WHITEPAPER

How to Design and Manage APIs

Bring Your Business into the API Economy

Executive Summary

Today everything truly is connected to everything, and APIs are the connection mechanisms. Many companies are finding that their use of APIs has accelerated over the past 12 months, and that it will continue to accelerate during the next 12 months. APIs aren't really a new technology; they've actually been around for a long time. What is new is that they are in far broader usage today than they have been in the past. As of late 2015, the *ProgrammableWeb* API directory listed more than 14,000 APIs.

When thinking about designing and managing APIs, a number of important questions arise. Why are APIs so well used and what effect would they have on your business? What is the impact of the expanding API economy on both providers and consumers? What are API tools and how can an investment in tools be an extremely cost efficient foundation for production grade API use cases?

This whitepaper will cover the following:

- The definition and strategic importance of APIs
- The principles of API design
- Best practices for API management and governance
- Consideration of API management tools

The Causes of the API Explosion

An API is simply code that is pre-written to enable interoperability between diverse systems and software. In using APIs, no one has to understand how the system on the other side actually functions. APIs allow anyone to access those systems with only working knowledge of the API itself. This is in contrast to the integration technologies of the past, in which custom programs had to be created for each integration use case.

APIs offer a much faster and easier way to connect diverse systems than has been in wide use in the past. This is becoming increasingly important for companies which are, or want to become, software driven businesses. Consumer facing applications, public cloud, and mobile applications are all driving growth in API usage, as are social media and the internet. APIs offer a way to share raw data and processed information from the cloud platform, with traditional back end enterprise applications and databases. This sharing of data reflects the importance of the interoperability of systems to the software-driven business. Although the techniques behind API-led connectivity differ from those supporting traditional data integration and ESB, or enterprise service based platforms, all represent interoperability. The number one driver for interoperability is the need to interact with business partners, such as suppliers, providers, customers, or partners. APIs make interoperability faster, easier, and cheaper.

The widespread use of APIs covers nearly every industry vertical because the need for speed and agility has become a universal enterprise requirement. APIs have become a mainstay supporting high velocity integrations. Before APIs, connecting a computer in the data center with a scanner, for example, required a custom project and weeks or months of effort. Essentially a programmer wrote custom software to enable the two systems, which spoke different protocol languages, to interoperate. As recently as 7 or 8 years ago, in the early days of component based service oriented architectures, each integration was still essentially a separate product. From our perspective today, that was a very slow way of doing business and one which today's companies really can't afford to tolerate. APIs offer a more standardized approach that makes brittle organizational borders more flexible.

The rapid changes we're seeing today across virtually every industry vertical are made possible to a great extent by the API economy. Companies can extend their

The widespread use of APIs covers nearly every industry vertical because the need for speed and agility has become a universal enterprise requirement.

borders to new geographies, suppliers, devices, or customers far faster than they could in the past. And, with the coming of internet sustained and similar hyper-integrated ecosystems, it is clear that API usage will only continue to grow over time.

The Impact on API Consumers and Providers

Most companies actually start out as API consumers. That is, they access APIs provided by other organizations. For example, SaaS or a cloud infrastructure customers are familiar with the APIs available from their cloud service vendors. Some, but not all companies, are API providers as well. These companies create APIs that provide access to their applications or services. Provider-written APIs can benefit either internal or external customers. One common use case for internal customers, for example, is to develop common data access APIs for developers to use in creating new applications. Such APIs can reduce development time by inventing the wheel only once, as opposed to reinventing it for every new application.

There are a few things that consumers and providers of APIs need to be aware of. The first is basic API readiness. With no discipline or governance, API usage can easily spin out of control.

Developing policies, standards, and governance up front helps to ensure that APIs are created and utilized in the most cost efficient way possible. API management typically means the governance aspects that apply to versioning APIs, but also applying external aspects that are beyond the API's description and implementation: security; SLA tiers around the API; registration models around the API; the collection of both usage and operational matrix data which then feed back into improving the API; and the introduction of new versions.

The APIs you use should be well-implemented. This means that you need to make sure that the API is built robustly. It must be built on a reliable, scalable, and highly available foundation and it should be easily integrated with your existing back-end APIs, services and assets that you already have. All of this leads to the API-led connectivity concept, which will be discussed in further depth below.

The Ideal State for APIs

APIs are multidimensional in that they touch lines of business and IT top level management, so companies may find they already have a large number of APIs. While it's relatively easy to provide or consume APIs, the number of API connections that an organization uses tends to grow very rapidly. As more users and more applications connect, and as development creates new APIs and versions, the API landscape can start to look more like a maze to be navigated than the straightforward way to flexibly extend the business.

