add code samples to give developers a head start in building the code to access your API in the applications they build. Finally, the Console includes "try it" functionality that allows developers to interact with and test the API. During the design phase before the API has been implemented, a mocking service allows developers to test the API's behavior and see the resulting body a call to that API would produce. Once the API is implemented, developers can test the live API.





Secret #4: Build a community

The application developers who consume your API are not just your customers; they are the ecosystem that will drive the success of your API. Treating them as valued members of your community can drive significant mutual benefit. An obvious benefit of a thriving developer community is a wider reach for your API. To support the organic growth of your API, your developer portal should include an easy way for developers to share knowledge with each other. The Notebook feature of the API Portal demonstrates this concept in action. It allows developers to document new uses they discover for your API to grow the addressable market for your API. In addition, they can share tips and tricks in forums and even add code samples to make it easy for new developers to get started quickly with your API. Finally, a valuable benefit of community that is sometimes overlooked is that the greater the number of developers using your API, the faster bugs and issues will be identified and communicated so that you can continue to improve the value of your API.

In addition, there is great benefit in having an established communication channel with your developer community. Your API is not a static entity – as new use cases are identified and use of your API expands, enhancements and fixes are inevitable. When you release a new version of your API, you can easily communicate the enhancements in the new version through your developer portal. You can also quickly assess who's using each version of your API and communicate an upgrade path to them as you deprecate older versions. Finally, understanding your developer community and having accurate insight into use cases and patterns provide invaluable knowledge that you can use to enhance your API over time.

Summary

APIs are becoming ubiquitous as their potential to transform business is becoming widely recognized. But delivering a successful API program that achieves defined business objectives requires a systematic approach to designing and managing APIs. Great APIs aren't difficult to develop if you design for your users and the business processes the API will support, if you make it easy for developers to find and consume your API, and you actively manage your API developer community as an extension of your business.

Get started designing API's api-portal.anypoint.mulesoft.com

About MuleSoft

MuleSoft's mission is to connect the world's applications, data and devices. MuleSoft makes connecting anything easy with Anypoint Platform™, the only complete integration platform for SaaS, SOA and APIs. Thousands of organizations in 54 countries, from emerging brands to Global 500 enterprises, use MuleSoft to innovate faster and gain competitive advantage.

For more information:

- www.mulesoft.com
- info@mulesoft.com
- Twitter
- Facebook
- Google+
- LinkedIn