IT is transforming into an enabler of technology within different business units; one of its key new roles is to promote existing APIs from a hidden shadow state into a published, ideal state where there is much more reliability, availability and discoverability. Also, IT not only allows the move of existing APIs into the ideal state, but also enables the incubation of new APIs to start in the ideal state by merging existing best practices, existing characteristics and learnings within the organizations around the creation of APIs, and, importantly, reusing existing APIs instead of reinventing them.

An API in an ideal state has certain important properties. First and foremost, the API needs to be designed in such a way that the consumers of the API will actually want to interact with it. The better designed that API is, the more easily it makes the data behind that interface accessible to the consumers and therefore makes it successful more rapidly.

Consistency is another important characteristic. For example, often a particular security scheme has been implemented five different ways or sometimes many more within an enterprise. That makes the life of the API consumers very difficult, but it’s difficult for API providers as well because then they have to go and reinvent the wheel every single time around implementing a security scheme. Another example might be a business need to create an application that consumes three different APIs: each of them has the same security scheme but defined in three different ways. Three different site variations of software have to be created, and therefore interaction with that API in order to implement the application quickly becomes

unmanageable. Consistent design, enabled by IT, then, is key to make API-led connectivity work.

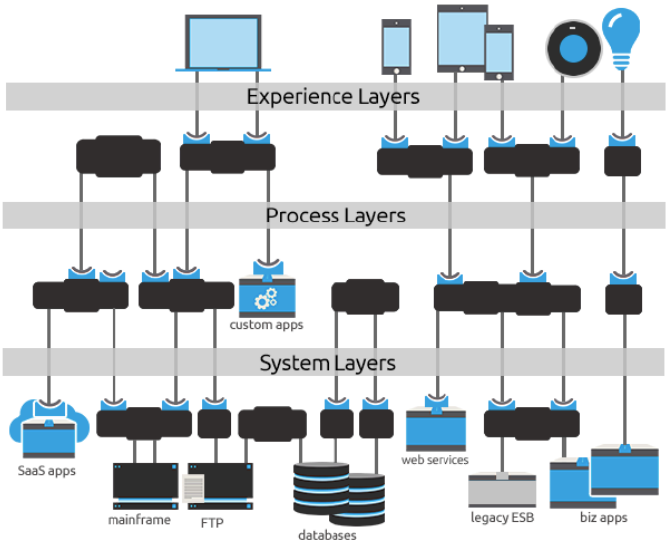
Discoverability for application developers or API consumers is also important. Developers need to be able to easily find what APIs within the organization are available for them to be successful. But in many cases, there are only various spreadsheets or various Wiki pages spread around documenting that there's an API. What companies really need is a central place where all the APIs are actually exposed and can easily be searched and used from there.

It's important to note that this should be complementary to the existing architecture. It's all about reusing existing assets and business logic. What API-led connectivity does is transform the present investment in order to make it more agile and efficient. You can build on top of your IT architecture, whether it's assets, mainframes, data and databases or big data stores, legacy middleware that might exist or any custom applications. These assets present very valuable sources of information within the organization. The challenge is to enable the organization, the business units, and your IT organization to leverage this data and present it in an easily consumable manner to the application developers and, ultimately, to the stakeholders who need access to this information in a context-dependent basis. These stakeholders could be your employees, your customers, a general audience, or your partners.

A Layered Approach to APIs

These principles will help you create an easily consumable interface to those application developers that create the touchpoints for the different types of audiences and your assets.

As discussed earlier, in order to function optimally, businesses need to organize their APIs into three different layers.



The system APIs are the APIs that expose the APIs of record. They expose the data information of the assets in a very easily consumable manner. Typically these APIs actually are the one s that change more slowly. At the top, the experience APIs are the ones that change at the fastest pace. That is because applications come to life at a much quicker rate than systems in the back end. And experience APIs expose the data from the back end data sources in exactly the right form for the application or devices in question that need to consume the data. In the middle are process APIs. This is the orchestration and composition that need to take place with system APIs in order to provide that data in a new way using business logic through the experience APIs.

If this layering is implemented using an API designed-first approach, that provides the ultimate agility and efficiency in order to make application developers to make not only API consumers successful but to make the API providers extremely successful with the proper governance that allows this type of solution to scale. When we say API connectivity, we start with the design of the API. Whether that API is actually built on top of existing back ends or whether it's built on top of existing APIs, it is important to start with success in mind. What does a successful interface look like? Once the outcome is imagined, then the API can be simulated.

This allows a rapid integration between API providers and API consumers in order to refine that interface definition to get it to the right point. This can be further

validated through the automatic creation of portals and then, finally publishing of that API without any implementation so far, but for it to be actually usable so that the API consumers and the API providers can now go off in parallel to implement the API as well as create the application on top of the APIs at the same time in order to gain maximum efficiency.

Once the interface has been defined, the API developers can implement the APIs. In most cases the implementation of APIs actually doesn't come down to new business logic, but rather to the orchestration of existing business logic, or connectivity to existing business logic - in other words, integration. That mocked API can be realigned by making an implementation very easily within the uniform platform by having that interface lead to first a skeleton implementation as based off the interface. And then it can be implemented the different resource methods and wired up with the right orchestration in the middle with the different back ends.

Setting up API Management and Governance

The right tools can help provide visibility to all stakeholders while supporting the day-to-day operations of the API ecosystem. They can help to rein in any complexity and confusion. If you are an API consumer for example, things like infrastructure, application, and version level changes are important. One day you're using a company's APIs and everything is fine. The next day the integration suddenly stops working. This is not an uncommon occurrence.

Has something changed on your side? Is there a problem with your application, your network, or your data center? Or, has the API provider updated the API or the systems it connects to? And, without the right tools, answering these questions can take a very long time. It's also important to know who within your organization is using APIs. Who's accessing paid services and subscriptions and how much is it costing? Connections by unauthorized users can prove to be quite expensive. It's also important to clearly understand which users have access to the information provided by the APIs you consume.

If you're in healthcare, for example, you probably don't want to give every user in your organization access to patients' health records that come from external health service providers. Without the right tools in place, issues such as these can definitely harm the business, and tools help ensure that this doesn't happen. API providers have many of the same concerns, but they have additional issues as well. One is managing the API life cycle. As your applications and data schemes change, for example, your APIs will need to change as well. This means you often have an API life cycle operating alongside your existing application development life cycle.

Both must be in sync to ensure that API changes reflect the latest changes in application code and data scheming. Access control becomes increasingly important particularly if APIs you provide are accessing production data, which is almost always the case. Security becomes a sensitive issue when anyone with a cell phone starts to access your mainframe. In addition, performance and availability management capabilities are an essential function for API providers, but this is often overlooked. Integrated applications have multiple moving parts and performance or availability problems with even one component can impact the entire delivery chain. This means that if I'm a customer connecting to your internal application using your API, your performance problems suddenly become my performance problems as well. So while, in the past, down time may have cost your company so many dollars per hour, now your downtime is costing me time and money as well.

There are other advantages to good API management and governance as well. First, it ensures that API use complies with organizational and governmental standards because the compliance standards in a lot of industries are very strict. Also, good API governance and management can ensure that all versions stay in sync. It's also advisable to have a single point of control for all things API related. On this point, many companies today have API managers who can either be part of IT or lines of business, who oversee and coordinate multiple aspects of API utilization. Coordination of factors such as security, access and user management is important to mitigate the risks associated with providing or consuming the wrong data. Post planning is also important.

API Management Tools

How can you choose the right tools to make sure your APIs are well managed and governed? First, look to find where your gaps are. Are you providing or consuming APIs, or are you doing both? Your tools needs will vary depending on which is the case for you. In addition to the security and user related issues, can you integrate data from your API focused tools with that of other enterprise data management solutions such as application performance and life cycle management products?

Application performance integrations are particularly important for providers for multiple reasons. Capacity planning becomes a concern as API usage accelerates as well. Wildly successful APIs present a good news/bad news scenario. It's nice to be successful, however successful APIs also need higher rates of connectivity and higher levels of resource utilization on your end. As your company becomes increasingly reliant on APIs, it becomes increasingly important to make sure that you can adequately support some of these issues.

A single platform to manage API-led connectivity has a number of advantages. It's much more effective and efficient to manage and govern APIs if the capabilities - design, implementation, monitoring, operational monitoring, etc. - are accessible under a single platform. For example, imagine that suddenly there's an operational alert that shows that there are SQL errors going on within the database systems API. Well, that's probably a cause of alarm for many reasons, one of which could be that there is security threat. Therefore, the first thing to do work through the API management layer using the same identity and the same interface and the same organizational structure. You can apply a SQL injection policy and protect the interface as a first step. You can then go ahead to the portal for that API and notify all users that this might have happened, and notify the API owner that this might have happened.

Then you can actually start testing the fact that that policy has taken place through the API console within the portal. You can make sure that you're applying policy violation policies on the analytic side to make certain you're getting notified whenever that policy is violated so you can further investigate. This is just one use case that brings the implementation, the monitoring, and the consumption aspects of an API level together. If you had three different systems that you had to go to to address this scenario, it could be much more complicated.

Conclusion

APIs can provide a wide variety of business benefits. They can enable a company to become more efficient in delivering new products and services. They provide a simpler way to integrate compared to the custom integration and complicated metaware of the past. They enable businesses to become more agile and flexible in their interactions with customers, partners, and suppliers. They help support modernization, enabling companies to very quickly adapt to changing technology and business requirements. But like any powerful tool, you have to make sure you're ready to use APIs. Failure to plan for the processes, tools and responsibilities which APIs entail can be a fatal mistake. Planning your API strategy well can help maximize the benefits of APIs while minimizing potential risks.